European Journal of Operational Research 207 (2010) 557–565

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution

M.N.M. Kahar [a,b,*], G. Kendall [a]

[a] Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
[b] Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang, Locked Bag 12, Kuantan 25000, Pahang, Malaysia

ARTICLE INFO

Article history:
Received 15 July 2009
Accepted 8 April 2010
Available online 14 April 2010

Keywords:
Optimisation
Timetabling
Heuristic
Scheduling

ABSTRACT

This paper presents a real-world, capacitated examination timetabling problem from Universiti Malaysia Pahang (UMP), Malaysia. The problem has constraints which have not been modelled before, these being the distance between examination rooms and splitting exams across several rooms. These constraints provide additional challenges in defining a suitable model and in developing a constructive heuristic. One of the contributions of this paper is to formally define this real-world problem. A further contribution is the constructive heuristic that is able to produce good quality solutions for the problem, which are superior to the solutions that are produced using the university's current software. Moreover, our method adheres to all hard constraints which the current systems fails to do.

1. Introduction

Examination timetabling problems can be categorised as either un-capacitated or capacitated. In the un-capacitated examination timetabling problem, room capacities are not considered, whilst in the capacitated problem the room capacities are treated as a hard constraint, in addition to the other commonly used hard constraints, e.g. a clash-free timetable (Pillay and Banzhaf, 2009; Abdullah, 2006). Most of the research in the literature has investigated the un-capacitated examination timetabling problem, concentrating on the algorithm and algorithmic performance in terms of producing solutions effectively and quickly (see Burke and Petrovic, 2002; Qu et al., 2009). In enabling comparisons to be made among the research community, a benchmark dataset proposed by Carter et al. (1996b) is often used. Although un-capacitated benchmark datasets are popular, McCollum (2007) and Carter and Laporte (1996a) believe that, researchers are not dealing with all aspects of the problem. That is, they are only working on a simplified version of the examination problems. Qu et al. (2009), in their survey paper, reveal that most research only addresses a few common hard constraints. For example, no exams with common students assigned simultaneously, the size of exams need to be below room capacity etc. Commonly used soft constraints include spreading conflicting exams as evenly as possible, or not in x consecutive timeslots or days.

The capacitated problem more closely reflects the real-world problem as it includes a room capacity constraint. However, the capacitated problem has received less attention from the research community. This is probably due to the lack of benchmark datasets. In addition, the capacitated problem is much harder to solve; see Burke et al.'s (1996a) survey paper where 73% of the universities agree that accommodating exams is a difficult problem. Capacitated problems also require more comprehensive data as they have to include the room capacity as well as the other data also required for the less complex problem (e.g. student and exam list). This extra information can be difficult to collect (McCollum, 2007).

This paper presents a capacitated examination problem drawn from a real world example from Universiti Malaysia Pahang (UMP). This dataset has several new constraints in addition to those commonly used. In Section 2, we describe the examination timetabling problem and present related work. A description of the UMP examination timetabling problem, including the constraints, is discussed in Section 3. A formal model of the problem is presented in Section 4. In Section 5, we describe the experimental setup for our proposed constructive heuristic. In Section 6, a comparison between the solutions achieved with the current method employed by Universiti Malaysia Pahang (which is produced using a proprietary system), and our method, is presented in order to evaluate the effectiveness of the proposed methodology. In Sections 7 and 8 we summarise the contribution and present our conclusions.

* Corresponding author at: Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK.
E-mail addresses: mnm@cs.nott.ac.uk, mnizam@ump.edu.my (M.N.M. Kahar), gxk@cs.nott.ac.uk (G. Kendall).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.04.011

2. Background

2.1. The examination timetabling problem

The university timetabling problem can be divided into two categories: course timetabling (de Werra, 1985; van den Broek et al., 2009) and exam timetabling (Burke et al., 2007). This paper concentrates on examination timetabling. The construction of an examination timetable is a challenging task and is quite often time consuming. The examination timetabling problem is concerned with assigning exams to a specific, or limited, number of timeslots and also assigning rooms so as to satisfy a given set of constraints. The constraints that contribute to the complexity of an examination timetable can be divided into two categories; hard and soft. Hard constraints cannot be broken and a timetable is considered feasible if all the hard constraints are satisfied. An example of a hard constraint is that no student should be required to sit two examinations simultaneously (i.e. the timetable should be clash free). Soft constraints, on the other hand, are requirements that are not essential but should be satisfied as far as possible. The soft constraints are therefore used to evaluate the quality of the solutions, which can be done by associating a weighted penalty value with each violation of the soft constraint. An example of a soft constraint is to spread exams as evenly as possible, from the perspective of individual students, throughout the exam period. A list of commonly used constraints is given in Qu et al. (2009), Merlot et al. (2003), Burke et al. (1996a).

In some situations, the problem becomes more difficult as constraints conflict with each other where satisfaction of one constraint can lead to the violation of another (Qu et al., 2009). For example, suppose we have a situation where we want to minimise the total examination period and at the same time we wish to spread out exams as much as possible. In such a situation, satisfaction of the first constraint will inevitably lead to poorer quality solutions of the second constraint, and vice versa. Moreover, examination timetabling becomes more challenging as the number of student enrollments, courses and combined courses increases. In addition, room constraints add even more complexity to the problem.

The examination timetabling problem varies from one institution to another (Burke et al., 1996a). Every institution has a different set of requirements in order to effectively utilise their resources, meet the requirements of their business, provide a high level of satisfaction to their students etc. Therefore, an examination timetabling system has to be built to meet these individual requirements. As evident from the literature a large number of papers published on variants of the timetabling problem have some common objectives. Examples of these are:

- Minimise the number of timeslots.
- Spread conflicting exams within the given examination period.
- Minimise students sitting exams in two consecutive timeslots or sitting two exams on the same day.

A discussion on the variants of the problem can be found in Qu et al. (2009).

Using the UMP dataset, this paper considers a real-world examination timetabling problem which has not previously been investigated in the scientific literature. The objectives (i.e. soft constraints) include the distance between rooms of an exam being held in multiple rooms and the minimisation of the number of rooms an exam is split across. We present further details in Section 3.

2.2. Related work

In the examination timetabling research community, the most commonly used datasets are those from Toronto (Carter et al., 1996b), Nottingham (Burke et al., 1996b) and Melbourne (Merlot et al., 2003). The introduction of these datasets has motivated researchers to develop many approaches. Recently the International Timetabling Competition (ITC2007) dataset has been introduced (McCollum et al., 2008) which includes more realistic problems than the Carter benchmark problems. Other examination datasets also exist, for example from UKM (Ayob et al., 2007) and UiTM (Kendall and Hussin, 2004). The Toronto and UiTM datasets are un-capacitated problems whilst the Nottingham, Melbourne, ITC2007 and UKM datasets are capacitated problems. The Toronto dataset consists of thirteen real-world exam timetabling problems from eight Canadian institutions, one from the London School of Economics, one from King Fahd University, Dhahran and one from Purdue University, Indiana (Carter et al., 1996b). The dataset does not allow any student from sitting two exams at the same time. The sole objective is to evenly spread out the exams within the allocated number of timeslots. The Toronto dataset has received the most research attention. Many papers, which use this dataset, can be found in the PATAT conference series of selected papers (i.e. Burke and Ross, 1996; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007), as well as in other papers such as Qu et al. (2009) and Burke et al. (2007).

In 1996, Burke Newall and Weare introduced the examination timetabling dataset from the University of Nottingham. The dataset includes a maximum number of exam sessions (23 timeslots) and imposed clashing and total capacity constraints. The objective of this dataset is to minimise the number of instances of a student having two exams in a row on the same day and overnight. Several researchers have investigated this dataset including Di Gaspero and Schaerf (2001), Caramia et al. (2001), Burke and Newall (1999) and Merlot et al. (2003). A dataset from the University of Melbourne was introduced by Merlot et al. (2003). They introduced two different datasets which include two timeslots on each day for each of the five workdays, and a varying capacity for each session. The datasets also includes exam availability constraints, which involve a restriction on some exams in some sessions (exams are pre-assigned to specific sessions or can only be held in a limited set of sessions). Researchers who have investigated this dataset include Merlot et al. (2003) and Cote et al. (2005).

Ayob et al. (2007) introduced a capacitated dataset from UKM, Malaysia. The dataset requires that all exams be scheduled and be scheduled only once. They forbid any student taking two exams at the same time and from sitting three consecutive exams in a day. The dataset also includes assigning exams to specified rooms and those students assigned to sit consecutive exams must be assigned to the same room. The objective involves evenly spreading the exams and minimising students having consecutive exams on the same day.

Kendall and Hussin (2004) introduced a dataset from UiTM Malaysia. The dataset requires that all exams must be scheduled and that some exams are required to be scheduled together in the same timeslot. No student should sit for more than one exam in the same slot. The objective is to spread exams as evenly as possible, which is calculated using the proximity value from Carter et al. (1996b), and penalising exams that are scheduled during the weekend.

The first international timetabling competition was established in 2002 with the aim of creating a platform for researchers to test their algorithms on real-world timetabling problems. ITC2007 (the second competition) has the following constraints; no student sits more than one exam at the same time and the exams should not exceed the room capacity. An exam assigned to a timeslot should not violate the timeslot lengths and the exams need to be arranged as specified (for example, assign examA after examB, examA must use room 15 etc.). The objective is to minimise the number of students sitting two exams in a row on the same day, minimise the

number of students sitting two exams in a day, minimise mixed duration of exams within a timeslot, minimise the usage of a particular timeslot or room and schedule larger examinations as early as possible. The details of the examination competition track can be found in McCollum et al. (2008) and McCollum et al. (2010). Researchers who have investigated this dataset include McCollum et al. (2009) and Gogos et al. (2008).

Table 1 summarises the constraints of datasets discussed above. For the capacitated datasets discussed above, Nottingham and Melbourne are concerned with the total seating capacity. That is, the total number of students sitting all exams, in the same timeslot, must be less than some specified number. According to Merlot et al. (2003), this represents a simplified problem whereas normally in solving a real-world problem, we would have to take into account individual room capacities, but this obviously depends on institutional requirements. UKM and ITC2007 specify individual room capacities.

Based on the datasets described above (see Table 1) and the other constraints listed in the literature (Burke et al., 1996a, Qu et al., 2009), we note that there is a gap in terms of the examination timetabling datasets from the literature and many of the requirements faced by many institutions. The UMP examination timetabling problem contains additional constraints which consider individual room capacities, whilst not allowing rooms to be shared by multiple exams (unless exams are combined, where they are treated as one exam). In addition, UMP also has a distance penalty cost (applied when an exam is split across more than one room for a given exam) and a splitting penalty cost (as it is favorable to use only one room) in the room assignment. A further discussion on the UMP examination timetabling problem is presented in the next section.

The solution approaches seen in literature for the exam timetabling problem can be separated into *exam-timeslot assignment* and *exam-classroom assignment*. The most published work seen in the literature is the exam-timeslot assignment. Only a few works discuss exam-classroom assignment (Dammak et al., 2006). Both the un-capacitated and capacitated (as total seating capacity) problem (i.e. benchmark dataset) can be solved using a two-phase approach, as both allow more than one exam in an examination room. This will provide a feasible solution in the exam-classroom assignment phase as long as the capacity of rooms is greater than the number of students (Dammak et al., 2006). However, if individual room capacities are used, that prohibiting having more than one exam in a classroom, it does not guarantee that we are able to find a feasible solution using the two-phase approach. We might even need to introduce a solution repair mechanism in order to arrive at a feasible solution. Therefore, in this problem, we are going to solve the UMP examination timetabling problem sequentially as an *exam-timeslot-classroom assignment*.

3. Universiti Malaysia Pahang (UMP): Examination timetabling problem

The Universiti Malaysia Pahang (UMP), formerly known as Kolej Univerisiti Kejuruteraan dan Teknologi Malaysia (KUKTEM), was established in 2002 and is located in Pahang, Malaysia. In 2007, UMP consisted of five faculties with a total of 3550 students. The number of students is growing rapidly as new faculties are being introduced along with an increase in the programs offered. Cur-

Table 1
Summary of datasets.

	Constraints	Toronto	Nottingham	Melbourne	UKM	UiTM	ITC2007	UMP
Examinations	Clash free	Hard	Hard	Hard	Hard	Hard	Hard	Hard
	Scheduled all exams		Soft	Soft	Hard	Hard		Hard
	Weekend scheduled					Soft		
	Exam preference - *specified arrangement: sa* - *specified room: sr* - *large exam schedule first: lf* - *restriction on exam in particular timeslot: rt* - *scheduled combined exam in the same timeslot: ct*			Hard *(rt)*		Hard *(ct)*	Hard *(sa)* Soft *(lf)*	
	Consecutive exam - *two exam in a row: 2r* - *two exam in a day: 2d* - *two exam in a row overnight: 2n* - *three exam in a day: 3d*		Soft *(2d & 2n)*	Soft *(2d & 2n)*	Hard *(3d)* Soft *(2r)*		Soft *(2r and 2d)*	
Timeslot related	Timeslot preference - *minimise/avoid usage: tu*						Soft *(tu)*	
	Timeslot length - *mixed duration of exams in one timeslot: mt*						Hard Soft *(mt)*	
	Spreading - *specified spread: ss*	Soft	Hard *(ss)*	Soft	Soft	Soft	Soft *(ss)*	Soft
Rooms related	Room distance							Soft
	No sharing of room with other exams - *for specified exam only: se*				Hard *(se)*			Hard
	Room preference - *consecutive exam scheduled in the same room: cr* - *minimise/avoid usage: ru* - *specified room: sr*				Hard *(cr)*		Hard *(sr)* Soft *(ru)*	
	Split exam into different rooms - *same building only: sb* - *as close as possible: cp*							Hard *(sb)* Soft *(cp)*
	Capacity - *total seats: ts* - *individual room: ir*		Hard *(ts)*	Hard *(ts)*	Hard *(ts and ir)*		Hard *(ir)*	Hard *(ir)*

Hard = hard constraint; Soft = Soft constraint; shaded cell = constraint not considered.

rently, a total of 17 programs are offered by these faculties. UMP is currently situated in a temporary campus, which presents many challenges in terms of available space, logistics and the human resources to manage the process.

In addition to these limitations, the UMP examination timetable problem has other challenging constraints which have never been tackled before in the literature, at least, as far as the authors are aware.

In UMP, the Academic Management Office is responsible for planning and managing the entire academic process. It provides all the academic space and facilitates academic affairs. All this is done with the aid of an Information Management System (IMS). This system encompasses a complete student life cycle process; from student intake to graduation. One of the modules in the IMS includes generating an examination timetable which has been used since 2003. However, although this proprietary system has been successful in producing the examination timetable, it involves manual processes in order to achieve a feasible solution. Moreover, the proprietary system is unable to determine the quality of the solutions it produces due to having no mathematical model (that we are aware of) that allows the quality of the generated timetable to be gauged. Therefore, one of our research objectives is to develop a formal model in order to evaluate the effectiveness of the solution produced by the proprietary system and thus enable a comparison with other methods, one of which we present in Section 5.

The hard constraints for the UMP examination timetable problem are as follows:

- H1: No student should be required to sit two examinations simultaneously.
- H2: The total number of students assigned to a particular room(s) must be less that the total room capacity.
- H3: Only one examination paper is scheduled to a particular room. That is, there is no sharing of rooms with other exam papers (even if enough seats are available to fit in another exam). However, some exams can be combined with others, for the following reasons:
  - The same examination for different academic programs.
  - Lecturers request exam papers to be combined. This might be caused by a lecturer teaching different courses but with similar content.
  - Faculties request that exams are combined. The combined exam papers might contain similar or identical questions.

  The request for combining exams is done before the exam schedule is generated. The combined exams are given a unique examination code and treated as one large exam.
- H4: The size of each exam room in UMP is relatively small (less than 100) and with a large number of registered students for each exam, this inevitably leads to splitting exams across different rooms. In splitting the exam we need to allocate the rooms as close as possible to each other (this actually represents a soft constraint, see below) but the rooms MUST be in the same building (a hard constraint).

In measuring the quality of the solution, the soft constraints are as follows:

- S1: Each set of student examinations should be spread as evenly as possible over the exam period.
- S2: The distance between exam rooms, for the same exam, should be as close as possible to each other (and within the same building, see hard constraint above).
- S3: There is a penalty associated with splitting an exam across several rooms, as we would like an exam to be in a single (or as few as possible) room whenever possible.

At UMP we cannot solve the capacity problem as a total seating capacity (i.e. as one large exam room) due to having a constraint that does not allow an exam to share a room with another exam. Therefore each room has a specific seating capacity, which must be respected. Therefore, the problem needs to be solved as an *exam-timeslot-classroom assignment* where the timeslot and room need to be selected sequentially in the searching process.

4. Problem formulation

In this section, we present the formal model of the UMP examination timetabling problem as discussed in Section 3.

Indices

i,j	$1\ldots N$
r,p	$1\ldots R$
s	$1\ldots S$
t	$1\ldots T$

Parameters

N	The number of examinations
R	The number of examination rooms
S	The number of students
T	The number of available timeslots
S_i	The number of registered students in exam i
R_t	The number of examination rooms available at timeslot t
B_r	The building for room r
f_r	The total capacity for room r
c_{ij}	The conflict matrix where each element $(c_{ij}, i,j \in \{1\ldots N\})$ is the number of students that have to take both exam i and j. The conflict matrix is a symmetrical matrix of size N, where diagonal elements $c_{ii} = S_i$
d_{rp}	The distance matrix where each element (denoted by $d_{rp}, r,p \in \{1\ldots R\}$) is the distance between rooms r and p. The distance matrix is a symmetrical matrix of size R, where diagonal elements $d_{rr} = 0$

Decision variables

x_{it}	1 if examination i is assigned to timeslot t, 0 otherwise
y_{ir}	1 if examination i is assigned to room r, 0 otherwise
z_{rt}	1 if room r is assigned to timeslot t, 0 otherwise

The objective is to spread out examinations over the exam period (timeslots) for each student, minimise the distance between rooms of an exam that is being held in multiple rooms and to minimise splitting an exam over several rooms. Therefore our formulation is as follows:

$$\text{(Minimise)}\ (F(x) = F_1 + F_2 + F_3. \tag{1}$$

The first component of the cost, F_1 (spreading exams over the exam period) is shown in Eq. (2).

$$F_1 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} \cdot \mathit{proximity}(t_i, t_j)}{2S}, \tag{2}$$

and

$$\mathit{proximity}(t_i, t_j) = \begin{cases} 32/2^{|t_i - t_j|} & \text{if } 1 \leqslant |t_i - t_j| \leqslant 5 \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where t_i and t_j specify the assigned timeslot for examination $i$and j ($i, j \in \{1,\ldots,N\}$). Eq. (2) represents the cost for an exam i that is given by the proximity value multiplied by the number of students in conflict. Proximity values of 16, 8, 4, 2 and 1 are used here. For example, if a student has two consecutive examinations then a proximity value of 16 is assigned. If a student has two examinations, with a free timeslot in between, then a value of 8 is assigned. Two empty periods correspond to a penalty of 4 and so on. These proximity values were introduced by Carter et al. (1996b) and have been

widely used by other researchers (see Burke et al., 2004; Ayob et al., 2007; Abdullah, 2006).

The second component of the cost, F_2 (distance of an exam in multiple rooms) is shown in Eq. (4):

$$F_2 = \frac{\sum_{i=1}^{N}\sum_{r=1}^{R-1}\sum_{p=r+1}^{R} d_{rp} y_{ir} y_{ip}}{N}. \quad (4)$$

Eq. (4) represents a cost for an exam i that is scheduled in multiple rooms. A subset of the distance matrix is shown in Fig. 2.

The third component of the cost, F_3 (splitting exam) is shown in Eq. (5):

$$F_3 = \frac{\sum_{i=1}^{N} m_i - 1}{N}, \quad (5)$$

where m_i is the number of rooms exam i has been split across. It can be calculated using the following formulation, $m_i = \sum_{r=1}^{R} y_{ir} \forall i \in \{1 \ldots N\}$. Eq. (5) represents a cost for an exam i that is being penalised for splitting the exam in multiple room ($m_i > 1$). For example, if an exam is being split into two rooms, then a value of 1 is given as the penalty value. Splitting the exam across three rooms corresponds to a penalty of 2 and so on.

Eq. (1) is subject to the following constraints:

a) No student can sit two exams concurrently (clash-free requirement). If examination i and j are scheduled in timeslot t, the number of students sitting both examination i and j must be equal to zero, i.e. $c_{ij} = 0$. This hard constraint is expressed in Eq. (7):

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T} x_{it} x_{jt} c_{ij} = 0. \quad (6)$$

b) All exams must be scheduled and each exam must be scheduled only once in available timeslots, T.

$$\sum_{t=1}^{T} x_{it} = 1 \quad \text{for all } i \in \{1, \ldots, N\}. \quad (7)$$

c) Only one examination paper is scheduled to a particular room in a particular timeslot. There is no sharing of rooms with other exam papers (even though seats might be available to fit in another exam), except for requested combined exams, which has been carried out as a pre-process operation.

$$\sum_{i=1}^{N} x_{it} y_{ir} = z_{rt} \quad \text{for all } t \in \{1, \ldots, T\} \quad \text{and for all } r \in \{1, \ldots, R\}. \quad (8)$$

d) Exam can only be split across several rooms in the same building.

$$\sum_{r=1}^{R-1}\sum_{p=r+1}^{R} y_{ir} y_{ip} b_{rp} = \frac{m_i(m_i - 1)}{2} \quad \text{for all } i \in \{1, \ldots, N\}, \quad (9)$$

where

$$b_{rp} = \begin{cases} 1 & \text{if } (B_r = B_p) \\ 0 & \text{otherwise} \end{cases}.$$

e) For each timeslot t the number of rooms assigned to a particular timeslot must not exceed the maximum number of rooms available in a timeslot, R_t

$$\sum_{r=1}^{R} z_{rt} \leqslant R_t \quad \text{for all } t \in \{1, \ldots, T\}. \quad (10)$$

f) The total number of students assigned to a particular exam room(s) must be less than the total room capacity.

$$S_i \leqslant \sum_{r=1}^{R} y_{ir} f_r \quad \text{for all } i \in \{1, \ldots, N\}. \quad (11)$$

5. Experimental setup

In this section we present our proposed constructive heuristic, along with other algorithmic details to aid reproducibility. The dataset is taken from Universiti Malaysia Pahang (UMP) for semester 1, 2007. The total number of examination papers is 252, across the 17 programs offered by 5 faculties. However, due to the combined exams requirement, the dataset has been pre-processed and the combined exams are given a new examination code and treated as one large exam. This results in a total of 157 examinations. By combining these exams, it helps to minimise and optimise the usage of the rooms. The total number of students is 3550 with 12,731 enrolments. The conflict matrix density is 0.05, meaning that 5% of students are in conflict among the examinations paper. The number of exam days and timeslots are 10 and 20, respectively. There are only 2 timeslots on each examination day. There are no exams during the weekend (Saturday and Sunday). We capture this by introducing gaps in our timeslots indices. Therefore the timeslots can be represented as shown in Fig. 1.

In Fig. 1, timeslot 1 and 2 refer to day 1, timeslot 3 and 4 refer to day 2 etc. Notice that indices 11 to 14 are missing. This is because those indices refer to Saturday and Sunday.

The total available exam space for this dataset is 24 rooms, with each room having a given capacity. To assist our constructive heuristic in the process of searching for the most suitable room(s) and minimising the room related cost value, we generate a list of room groupings (based on the list of rooms provided). These pre-determined room groupings are generated within the same building only. Note that we limit the room groupings up to a maximum of 4 possible rooms for each exam. In our observations, 4 rooms are adequate to satisfy any exam capacity. Besides, increasing the room grouping possibilities (>4) will obviously increase the distance cost, splitting cost and the search space. The room groupings are sorted in decreasing order based on the total room(s) capacity. By doing so we could directly search for suitable room(s) and end the search procedure when an unsuitable room capacity is encountered.

To illustrate the procedure we provide the following example. Assume, we have five rooms in two different buildings, where four of the rooms are in the same building, and each room has a specific capacity (see Fig. 2). The travel cost for rooms in different buildings is not shown, as this is not permitted. Therefore, we could create 15 room groupings with 14 room groupings from building W and 1 room grouping from building X. Referring to Fig. 3, each of the room groupings have their new total capacity, distance cost (total of the distance value prior to the distance matrix for every rooms) and splitting cost ($m_i - 1$). These room groupings are sorted based on their capacity.

Having the decreasing order of pre-determined room groupings assists the search algorithm in selecting the most suitable rooms, aiming to minimise the room related cost value and speeding up the search by stopping the room search procedure if an unsuitable room grouping capacity is encountered.

The experiments are conducted using graph heuristic approaches including Largest Degree (LD), Largest Weighted Degree (LWD), Saturation Degree (SD) and Largest Enrolment (LE). The description of these methods is presented below:

– Largest degree (LD): this heuristic takes the exams that have the most conflicts with other exams and schedules them first.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24)

Fig. 1. Timeslot indices.

Room	Capacity	Building	WDK26	WDK28	WDK29	WDK30	XDK04
WDK26	92	W	0	2	3	4	-
WDK28	90	W	2	0	1	2	-
WDK29	40	W	3	1	0	1	-
WDK30	40	W	4	2	1	0	-
XDK04	47	X	-	-	-	-	-

Fig. 2. Room information and distance matrix.

No.	Room Grouping	Room Grouping Capacity	Distance Cost	Split Cost
1	WDK26 - WDK28 - WDK29 - WDK30	262	13	3
2	WDK26 - WDK28 - WDK29	222	6	2
3	WDK26 - WDK28 - WDK30	222	8	2
4	WDK26 - WDK28	182	2	1
5	WDK28 - WDK29 - WDK30	170	4	2
6	WDK26 - WDK29	132	3	1
7	WDK26 - WDK30	132	4	1
8	WDK28 - WDK29	130	1	1
9	WDK28 - WDK30	130	2	1
10	WDK26	92	0	0
11	WDK28	90	0	0
12	WDK29 - WDK30	80	1	1
13	XDK04	47	0	0
14	WDK29	40	0	0
15	WDK30	40	0	0

Fig. 3. Decreasing order of pre-determined rooms grouping.

- Largest weighted degree (LWD): this heuristic is similar to largest degree except that it takes exams that have the most number of students who are involved in the conflict and schedules them first.
- Largest enrolment (LE): this heuristic takes exams with the largest number of registered students and schedules them first.
- Saturation degree (SD): this heuristic chooses exams which have the least number of available periods in the timetable that can be selected and schedules them first.

In general, the algorithm (see Fig. 4) starts (line 2) by sorting the examinations based on a graph colouring heuristic (e.g. LD, SD, etc.) and also sorting the room groupings *G* in decreasing order based on total room(s) capacity. For all examination `i`, (step 2) we randomly select a timeslot `t` (the number of timeslots we consider is referred to as a candidate list, and we show the effect of different candidate list sizes in the results section), which is clash free and we only accept `t` if it is not equal with any *t* previously generated in `C` and the total available seating capacities in timeslot `t` (`totalSeatAvailable (t)`) able to accommodate exam `i` (`capacity (i)`). If the total available seating capacities in `t` is greater or equal to exam `i` (capacity (`i`) $\leqslant$`totalseatAvailable(t)`), we will continue to calculate the spreading penalty based on the selected timeslot and store it in `spreadCost[c]`. The `spreadCost[c]` value will be used later in selecting the timeslot and room with the minimum cost values (line 24). However, if the total available seating capacities unable to accommodate the exam, the search will continue to look for other `t` until a number of `count` trials. Here we set a maximum of three trials. If after a number of `count` trials the algorithm still could not find a feasible `t`, then the search will proceed with the next `c`. Otherwise, it will continue with the room assignment which goes through the room groupings `G`. Selections of `g` is based on its capacity. If room grouping `g` able to accommodate exam `i`, an availability check on the individual room(s) in the `g` is carried out and if the exam `i` can be accommodated, the room distance and the splitting penalty within the room(s) in room grouping `g` is calculated as `distPenalty (g)` and `splitPenalty (g)`), respectively. These values are compared with the distance cost (`distCost[c]`) and the splitting cost (`splitCost[c]`) in `c`. The value of these arrays, `distCost[c]` and `splitCost[c]`, are overwritten if the distance and splitting costs are minimum. Otherwise, we will continue to search for other rooms in `G`. Once, room(s) in `g` been selected, we select the minimum cost value found by comparing each of the `spreadCost`, `distCost` and `splitCost` in `C` and set the decision variable to 1. The algorithm will continue the search for all exam *i*. Lastly, we verified the solution by checking the solution to ascertain that the timeslot and rooms found satisfied the constraints and calculate the cost value.

Algorithm Parameters:

- `i` = 1...`N` where `N` is the number of examinations
- `g` = 1...`G` where `G` is the number of pre-determined *roomGrouping*
- `r` = 1...`R` where `R` is the number of rooms
- `c` = 1...`C` where `C` is the candidate list size
- `t` = 1...`T` where `T` is the number of timeslots
- `totalSeatAvailable(t)` is the total seating capacity available calculated in timeslot `t`
- `capacity(i)` is the size of examination `i`
- `spreadCost`[c] store the spreading penalty for candidates list `c`
- distCost[c] store the room distance penalty for candidates list `c`
- `splitCost`[c] store the splitting room penalty for candidates list `c`
- `roomCapacity`(*g*) is the total room seating capacity in `g`,
- `distPenalty` is the room distance penalty in *g*
- `splitPenalty` is the splitting room penalty in *g*
- x_{it}=1 if examination *i* is assigned to timeslot `t`, 0 otherwise
- y_{ir}=1 if examination `i` is assigned to room `r`, 0 otherwise
- z_{rt}=1 if room *r* is assigned to timeslot `t`, 0 otherwise

```
1   Step 1: Ordering:
2      Sort examination N based on the Graph Colouring heuristic;
3      Sort roomGrouping G in decreasing order based on the total capacity;
4   Step 2: Assigning exams to timeslot and room(s):
5      Set i ← 1;
6      Until i = N, repeat:
8      (2.1) Set c ← 1;
9      (2.2) Until c = C, repeat:
10       (2.2.1) Set count ← 0, g ← 1 and t ← -1;
11       (2.2.2) until t = -1 && count < 3, repeat:
12               (a) Generate t randomly and no clashing with other exams
13               (b) If t is not equal with t previously generated and capacity(i) ≤
14                     totalseatAvailable(t), Then, calculate spreading penalty as spreadCost[c]
15                     Otherwise, Set t ← -1 and increase count;
16       (2.2.3) Set distCost[c] ← +∞  and splitCost[c] ← +∞;
17       (2.2.4) Do the following if t ≠ -1:
18               (a) Until capacity(i) ≤ roomCapacity(g), repeat:
19                  (i)  If room g is available and distCost[c]+splitCost[c] >
20                         distPenalty(g)+splitPenalty(g), Then, set distCost[c] ← distPenalty(g),
21                         splitCost[c] ← splitPenalty(g)
22                  (ii) Increase g
23       (2.2.5) Increase c;
23     (2.3) Select the minimum total cost value from C and set x_it←1 y_ir←1 and z_rt←1, if t ≠ -1
24             for every c
25     (2.4) Increase i
26  Step 3: Verification
27     Check the solution prior to the constraints
28     Calculate the cost value
```

Fig. 4. Pseudo-codes for the examination timetable.

5.1. Discarding moves sub-algorithms

The algorithm is able to find superior solutions, compared to the proprietary software, in a small amount of computational time. This is done by discarding unnecessary moves as early as possible in the algorithm. Referring to the algorithm (Fig. 4), the discarding move algorithms are as follows, and we present them here to assist in reproducibility:

(a) Total available seating capacities in timeslot `t` (lines 13–15). Line 13–15 check the room availability prior to timeslot `t` is generated. It calculates the total available seats in `t`. If the total available seats are unable to accommodate exam `i` (see line 15), then a new clash-free timeslot `t` is generated. Having to calculate the total available seating capacities would avoid the search from selecting an inappropriate timeslot. It is good to recognize that we don't have enough room early in the search, rather than at the end, in order to make effective use of the computational time available.

b) Room grouping capacity checking (line 18).
In line 18 the algorithm will check whether the room grouping `g` (start at `g` = 1) able to accommodate exam `i`. If the condition is TRUE, the algorithm will continue to determine whether each room in `g` (line 19) is available or otherwise it will look for other `g` in the list. The room grouping search will stop once an unsuitable room grouping capacity is found (as we have already sorted the room grouping in descending order) as it will only consume computational time if the search in the `G` is continued.

c) Determine room availability in `g` (lines 19).
Line 19, checks every room(s) in the room grouping `g` to determine whether the room is available or not. This is done by checking z_{rt} (z_{rt}=1 if room `r` is assigned to timeslot *t*, 0 otherwise). If z_{rt}=0 it means that the room is available and the search will continue to check other rooms. However, if z_{rt}=1 which means that the room is unavailable, the algorithm will stop searching the room members in the selected room groupings `g` and continue to select the next suitable room groupings `g`.

d) Selecting minimum value of distance and splitting cost (line 19).
In line 19, the algorithm will only proceed if all the rooms in room grouping `g` are available. Hence, it will compare the distance (`distPenalty (g)`) and splitting penalty (`splitPenalty (g)`) in `g` with the `distCost[c]` and `splitCost[c]`. If these penalty values are less than the current value stored in `distCost[c]` and `splitCost[c]`, we will store this value in `distCost[c]` and `splitCost[c]`.

All of these discarding moves help in finding a feasible solution with minimum cost value compare to UMP proprietary software in a small computational time. In the next section we present our results.

6. Results

In this section, we compare the examination timetable generated by the proprietary software and the result from our proposed algorithm, shown in Fig. 4.

6.1. UMP proprietary software

In the solution generated by the proprietary software for semester I, 2007, prior to the model being developed presented in Section 4, the solution exhibited the following characteristics:

– Based on the five hard constraints stipulated by UMP, the examination timetable that was produced complied with all the constraints except for the *no student should be required to sit two examinations simultaneously* constraint. Eight students were

scheduled to sit exams at the same time and UMP had to quarantine these students.
– As mentioned previously, the quality of the solution is measured based on three objectives. The calculated cost for each of the objectives is $F_1 = 8.82$ for the spreading of exams over the examination period, $F_2 = 3.63$ for the distance of an exam in multiple rooms and $F_3 = 0.71$ for the number of room(s) an exam being split across. The sum of the cost is therefore 13.16. Recall that this includes violation of the hard constraint on the clash-free requirement (eight students).

6.2. Graph colouring heuristic

Using the proposed heuristic, several experiments have been run with different candidate lists. In the context of this work a candidate list is how many timeslots are considered when placing an examination. Each experiment was run 50 times in order to produce average and standard deviation statistics. Every one of the 50 runs produced a feasible solution. The experiments were run on a Pentium core 2 processor. The running time for a candidate list of one is around 99 seconds and 470 seconds for a candidate list of five.

With a candidate list of one ($C = 1$), the algorithm searches randomly for one available timeslot and selects the room grouping that produces the minimum cost value for room distance and the number of splitting rooms. Referring to Table 2, the result using a candidate list of one produces comparable solutions with the proprietary software while adhering to all the constraints. On average, our approach produces a cost value that is higher compared to the proprietary software solution (see Table 2). However, our solutions adhere to all of the constraints compared to proprietary software, which does not. Referring to Table 2 (column min), we are able to produce a solution that is 17% (13.16 compared with 10.98 $((13.16 - 10.98)/13.16 \times 100\%)$) better when compared to the solution produced by the proprietary software. Of the heuristics we have used, largest enrolment (LE) produced the best cost value of 10.98 where the spreading cost is $F_1 = 9.01$, the distance cost is $F_2 = 1.39$ and the splitting cost is $F_3 = 0.58$ with a standard deviation of 2.10. LWD is second best with a minimum cost of 11.43 followed by saturation degree-LE, saturation degree-LD, largest degree (LD) and Saturation degree-LWD. Overall, using a candidate list of one is able to produce a good solution, which adheres to all the hard constraints (unlike the proprietary software).

When using a candidate list of five, the algorithm randomly searches for five available timeslots. For each of the timeslots selected, the algorithm will search the room groupings that give the minimum cost value (distance and splitting cost). Finally, among all the selected timeslot and room(s), we will select the one which produces the minimum total cost value. Referring to Table 2, the result constructed using a candidate list of five produced a solution that is between 15% (13.16 compared with 11.12 $((13.16 - 11.12)/13.16 \times 100\%)$) to 64% (13.16 compared with 4.74 $((13.16 - 4.74)/13.16 \times 100\%)$) better when compared to the proprietary software. Largest Enrollment (LE), again produces the minimum cost value (4.74). Other heuristics perform relatively the same, with respect to their ordering based on their performance, with a candidate list of one. However, with candidate lists of five all heuristics outperform the UMP proprietary software with the minimum spreading cost found being $F_1 = 3.31$, distance cost $F_2 = 0.98$ and splitting cost $F_3 = 0.45$ (produced using LE).

Our proposed algorithm always produces a feasible solution over the 50 runs for candidate lists one and five. LE obtained the best result compared to the other heuristics due to the room related constraints (i.e. distance and splitting constraint). Having those two constraints reduces the effectiveness of SD and LD. This is perhaps not surprising as SD and LD are designed to specifically target spreading the examinations.

7. Statement of contribution

This paper has provided a study of a real-world examination timetabling problem from UMP. In particular, we have investigated the scheduling of exams in a capacitated environment with the aim of minimising the spreading, distance and splitting cost. One of the contributions of this paper is the collection of the necessary requirements (constraints) which has never before been properly documented at UMP. This data collection was carried out with the help and assistance of UMP employees. Studying the problem has led to two new constraints being identified; the travel distance for lecturers/invigilators and splitting exams across rooms. A further contribution of this work is the formulation of the UMP examination timetabling problem as a mathematical model. A simple yet effective approach of single or multiple room searching and selection is introduced through the pre-determined room grouping. Finally, we have presented an algorithm, based on graph colouring heuristics, which we have shown can produce superior solution to the software currently used. In addition, the proposed algorithm adheres to all the hard constraints which the current methodology fails to do.

8. Conclusion and future work

It is recognised that a gap exists between theory and practice in examination timetabling. Different institutions have different requirements and it is difficult to produce a common solution methodology. In this paper we have introduced a new examination dataset and two new constraints. A constructive heuristic has been used to generate solutions that produce better solutions when compared to the proprietary software that is used by UMP. For future work, we plan to schedule the invigilators and include other additional requirements requested by UMP. For example:

– Last minute exam paper addition. In certain cases, lecturers (or faculties) require a last minute addition of exam paper into the exam period.

Table 2
Result using graph colouring heuristic.

Graph colouring heuristic	Candidate list = 1				Candidate list = 5			
	Ave	stdev	min	max	Ave	stdev	min	max
Largest degree (LD)	16.21	**1.52**	12.74	20.42	7.84	0.98	5.99	11.12
Largest weighted degree (LWD)	15.82	1.97	11.43	20.70	6.09	0.67	5.05	8.29
Largest enrolment (LE)	**15.51**	2.10	**10.98**	20.03	**6.06**	0.76	**4.74**	**7.98**
Saturation degree (SD)-LD	16.17	1.53	13.11	**19.39**	7.22	0.84	5.76	8.72
Saturation degree (SD)-LWD	16.29	1.54	13.97	20.41	7.00	1.02	5.49	9.78
Saturation degree (SD)-LE	16.09	1.80	12.66	20.74	6.96	**0.66**	5.28	8.49

Ave = average; var = variance; stdev = standard deviation; min = minimum; max = maximum.

– Invigilator request for change. An invigilator may request a change in the timetable. This may be due to changes of their own schedule or changes to the exam timetable itself.

Acknowledgements

The examination dataset has been provided by the Academic Management Office, UMP and the research has been supported by the Public Services Department of Malaysia (JPA) and the Universiti Malaysia Pahang (UMP).

References

Abdullah S., 2006. Heuristic Approaches for University Timetabling Problems. Ph.D. Thesis, School of Computer Science and Information Technology, University of Nottingham, June 2006.

Ayob, M., Abdullah, S., Malik, A.M.A., 2007. Practical examination timetabling problem at the Universiti Kebangsaan Malaysia. IJCSNS International Journal of Computer Science and Network Security 7 (9), 198–204.

Burke, E.K., Carter, M.W. (Eds.), 1998. Second International Conference on Practice and Theory of Automated Timetabling II, PATAT'97, Toronto, Canada, August 20–22, 1997, Selected Papers. Lecture Notes in Computer Science, vol. 1408. Springer, ISBN 3-540-64979-4.

Burke, E.K., De Causmaecker, P. (Eds.), 2003. Fourth International Conference on Practice and Theory of Automated Timetabling IV, PATAT 2002, Gent, Belgium, August 21–23, 2002, Selected Revised Papers. Lecture Notes in Computer Science, 2740. Springer, ISBN 3-540-40699-9.

Burke, E.K., Erben, W. (Eds.), 2001. Third International Conference on Practice and Theory of Automated Timetabling III, PATAT 2000, Konstanz, Germany, August 16–18, 2000, Selected Papers. Lecture Notes in Computer Science, 2079. Springer, ISBN 3-540-42421-0.

Burke, E.K., Newall, J., 1999. A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation 3 (1), 63–74.

Burke, E.K., Petrovic, S., 2002. Recent research directions in automated timetabling. European Journal of Operational Research 140 (2), 266–280. 16 pp.

Burke, E.K., Ross, P. (Eds.), 1996. First International Conference on Practice and Theory of Automated Timetabling, Edinburgh, UK, August 29–September 1, 1995, Selected Papers. Lecture Notes in Computer Science, vol. 1153. Springer, ISBN 3-540-61794-9.

Burke, E.K., Rudova, H. (Eds.), 2007. Sixth International Conference on Practice and Theory of Automated Timetabling VI, PATAT 2006, Brno, Czech Republic, August 30–September 1, 2006, Revised Selected Papers. Lecture Notes in Computer Science, 3867, ISBN 978-3-540-77344-3.

Burke, E.K., Trick, M. (Eds.), 2005. Fifth International Conference on Practice and Theory of Automated Timetabling V, PATAT 2004, Pittsburgh, PA, USA, August 18–20, 2004, Revised Selected Papers. Lecture Notes in Computer Science, 3616. Springer, ISBN 3-540-30705-2.

Burke, E.K., Elliman, D.G., Ford, P.H., Weare, R.F., 1996a. Examination timetabling in British universities – A survey. In: Burke, E.K., Ross, P. (Eds.), The Practice and Theory of Automated Timetabling I: Selected Papers from First International Conference on the Practice and Theory of Automated Timetabling, Edinburgh, UK, Lecture Notes in Computer Science, vol. 1153. Springer, pp. 76–92.

Burke, E.K., Newall, J., Weare, R.F., 1996b. A Memetic algorithm for university exam timetabling. In: Burke, E.K., Ross, P. (Eds.), Selected Papers from the First International Conference on Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153. Springer-Verlag, pp. 241–250.

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.

Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176, 177–192.

Caramia, M., Dell'Olmo, P., Italiano, G.F., 2001. New algorithms for examination timetabling. In: Naher, S., Wagner, D. (Eds.), Algorithm Engineering Fourth International Workshop, Proceedings WAE 2000, Lecture Notes in Computer Science, vol. 1982. Springer-Verlag, pp. 230–241.

Carter, M.W., Laporte, G., 1996a. Recent developments in practical examination timetabling. In: Burke, E.K., Ross, P. (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science, vol. 1153. Springer-Verlag, pp. 3–21.

Carter, M.W., Laporte, G., Lee, S.Y., 1996b. Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society 47 (3), 373–383.

Cote, P., Wong, T., Sabouri, R., 2005. Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. In: Burke, E.K., Trick, M. (Eds.), Selected Papers from the Fifth International Conference on Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science, vol. 3616. pp. 151–168.

Dammak, A., Elloumi, A., Kamoun, H., 2006. Classroom assignment for exam timetabling. Advances in Engineering Software 37 (10), 659–666.

de Werra, D., 1985. An introduction to timetabling. European Journal of Operational Research 19 (2), 151–162.

Di Gaspero, L., Schaerf, A., 2001. Tabu search techniques for examination timetabling. In: Burke, E.K., Erben, W. (Eds.), Selected Papers from the Third International Conference on Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 2079. Springer-Verlag, pp. 104–117.

Gogos, C., Alefragis, P., Housos, E., 2008. A multi-staged algorithmic process for the solution of the examination timetabling problem. Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19–22 August 2008.

Kendall, G., Hussin M.N., 2004. Tabu search hyper-heuristic approach to the examination timetabling problem at university technology mara. In: Burke, Edmund K., Trick, M., (Eds.), Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling (PATAT), Pittsburgh, USA, 18–20 August 2004, pp. 199–217.

McCollum, B., 2007. A perspective on bridging the gap between theory and practice in university timetabling. In: Burke, E.K., Rudova, H. (Eds.), Selected Papers from the Sixth International Conference on Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science, vol. 3867, pp 3–23.

McCollum, B., McMullan, P., Burke, E.K., Parkes, A.J, Qu, R., 2008. The second international timetabling competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/Exam/v1.0/1. Queen's Belfast University, N. Ireland.

McCollum, B., McMullan, P.J., Parkes, A.J., Burke, E.K., Abdullah, S., 2009. An extended great deluge approach to the examination timetabling problem. In: The Fourth Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin, August 2009, pp. 424–434.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A.J., Di Gaspero, L., Qu, R., Burke, E.K., 2010. Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS Journal on Computing 22 (1), 120–130.

Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J., 2003. A hybrid algorithm for the examination timetabling problem. In: Burke, E.K., De Causmaecker, P. (Eds.), Selected Papers from Fourth International Conference on Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science, vol. 2740. Springer-Verlag, pp. 207–231.

Pillay, N., Banzhaf, W., 2009. A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. European Journal of Operational Research 197 (2), 482–491.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y., 2009. A survey of search methodologies and automated system development for examination timetabling. Journal of Scheduling 12 (1), 55–89.

van den Broek, J., Hurkens, C., Woeginger, G., 2009. Timetabling problems at the TU Eindhoven. European Journal of Operational Research 196 (3), 877–885.