Appl Intell (2013) 39:438–450
DOI 10.1007/s10489-013-0422-z

Adaptive selection of heuristics for assigning time slots and rooms in exam timetables

Amr Soghier · Rong Qu

Published online: 27 February 2013

Abstract This paper presents an iterative adaptive approach which hybridises bin packing heuristics to assign exams to time slots and rooms. The approach combines a graph-colouring heuristic, to select an exam in every iteration, with bin-packing heuristics to automate the process of time slot and room allocation for exam timetabling problems. We start by analysing the quality of the solutions obtained by using one heuristic at a time. Depending on the individual performance of each heuristic, a random iterative hyper-heuristic is used to randomly hybridise the heuristics and produce a collection of heuristic sequences to construct solutions with different quality. Based on these sequences, we analyse the way in which the bin packing heuristics are automatically hybridised. It is observed that the performance of the heuristics used varies depending on the problem. Based on these observations, an iterative hybrid approach is developed to adaptively choose and hybridise the heuristics during solution construction. The overall aim here is to automate the heuristic design process, which draws upon an emerging research theme which is concerned with developing methods to design and adapt heuristics automatically. The approach is tested on the exam timetabling track of the second International Timetabling Competition, to evaluate its ability to generalise on instances with different features. The hyper-heuristic with low-level graph-colouring and bin-packing heuristics approach was found to generalise well over all the problem instances and performed comparably to the state of the art approaches.

A. Soghier (✉) · R. Qu
School of Computer Science, University of Nottingham,
Nottingham NG8 1BB, UK
e-mail: a_soghier@hotmail.com

R. Qu
e-mail: rxq@cs.nott.ac.uk

Keywords Heuristics · Exam timetabling · Bin packing · Graph colouring

1 Introduction

For more than 40 years exam timetabling has become one of the most studied domains in the AI and OR communities. This is due to its importance in many academic institutions worldwide. The basic problem is to allocate a time slot and a room for all the exams within a limited number of permitted time slots and rooms in order to find a feasible timetable. This assignment process is subject to 'hard' constraints which must be satisfied in order to get a *good* timetable. An example of such constraint is that no student is required to attend two exams at the same time. On the other hand, other constraints can be violated but must be satisfied as much as possible to obtain a good timetable. These are called 'soft' constraints.

As this task is time consuming and tedious to carry out manually, much effort during the last few decades has been directed to generate timetables automatically. With a large number of events needing to be assigned to resources (time slots and rooms) and a list of constraints (both hard and soft) to be addressed, there are a large number of potential solutions to this problem. Therefore, much of the research has been aimed at developing methodologies that focus on producing the best quality timetables for a specific problem or even a specific problem instance. A more recent direction in this field, namely, hyper-heuristics, aims to raise the level of generality of search methodologies to create algorithms that act well over a range of problems. A hyper-heuristic is seen as a heuristic to choose heuristics. In this paper, we present an adaptive approach which hybridises heuristics used in bin packing to assign time slots and rooms to exams during the construction of a timetable. It is based upon the observations and statistical analysis over a large number of different heuristic sequences obtained by a random iterative hyper-heuristic generator. This approach was tested on the second International Timetabling Competition (ITC2007) exam timetabling instances. It was found to generalise well over the instances of the data set. Furthermore, very competitive results have been produced against other approaches in the literature.

The following section presents a brief description of the problem used in this investigation. Section 2.2 provides an overview on some approaches from the literature applied to the problem. Furthermore, Sect. 2.3 concentrates on hyper-heuristic approaches. A random iterative hyper-heuristic to hybridise bin packing heuristics is proposed in Sect. 4. An adaptive methodology to select low-level heuristics and the results obtained are presented in Sect. 5. Finally, the future extensions of this work are summarised in Sect. 6.

2 Exam timetabling

Exam timetabling [7, 20] is the process of assigning a number of exams into a limited number of time slots while taking into consideration a number of constraints that must be satisfied. Many academic institutions face problems in scheduling their exams every semester or term [2]. The greater the number of constraints in the problem, the more difficult the exam timetabling problem becomes. In addition, there are a great number of different constraints that make exam timetabling problems different from one institution to the other. It is vital that some constraints are fully satisfied. These are called hard constraints. For example, a student cannot attend two exams at the same time. Therefore, any two exams that are attended by the same student must not be scheduled at the same time. A timetable that satisfies all hard constraints is called a feasible timetable.

Other constraints may be violated but the higher the degree of satisfying such constraints the better the solution. These are called soft constraints. For example, providing students with longer gaps between their exams is very desirable. However, not providing the gaps will not affect the exam being held and the timetable followed. Violations of soft constraints are used to decide the quality of the timetable created.

Every institution will have its own set of constraints according to its preferences. An institution could decide that each student should have only one exam per day, which is therefore regarded as a hard constraint. Another institution would accept that students can have more than one exam per day. In this case, the constraint of having a single exam each day is no longer a hard constraint.

In practice, the quality of a feasible timetable is evaluated in a different way. In the majority of the cases, the measure of quality is calculated based upon a penalty function which represents the degree to which the soft constraints are violated. The following section provides a description for the ITC2007 data set used in this investigation, followed by the approaches obtaining the best results for the data set in Sect. 2.2.

2.1 The International Timetabling Competition (ITC2007) data set

The ITC2007 problem set has the following hard constraints:

– No student attends more than one exam at the same time.
– The capacity for each individual room should not be exceeded at a given period.
– Period lengths should not be violated.
– All period related hard constraints need to be satisfied e.g. Exam A after Exam B.
– All room related hard constraints need to be satisfied e.g. Exam A must use Room X.

The soft constraints violations are summarised as follows:

– *Two Exams in a Row* The number of occurrences where a student sits two exams in a row on the same day.
– *Two Exams in a Day* The number of occurrences where a student sits two exams on the same day. If the exams are back to back then this is considered as a Two Exams in a Row violation to avoid duplication.
– *Period Spread* The exams have to be spread a certain number of time slots apart.
– *Mixed Durations* The number of occurrences where exams of different durations are assigned to the same room.
– *Larger Exams Constraint* The number of occurrences where the largest exams are scheduled near the end of the examination session. The number of the largest exams and the distance from the end of the exam session are specified in the problem description.
– *Room Penalty* The number of times where certain rooms, which have an associated penalty, are used.
– *Period Penalty* The number of times where certain time slots, which have an associated penalty, are used.

A complete description of the problem and the evaluation function can be found in [10]. In addition, the characteristics which define the instances are summarised in Table 1.

2.2 Exam timetabling approaches for the ITC2007 data set

A three phased approach was developed by Muller [12] to solve the problems in the ITC2007 exam timetabling track.

Table 1 Characteristics of the ITC2007 data set

Instance	Conflict density	Exams	Students	Periods	Rooms	No. of hard constraints
Exam 1	5.05	607	7891	54	7	12
Exam 2	1.17	870	12743	40	49	14
Exam 3	2.62	934	16439	36	48	185
Exam 4	15.0	273	5045	21	1	40
Exam 5	0.87	1018	9253	42	3	27
Exam 6	6.16	242	7909	16	8	23
Exam 7	1.93	1096	14676	80	15	28
Exam 8	4.55	598	7718	80	8	21

The first phase consists of an Iterative Forward Search algorithm to find a feasible solution. Conflict based statistics were used to avoid cycling. In the second phase, hill climbing is used to further improve the solution. Finally, a Great Deluge algorithm is applied to further explore the search space and further improve the solution.

Gogos et al. [9] proposed a method which used a GRASP (Greedy Randomised Adaptive Search Procedure) approach. Solutions are constructed using five orderings of exams based on various criteria. Tournament selection is used to select exams until they are all scheduled. A backtracking strategy using a tabu list is employed as required. In the improvement phase, Simulated Annealing is applied. Finally, room allocations are arranged using integer programming in the third phase.

Atsuta et al. [1] implemented a constraint satisfaction problem solver incorporating tabu search and iterated local search. The solver differentiates between the constraints and their corresponding weights during computation to improve performance. De Smet [8] also incorporated tabu search in a solver called Drools, an Open-Source Business Rule Management System (http://www.jboss.org/drools/).

Pillay [14] introduced a developmental approach (DA) which mimics the processes of cell behaviour. The saturation degree heuristic is used to order the exams and schedule them sequentially in the available "cells" i.e. time slots. The slot which causes the least overall constraint violation is chosen when more than one time slot is available. The best fit heuristic is used for room allocation. If a conflict occurs before all the exams are scheduled, the timetable is rearranged to reduce the soft constraint violation. This is described as cell division. If the overall soft constraint violation is not improved without breaking the hard constraints, cell interaction occurs. The time slots are swapped in this process to remove hard constraint violations. The process continues until a feasible solution is achieved. Finally, the contents of cells having equal durations are swapped to improve the solution. This is called cell migration.

McCollum et al. [11] proposed a two phased approach. The first phase is a construction phase which uses an adaptive ordering heuristic to create a feasible solution. The second phase improves the solution using an extension of the Great Deluge Algorithm.

Finally, Pillay [14] presented an evolutionary algorithm based hyper-heuristic using three different chromosome representations. An initial population is created and iteratively improved by applying the processes of evaluation, selection and recreation.

The results obtained by these methods are presented in Sect. 5.1.

2.3 Hyper-heuristics in exam timetabling

There are two classes of hyper-heuristics. One class aims to generate heuristics from a set of components while the second class aims to intelligently choose heuristics from a set of predefined heuristics which have been previously developed. This can be seen as a framework to automate the process of predefined heuristic choice and hybridisation. It is this second class that is the focus of the work presented in this paper.

A hyper-heuristic can be seen as a method to choose low-level heuristics depending on the problems in hand. Furthermore, it could be used to adapt or tune heuristics and meta-heuristics. Hyper-heuristics in exam timetabling can be categorised, according to the low-level heuristics they use, into two types as follows:

1. Hyper-heuristics with constructive low-level heuristics.
2. Hyper-heuristics with improvement low-level heuristics.

Ozcan et al. [13] state that one of the hyper-heuristic frameworks is based on a single point search in which heuristic selection and move acceptance are performed. They also highlight that most of the existing move acceptance methods compare a new solution to the current one to decide whether to accept it or not. They use a late acceptance strategy which compares between the new candidate solution and a previous solution, which is generated a number of steps earlier. A set of hyper-heuristics utilising different heuristic selection methods combined with the late acceptance strategy were investigated on exam timetabling problems.

Another approach was developed in [5] where a Case-Based Reasoning system was implemented. The system is used as a heuristic selector for solving exam timetabling problems. The different problem states and their corresponding constructive heuristics are stored in the system. The system then compares the problem to be solved with the cases that are already stored. The solutions are then constructed by repeatedly using the constructive heuristics suggested by the system.

Pillay and Banzhaf [17] present a genetic programming (GP) hyper-heuristic approach which acts over a search

Table 2 Graph colouring heuristics used in previous research [19]

Graph heuristics	Ordering strategies that order the events in the problem
Largest Degree (LD)	Decreasingly by the number of conflicts the event has with the others (i.e. two events have common students thus are conflicted)
Largest Weighted Degree (LWD)	The same as Largest Degree but weighted by the number of students involved in the conflicted events
Largest Enrolment (LE)	Decreasingly by the number of enrollments in the event
Saturation Degree (SD)	Increasingly by the number of valid time slots left for the event in the partial timetable
Colour Degree (CD)	Decreasingly by the number of conflicts the event has with those already scheduled

Table 3 Neighbourhood operators used in previous research [6]

Neighbourhood operators	Move strategies that change the position of events in the problem
Move Exam (ME)	Reassigns an exam to the time slot causing the least penalty
Swap Exam (SE)	Tries swapping an exam with another exam leading to the least penalty timetable
Kempe Chain Move (KCM)	Similar to the SE heuristic but is more complex as it involves swapping a subset of conflicting exams in two distinct time slots
Swap time slot (ST)	Swaps all the exams in a time slot with another set of exams in a different time slot

space of functions to assess the difficulty of allocating an exam during the timetable construction process. Each function is a heuristic combination of low-level construction heuristics combined by logical operators. The approach was applied to the Toronto benchmark on five instances of varying difficulty. The GP hyper-heuristic approach was found to generalise well over the five problems and performed comparably to other hyper-heuristic approaches, combining low-level construction heuristics. In addition, a study to investigate the use of genetic algorithms (GAs) as a means of inducing solutions was conducted in [18]. They used a two-phased approach to the problem which focuses on constructing timetables during the first phase, while improvements are made to these timetables in the second phase to reduce the soft constraint violations. Domain specific knowledge in the form of heuristics was used to guide the evolutionary process.

Finally, Sabar et al. [21] implemented a graph colouring constructive hyper-heuristic utilising the hierarchical hybridisations of the largest degree, saturation degree, largest coloured degree and largest enrollment graph colouring heuristics. The heuristics are hybridised to produce four ordered lists. A difficulty index of the first exam in each list is calculated by considering its order in all lists to obtain a combined evaluation of its difficulty. The most difficult exam to be scheduled is scheduled first. A roulette wheel selection mechanism is included in the algorithm to probabilistically select an appropriate timeslot for the chosen exam.

3 The random iterative hyper-heuristic

In previous work on hyper-heuristics [3, 19], heuristic sequences consisting of graph heuristics presented in Table 2 were used to construct solutions for exam timetabling problems. In [6], neighbourhood operators presented in Table 3 were investigated to improve exam timetables. The Kempe-chain (KCM) and swap time slot (ST) moves were addressed by an adaptive approach and were used to reschedule exams causing soft constraint violations to improve exam timetables.

In the first stage of the approach, heuristic sequences are generated using a random iterative improvement hyper-heuristic generator and applied to the sequence of exams causing violations ordered by Saturation Degree (SD). Algorithm 1 presents the pseudo-code of this initialisation stage of the hyper-heuristic.

The first stage of improvement, using a heuristic sequence (of length e, which is the number of exams causing violations), is an iterative process where, at the ith iteration, the ith heuristic in the sequence is used to reschedule the ith exam. The exam is assigned to the time slot and room leading to the least cost in the timetable and ties are broken by randomly choosing a time slot and room.

Algorithm 1 The pseudocode of the initialisation stage of the adaptive hyper-heuristic with low-level improvement heuristics

```
create an ordered list of the exams which cause a penalty using SD
create a heuristic sequence $h_s = \{KCM, KCM, KCM, \ldots, KCM\}$ //$h_s$ has the same size has the list of exams
for $i = 0 \to e \times 10$, do //$e$: number of exams causing penalty
  for $n = 1 \to e$ do
    $h$ = randomly change $n$ heuristics in $h_s$ to ST
    construct a solution $c$ using $h$
  end for
end for
if solution $c$ is feasible then
  save $h$ and the penalty of its corresponding solution $c$
end if
```

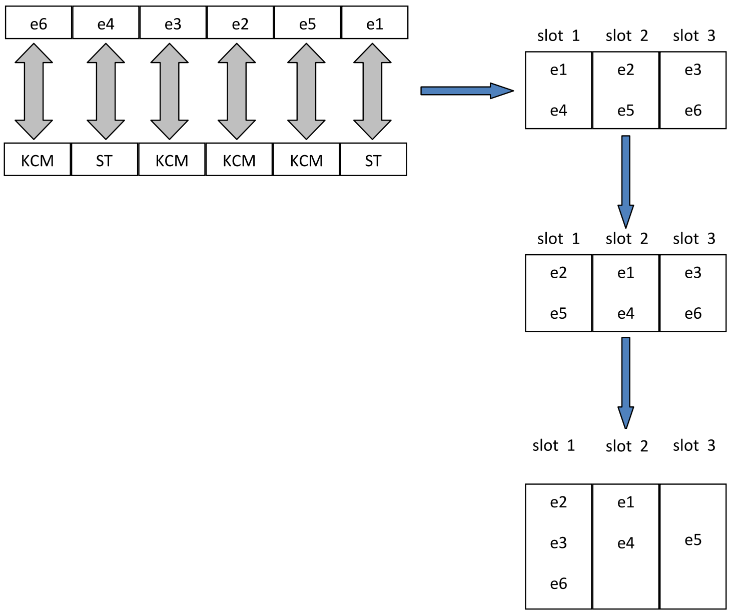


Fig. 1 An illustrative example of solution improvement using a sequence of neighbourhood operators

Figure 1 presents an illustrative example of the improvement process for a simple problem with six events (e1–e6) causing violations and ordered using SD. Assume that the sequence of six moves generated is "KCM ST KCM KCM KCM ST". An initial solution has been improved iteratively by using the first two heuristics in the sequence. A swap time slot move is applied to exam e1 to swap all the exams in slot 1 with the exams in slot 2. In the next iteration, assuming that exam 5 is in conflict with exam 6, a Kempe chain move is applied to exam 5 where it is moved from slot 1 to slot 3 and exam 6 is moved to slot 1. The rest of moves are applied to the corresponding exams in the sequence. In the case where an improvement is not achieved, the exam is kept in its current position. The sequences obtaining the best improvements are stored for the second stage where an adaptive approach is applied to generate more sequences which further improves the solution.

In the approach described, a set of heuristics (rather than solutions themselves) are considered for building solutions. This can be seen as a mechanism which adaptively calls the appropriate heuristics during the solution improvement process. Therefore, the hyper-heuristic can be seen as being able to, at a higher level, adaptively hybridise different heuristics. However, this hybridisation is considered to be more random as the search is only guided by the quality of the final solutions produced without focusing on the domain knowledge present within the problem being solved [19].

Bin packing heuristics on their own are simple techniques where items in the problem are packed using a specific strategy to construct solutions. For example, by using Best Fit (see Table 4), an item in a bin packing problem is placed into the bin which has the least space remaining. In exam timetabling problems, the exam is placed in a time slot and a room which has the least remaining capacity. The overall strategy is to make use of the same heuristics used in bin packing to assign resources in other problems such as exam timetabling.

Table 4 Packing strategies used to assign a time slot and room in timetabling

Packing heuristics	Packing strategies that allocate a time slot and room in the problem
Best Fit (BF)	Puts the exam in the feasible time slot and room which has the least remaining capacity
Almost Best Fit (ABF)	Puts the exam in the feasible time slot and room which has the second least remaining capacity
Worst Fit (WF)	Puts the exam in the feasible time slot and room which has the largest remaining capacity
Almost Worst Fit (AWF)	Puts the exam in the feasible time slot and room which has the second largest remaining capacity
First Fit (FF)	Puts the exam in the lowest indexed feasible time slot and room
Last Fit (LF)	Puts the exam in the highest indexed feasible time slot and room

Algorithm 2 The pseudo-code of the random iterative bin packing based hyper-heuristic

```
Create an ordered list $O$ of all the exams using LWD
for $i = 0 \to e \times 50$, do //$e$: the number of exams
  for $n = 1 \to e$ do
    initialise heuristic sequence $h_1 = \{$BF BF... BF BF$\}$
    initialise heuristic sequence $h_2 = \{$ABF ABF... ABF ABF$\}$
    initialise heuristic sequence $h_3 = \{$WF WF... WF WF$\}$
    initialise heuristic sequence $h_4 = \{$AWF AWF... AWF AWF$\}$
    initialise heuristic sequence $h_5 = \{$FF FF... FF FF$\}$
    initialise heuristic sequence $h_6 = \{$LF LF... LF LF$\}$
    for $s = 1 \to 6$ do
      $h_s$ = randomly change $n$ heuristics in $h_s$ to BF, ABF, WF, AWF, FF or LF
      construct a solution $c$ by applying the heuristic sequence $h_s$ to the exams in list $O$ (see Fig. 1)
      if solution $c$ is feasible then
        save $h_s$ and the penalty of its corresponding solution $c$
      end if
    end for
  end for
end for
```

In this paper, we present an adaptive approach which hybridises heuristics used in bin packing to assign time slots and rooms to exams during the construction of a timetable. It is based upon the observations and statistical analysis over a large number of different heuristic sequences obtained by a random iterative hyper-heuristic generator. A similar process to the one described above will be used in the approach developed in this paper. However, more than two heuristics are used in the hybridisation and a different method to adapt the heuristic sequences is presented.

4 Hybridisations of bin packing heuristics within a hyper-heuristic

4.1 A random iterative time slot and room assignment hyper-heuristic

The approach presented in this paper focuses on the assignment of exams to time slots and rooms. It takes a similar approach to the one presented in [6] and [19] where a random iterative hyper-heuristic generates heuristic sequences for the benchmark problem mentioned in Sect. 2.1. Instead of using the heuristic sequences to order the exams to construct solutions or to improve constructed solutions, they are used here to assign exams to time slots and rooms.

Algorithm 2 presents the pseudo-code of this random iterative hyper-heuristic. The process starts by ordering exams using the LWD heuristic which orders the exams in a descending order according to the number of conflicts each exam has with others. The exams are weighted according to the number of students involved in the conflict (see Table 2). Although it was proven in previous research [4, 7] that using SD performs the best in most cases due to its ability to dynamically order the events according to the number of remaining valid time slots, we have chosen to fix the order of the exams by using a static ordering heuristic to be able to focus on the effect of the heuristics assigning time slots and rooms.

Table 5 Best and average results obtained by using a single heuristic. A (–) indicates that a feasible solution could not be obtained

Instances	BF best	BF average	ABF best	ABF average	WF best	WF average	AWF best	AWF average	FF best	LF best
Exam 1	12938	13690	13139	13712	–	–	–	–	–	**12577**
Exam 2	**3598**	4636	4403	5278	4043	5194	–	–	4158	3709
Exam 3	**17586**	19690	20143	21558	20261	21040	–	–	20083	20308
Exam 4	**25165**	28776	44358	44358	–	–	–	–	–	–
Exam 5	4909	5829	4826	5726	5553	5673	–	–	**4778**	4844
Exam 6	32175	32175	32575	32575	–	–	–	–	**31480**	38740
Exam 7	22388	23483	25446	26982	24203	24874	–	–	**21314**	23331
Exam 8	15972	16231	16046	16338	17766	17766	–	–	**15740**	18141

At every iteration, the exam at the top of the list is chosen and the corresponding heuristic from the generated sequence is used to assign a time slot and room to the exam. The heuristics used for the assignment are those used in the one dimensional bin packing domain as described in Table 4. In cases where ties appear when using BF, ABF, WF or ABF, the time slot and room leading to the least penalty is chosen. If more than one time slot and room combination lead to the least penalty, one of these time slot and room combinations are randomly chosen. However, this does not apply when using FF or LF since these heuristics aim to assign exams in the first or last feasible time slot and room found. The process stops and the sequence is discarded if a feasible solution could not be generated. After a certain number of steps, a set of heuristics is collected for further analysis on the characteristics of good hybridisations of packing heuristics.

In this work, to investigate the effect of hybridising different heuristics to allocate time slots and rooms to exams, we generate a large number of heuristic sequences which consist of the six bin packing heuristics described in Table 4. Since the allocation of exams depends on the hard and soft constraints of the time slots and rooms in the problem, it is essential that we examine a number of different strategies in assigning the exams. Since one of the objectives of hyper-heuristics is to avoid using any domain specific knowledge, we are only interested in the quality of the solutions obtained from the different allocation strategies to guide our search.

We start by applying the six initial heuristic sequences to the problem and analyse the effect of these sequences without performing a hybridisation in Sect. 4.2. In Sect. 4.3, we use the random iterative hyper-heuristic described above to generate a large number of hybridised heuristic sequences and analyse the results after applying the sequences to the problem instances. Finally, we develop an adaptive approach based on our observations and analysis in Sect. 5.

4.2 Analysis of the initial heuristic sequences

We start by applying the initial heuristic sequences, which contain only a single heuristic, to eight instances of the International Timetabling Competition (ITC2007) exam timetabling data set presented in Sect. 2.1. The aim is to be able to compare the results before and after hybridising heuristics in the sequence. Since the timetable is empty, many ties appear at the beginning of the process. A random choice is made from the set of feasible time slots and rooms available. Therefore, the initial sequences were run for 10 distinct seeds for each instance and the average and best results were collected. Table 5 presents the results obtained by running the six heuristic sequences which consist of a single heuristic.

It is clear, from Table 5, that none of the heuristics used performed the best for all the exam timetabling instances. However, varying solution quality was obtained when applying the heuristics to the instances. This is due to the fact that the instances have different structures and constraints. This indicates that different allocation strategies better suited the different instances. In addition, the results show that a feasible solution could not be obtained for any of the instances when using AWF.

Table 6 presents the capacities of the rooms in each instance ordered in the way they appear in the problem. For the first instance, feasible solutions were generated using BF, ABF and LF only. One possible reason may be that the first room in each time slot is the largest and using FF, WF or AWF would start by filling this room. This leaves exams with a large enrollment difficult to schedule later on in the process. The best solution was obtained using LF.

It can be seen that BF performed the best for the second and third instances. The second best solution was achieved using LF and FF for the second and third instances respectively. This may be due to the fact that the performance of the heuristics vary depending on the position of the room with the largest capacity in each instance. In addition, delaying the use of the largest room leads to better solutions. For example, in the second instance, the first room in each time slot has the largest capacity. Therefore, LF performed better than FF for this instance. On the contrary, FF performed better than LF on the third instance since the rooms with the

Table 6 Room capacity ordered as they appear in each instance

Instance	Room capacity ordered as they appear in the problem
Exam 1	**260**, 100, 129, 60, 77, 65, 111
Exam 2	**424**, 219, 120, 100, 40, 60, 60, 40, 36, 30, 30, 25, 72, 40, 35, 40, 38, 30, 60, 30, 85, 110, 100, 80, 70, 80, 40, 50, 92, 58, 195, 400, 90, 110, 264, 50, 19, 27, 60, 40, 51, 30, 39, 50, 50, 14, 127, 143, 23
Exam 3	127, 77, 41, 101, 93, 93, 76, 30, 70, 545, 275, 24, 171, 44, 70, 78, 59, 49, 27, 20, 39, 182, 32, 65, 58, 56, 61, 28, 26, 44, 78, 120, 500, 18, 30, 100, 11, **800**, 16, 40, 16, 14, 116, 42, 51, 39, 500, 60
Exam 4	**1200**
Exam 5	896, 500, **999**
Exam 6	240, 90, 210, 210, 110, 110, 80, **1000**
Exam 7	240, 90, 210, 210, 110, 110, 70, 75, 70, 110, 110, 35, 45, 45, **1000**
Exam 8	**260**, 100, 129, 60, 77, 65, 111, 120

Table 7 Best and average results obtained by using FF and LF after reordering the room capacities in ascending order. A (–) indicates that a feasible solution could not be obtained

Instances	FF before re-ordering	FF after re-ordering	LF before re-ordering	LF after re-ordering
Exam 1	–	12694	12577	–
Exam 2	4158	3923	3709	–
Exam 3	20083	20944	20308	–
Exam 4	–	–	–	–
Exam 5	4778	4965	4544	5281
Exam 6	31480	30679	38740	–
Exam 7	21314	18571	23331	24186
Exam 8	15740	15938	18141	

largest capacity are at the end of each time slot. The same observation also applies to the performance of FF and LF in the rest of the instances where a feasible solution was obtained using the two heuristics. Furthermore, since the distribution of room capacity is uniform in each time slot in the third instance, a small difference can be seen in the results obtained using FF and LF. To verify this observation, the sequences were applied to the instances after reordering the capacities of the rooms in ascending order. Table 7 presents the results of LF and FF before and after reordering the capacities of the rooms. Since the fourth instance contains only a single room in each time slot, only BF and ABF could find a feasible solution to this instance. Note that although the LWD heuristic is used to order the exams, this does not mean that the exams with the largest number of students which require large rooms are scheduled first since the degree of conflicts with other exams is also considered.

4.3 Analysis of the hybrid heuristic sequences

The random iterative time slot and room assignment hyper-heuristic generates a collection of heuristic sequences by hybridising different rates of BF, ABF, WF, AWF, FF or LF as described in Algorithm 2. The idea is to generate a variety of sequences which contain heuristics in different positions to use different strategies to schedule exams. Since the order of exams does not change during the solution construction, the effect of the heuristic used to schedule each exam can be clearly observed.

Due to the amount of heuristic sequences that could be generated from using a large number of different heuristics, 5 heuristic sequences are initialised each using BF, ABF, WF, FF and LF. AWF was discarded from the hybridisation since it could not obtain any feasible result in Table 5. The random iterative hyper-heuristic then systematically hybridises n BF, ABF, WF, FF or LF, $n = [1, \dots, e]$, in each sequence. For each unit of hybridisation for each sequence, 50 samples are obtained. Although it is very difficult to cover the whole range of heuristic sequences, our focus is to explore the different areas of the search space by generating sequences which contain different hybridisations. Table 8 presents the best and worst solutions obtained when applying the sequences to the problem instances. To analyse the performance of each heuristic in the sequences leading to the best and worst solutions for each instance, the corresponding amounts of BF, ABF, WF, FF and LF in the sequences are also presented.

It can be seen, from Table 8, that the hybridisations leading to the best solutions consists mainly of the heuristic which produced the best solution using the initial sequences

Table 8 Penalties of the best and worst solutions from the heuristic sequences and the amount of hybridisation of BF, ABF, WF, FF and LF

Instances	Best	% of hybridisation					Worst	% of hybridisation				
		BF	ABF	WF	FF	LF		BF	ABF	WF	FF	LF
Exam 1	7821	26	17	0	0	57	14358	19	38	31	9	3
Exam 2	4295	34	6	18	16	26	6827	8	43	29	18	2
Exam 3	15386	32	19	9	28	12	22149	6	28	11	7	48
Exam 4	23713	63	30	0	0	0	50288	12	63	18	3	4
Exam 5	4288	14	20	6	34	26	6239	39	8	43	4	6
Exam 6	29349	28	17	0	42	13	40340	5	19	37	13	26
Exam 7	9752	25	8	13	35	19	27417	13	31	24	14	18
Exam 8	15675	27	21	11	39	2	18599	3	18	22	5	52

as presented in Table 5. Furthermore, the overall hybridisation of each heuristic in the best heuristic sequences for different instances depends on the performance of the heuristics before the hybridisation. For example for the first instance, from Table 5, LF performed the best followed by BF then ABF. Therefore, the best heuristic sequence for the first instance employs more LF followed by BF then ABF.

Another observation from Tables 5 and 8 is that the worst solutions were obtained when hybridising heuristics which performed poorly when used on their own.

The initial heuristics help the hyper-heuristic to gather knowledge about the nature of the problem. Using this information, the search can be guided and the hybridisation process can be adapted and automated to improve the quality of the solutions.

5 Adaptive hybridisation of bin packing heuristics

The above observations indicate that although the heuristics to be hybridised can be identified from running the initial sequences, the level of hybridisation of each heuristic and their appropriate ranges vary a lot for different instances. In addition, running the initial sequences gives an indication of the nature of the problem from the quality of the solution produced. Therefore, an intelligent approach needs to be developed to adaptively choose the heuristics to be used and the amount of each heuristic in the sequence. The adaptive approach is applied to all the instances and shown to be effective in comparison to the best approaches in the literature.

The adaptive approach is a three staged approach which identifies the heuristics to use in the hybridisation. A sequence is then initialised with a certain amount of each heuristic. Finally, an iterative adjustment is made to the amount of each heuristic in the sequence. The process is presented as follows:

1. In the first stage, the five initial heuristics containing only BF, ABF, WF, FF or LF are applied to the problem and the results are collected. The heuristics are ranked in a descending order according to the quality of the solutions produced. Heuristics which cannot produce a solution are discarded. A sequence is initialised using the heuristic producing the best result in this stage.
2. Based on the results obtained from the first stage, an adjustment is made to hybridise the initial sequence with the rest of the heuristics. The level of hybridisation of each heuristic is based on the number of heuristics, producing a feasible solution and the ranking of heuristics from stage 1. According to the ranking of heuristics, the level of hybridisation of each heuristic is defined. The level of hybridisation is proportional to the ranking. For example, if two heuristics obtain a feasible solution for a problem which contains 100 exams, 67 of the exams will use the 1st heuristic in the ranking while 33 will use the 2nd. The following equation is used to determine the level of hybridisation of each heuristic:

$$\frac{e}{\frac{f^2+f}{2}} \times r \tag{1}$$

e: the number of exams in the problem; f: the number of heuristics obtaining a feasible solution.

The ranking r is calculated using the following equation:

$$r = f - i + 1 \tag{2}$$

i: the ranking of the heuristic according to the quality of the solution produced. The heuristic producing the best solution has the highest ranking.

The ranking evaluates the quality of a heuristic on a scale from $1 - f$. For example if four heuristics obtain feasible solutions then the top heuristic will have $r = 1$ and the worst will have $r = 4$. The equation calculates the amount of occurrence of each heuristic in a sequence. Therefore, for a problem which contains fourty exams and four heuristics the amount of each heuristic will be calculated as follows $r_1 = 3$, $r_2 = 6$, $r_3 = 9$ and $r_4 = 12$.

Algorithm 3 The pseudo-code of the initialisation stages of the adaptive bin packing based hyper-heuristic

```
create an ordered list $O$ of all the exams using LWD
initialise heuristic sequence $s_1 = \{BF BF ... BF BF\}$
initialise heuristic sequence $s_2 = \{ABF ABF ... ABF ABF\}$
initialise heuristic sequence $s_3 = \{WF WF ... WF WF\}$
initialise heuristic sequence $s_4 = \{FF FF ... FF FF\}$
initialise heuristic sequence $s_5 = \{LF LF ... LF LF\}$
$x$ = id of the heuristic sequence obtaining the best solution
$f$ = number of sequences from $s_1$ to $s_5$ producing a feasible solution
$s_h = s_x$
for $i = 1 \to f$ do
    $n = (e/((f^2 + f)/2)) * (f - i + 1)$ //Equation (1)
    $s_h$ = randomly change $n$ heuristics in $s_h$ to the heuristic used in sequence $s_i$
end for
construct a solution $c$ by applying the heuristic sequence $s_h$ to the exams in list $O$ (see Fig. 1)
if solution $c$ is feasible then
    save $s_h$ and the penalty of its corresponding solution $c$
end if
```

Algorithm 4 The pseudo-code of the adaptive tuning of the levels of hybridisations in a heuristic sequence

```
create an ordered list $O$ of all the exams using LWD
$f$ = number of sequences producing a feasible solution
for $i = 1 \to f$ do
    $h_i$ = the heuristic with rank $i$
    while an improvement is achieved do
        $s_h$ = randomly change 1 % of the heuristics in the sequence to heuristic $h_i$
        construct a solution $c$ by applying the heuristic sequence $s_h$ to the exams in list $O$ (see Fig. 1)
        if solution $c$ is better than the previous solution then
            save $s_h$ and the penalty of its corresponding solution $c$
        end if
    end while
end for
```

Algorithm 3 presents the pseudo-code of the first two stages of the approach. In the first stage, the exams are ordered using LWD and a single heuristic is used to assign all the exams to rooms and time slots. The five heuristics are applied, one heuristic at a time, to see whether a feasible solution could be obtained. The number of heuristics obtaining a feasible solution is used to calculate the amount of hybridisation required in a sequence, using each heuristic. Random sequences are then generated using the calculated amount of hybridisations to allow different heuristics to be applied to different exams. The hybridised sequences obtaining feasible solutions are saved for the next stage of the approach.

3. Based on the heuristic sequence obtained from the second stage, an iterative adjustment is made to tune the levels of hybridisations of each heuristic over the whole heuristic sequence. The aim in this stage is to increase the levels of hybridisation of the best performing heuristics.

Algorithm 4 presents the pseudo-code of the approach used to tune the levels of hybridisation in the sequence obtained from stage 2. The heuristics in the sequence are tuned starting with the best heuristic in the ranking. The level of hybridisation in the sequence is increased by 1 % by randomly changing other heuristics. Note that if the same heuristic is used in a randomly chosen position, another position is chosen to guarantee that the level of hybridisation is increased. Depending on the size of the problem, a number of sequences are generated and applied to the problem. The level of hybridisation is only increased if a better solution is obtained. When an improvement is not obtained, the level of hybridisation of the next heuristic in the ranking is increased. The process stops when all the heuristics used in the sequence are increased and an improvement is not achieved.

Table 9 Best results obtained by the Room and Time slot Assignment Hyper-heuristic (RTAH) compared to the best approaches in the literature on the ITC2007 data set. The ranking of the approaches is presented between brackets

Instances	Exam 1	Exam 2	Exam 3	Exam 4	Exam 5	Exam 6	Exam 7	Exam 8	Average result
RTAH best	5752 (3)	1693 (7)	14586 (6)	21491 (5)	3844 (3)	28480 (7)	5182 (3)	13711 (6)	11842.38 (5)
McCollum (2009) [11]	4633 (2)	405 (3)	9064 (1)	15663 (1)	3042 (2)	25880 (1)	4037 (1)	7461 (1)	8773.13 (1)
Muller (2008) [12]	4370 (1)	400 (2)	10049 (2)	18141 (3)	2988 (1)	26950 (2)	4213 (2)	7861 (2)	9371.50 (2)
Gogos (2008) [9]	5905 (4)	1008 (6)	13862 (5)	18674 (4)	4139 (7)	27640 (4)	6683 (6)	10521 (4)	11054 (4)
Atsuta (2008) [9]	8006 (7)	3470 (9)	18622 (8)	22559 (7)	4714 (8)	29155 (8)	10473 (8)	14317 (7)	13914.5 (7)
De Smet (2008) [8]	6670 (6)	623 (4)	– (9)	– (9)	3847 (4)	27815 (5)	5420 (4)	– (9)	– (9)
Pillay (2008) [14]	12035 (9)	3074 (8)	15917 (7)	23852 (8)	6860 (9)	32250 (9)	17666 (9)	16184 (8)	15946 (8)
Pillay (2010) [16]	8559 (8)	830 (5)	11576 (3)	21901 (6)	3969 (6)	28340 (6)	8167 (7)	12658 (5)	12000 (6)
Sabar (2012) [21]	6234 (5)	395 (1)	13002 (4)	17940 (2)	3900 (5)	27000 (3)	6214 (5)	8552 (3)	10404.63 (3)

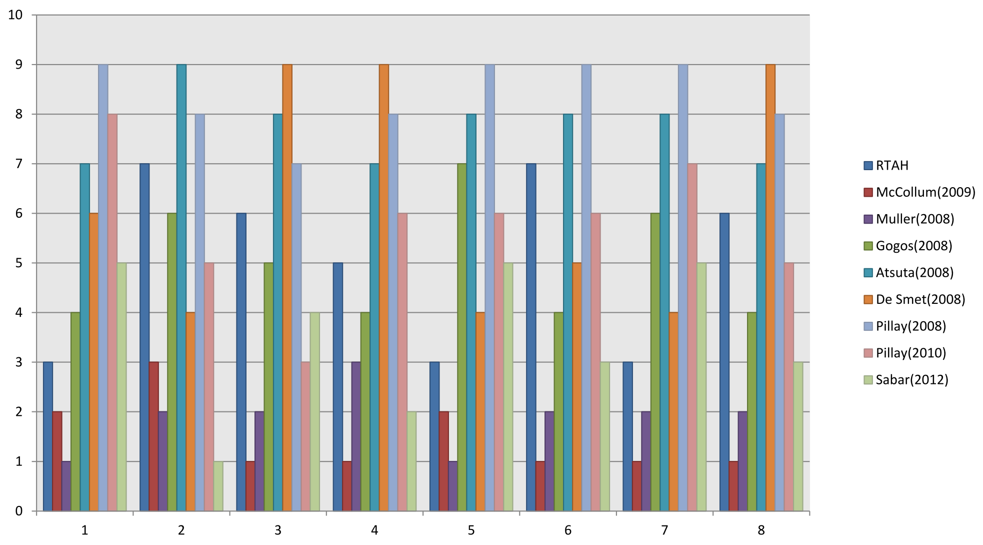


Fig. 2 Ranking of approaches for each data set

5.1 The International Timetabling Competition (ITC2007) results

We evaluate our adaptive approach by applying it to the ITC2007 instances described in Table 1. The approach is run for the same amount of time specified in the timetabling competition, on a Pentium IV machine with a 1 GB memory, to allow a fair comparison with the reported results. The results are presented in Table 9.

Table 9 shows that the results produced by the adaptive approach are better than the random approach presented in Table 8. In addition, Table 9 presents the results we obtained in comparison with the best in the literature. The description of the approaches used for comparison is presented in Sect. 2.2. Furthermore, the ranking of each approach and the average results are presented. The rankings are also illustrated in Fig. 2. We do emphasise that the objective here is not to beat the best reported results but to demonstrate the generality of our approach to the different problem instances and the ability of a hyper-heuristic to adaptively hybridise heuristics used in bin-packing to assign events (exams in our case) to time slots and rooms.

The Extended Great Deluge in [11] obtained the best results for five out of the eight instances. However, the approach was run for a longer time as it was developed after the competition. In the competition, the best results for all the eight instances were reported in [12] using a three phased approach. The GRASP used in [9] produced the second best results.

In comparison to the evolutionary algorithm based hyper-heuristic presented in [15], our approach was able to produce better results in only one instance. However, it was stated that they did not adhere to the time limitation imposed by the competition.

In comparison to the Constraint Based Solver developed in [1], our approach performed better in all eight instances. The approach using the Drools solver in [8] obtained feasibility for only five instances. Our approach out-performed it, gaining feasibility for all eight instances. This demonstrates the generality of our approach to solving exam timetabling problems. Furthermore, our approach performed better on all instances in comparison with the biologically inspired approach proposed in [14]. Finally, our approach performed better on three instances in comparison with the graph colouring hyper-heuristic recently presented in [21].

6 Conclusions

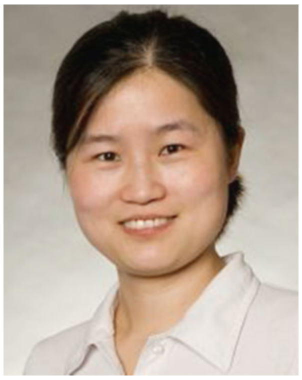
The study presented in this paper implements a hyper-heuristic approach which adaptively hybridises bin packing heuristics to assign exams to time slots and rooms. An investigation is made on the low-level heuristics used and their performance on different problem instances. A random iterative hyper-heuristic is developed to hybridise the heuristics and generate a large number of sequences of differing quality. It is shown that the hybridisations leading to the best solutions consist mainly of the heuristic which produced the best solution when used on its own. Furthermore, the overall hybridisation of each heuristic in the best heuristic sequences for different instances depends on the performance of the heuristics before the hybridisation. Based on these observations, an adaptive approach which chooses the heuristics to use and tunes the level of hybridisation of each heuristic is implemented. Furthermore, the hyper-heuristic approach is applied to the International Timetabling Competition (ITC2007) data set and showed to produce very competitive results compared to other approaches in the literature.

Future research directions include performing improvements during the timetable construction stage. This would allow changing the time slot or room assignment of a scheduled exam to make room for another exam. In addition, a dynamic heuristic can be used such as SD instead of LWD to order the exams. The objective is to combine heuristics used to order exams with heuristics used to assign time slots and rooms. Finally, the approach investigated in this paper can be applied to course timetabling or any other resource allocation problem.

References

1. Atsuta M, Nonobe K, Ibaraki T (2008) Itc2007 track 1: an approach using general csp solver. In: Practice and theory of automated timetabling (PATAT 2008), August 2008, pp 19–22
2. Burke EK, Elliman DG, Ford PH, Weare RF (1996) Examination timetabling in British universities. In: Burke EK, Ross R (eds) The practice and theory of automated timetabling. Lecture notes in computer science, vol 1153. Springer, Berlin, pp 76–92
3. Burke EK, McCollum B, Meisels A, Petrovic S, Qu R (2007) A graph-based hyper-heuristic for educational timetabling problems. Eur J Oper Res 176:177–192
4. Burke EK, Newall J (2004) Solving examination timetabling problems through adaptation of heuristic orderings. Ann Oper Res 129(2):107–134
5. Burke EK, Petrovic S, Qu R (2006) Case based heuristic selection for timetabling problems. J Sched 9(2):115–132
6. Burke EK, Qu R, Soghier A (2010) Adaptive selection of heuristics for improving constructed exam timetables. In: Proceedings of the 8th international conference on the practice and theory of automated timetabling (PATAT'10), pp 136–151
7. Carter MW, Laporte G, Lee SY (1996) Examination timetabling: algorithmic strategies and applications. J Oper Res Soc 74:373–383
8. De Smet G (2008) Examination track, practice and theory of automated timetabling. In: Examination track, practice and theory of automated timetabling (PATAT'08), Montreal, 19–22 August 2008
9. Gogos C, Alefragis P, Housos E (2008) A multi-staged algorithmic process for the solution of the examination timetabling problem. In: Practice and theory of automated timetabling (PATAT 2008), pp 19–22
10. McCollum B, McMullan P, Burke EK, Parkes AJ, Qu R (2012) A new model for automated examination timetabling. Ann Oper Res 194(1):291–315
11. McCollum B, McMullan P, Parkes AJ, Burke EK, Abdullah S (2009) An extended great deluge approach to the examination timetabling problem. In: Proceedings of the 4th multidisciplinary international scheduling: theory and applications 2009 (MISTA 2009), Dublin, Ireland, 10–12 August 2009, pp 424–434
12. Muller T (2008) Itc 2007 solver description: a hybrid approach. In: Practice and theory of automated timetabling (PATAT 2008), August 2008, pp 19–22
13. Ozcan E, Bykov Y, Birben M, Burke EK (2009) Examination timetabling using late acceptance hyper-heuristics. In: Evolutionary computation 2009 (CEC'09), pp 997–1004
14. Pillay N (2008) A developmental approach to the examination timetabling problem. In: Practice and theory of automated timetabling (PATAT 2008), August 2008, pp 19–22
15. Pillay N (2010) Evolving hyper-heuristics for a highly constrained examination timetabling problem. In: Proceedings of the 8th international conference on the practice and theory of automated timetabling (PATAT'10), pp 336–346
16. Pillay N (2010) A study into the use of hyper-heuristics to solve the school timetabling problem. In: Proceedings of the 2010 annual. Research conference of the South African institute of computer scientists and information technologists (SAICSIT'10)
17. Pillay N, Banzhaf W (2009) A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. Eur J Oper Res 197:482–491
18. Pillay N, Banzhaf W (2010) An informed genetic algorithm for the examination timetabling problem. Appl Soft Comput 10(2):457–467
19. Qu R, Burke EK, McCollum B (2009) Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. Eur J Oper Res 198(2):392–404
20. Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY (2009) A survey of search methodologies and automated approaches for examination timetabling. J Sched 12(1):55–89
21. Sabar N, Ayob M, Qu R, Kendall G (2012) A graph colouring constructive hyper-heuristic for examination timetabling problems. Appl Intell 37:1–11

Amr Soghier is a senior software developer at Amadeus which is one of the world's leading suppliers of IT solutions in the travel and tourism industry. He received his Ph.D. and M.Sc. degrees in Computer Science from the University of Nottingham. His research interests are in Airport Operations Optimisation, timetabling problems and business intelligence using hyper-heuristics, meta-heuristics and constraint programming.



Rong Qu received a B.Sc. (Hons) degree in Computer Science from the XiDian University, Xi'an, China in 1996 and Ph.D. degree in Computer Science from the University of Nottingham, UK in 2002. She is currently a Lecturer in the School of Computer Science at the University of Nottingham, UK. She has authored and coauthored over 80 research papers in journals and conferences. She is a member of the Automated Scheduling, Optimization and Planning (ASAP) research group. Her current research interests are multicast network routing, computational finance, workforce scheduling and routing, personnel scheduling, timetabling problems by using hyperheuristics, meta-heuristics, constraint programming, integer programming and case based reasoning.