Adaptive Selection of Heuristics for Improving Constructed Exam Timetables

Edmund K. Burke, Rong Qu, and Amr Soghier

School of Computer Science, University of Nottingham
Nottingham, NG8 1BB, UK
{ekb,rxq,azs}@cs.nott.ac.uk

Abstract. This paper presents a hyper-heuristic approach which hybridises low-level heuristics to improve constructed timetables. The constructed timetable is analysed and the exams causing a soft-constraint violation are identified. It is observed that both the type of move performed and the order in which exams are rescheduled in the timetable affect the quality of the solution produced. After testing different combinations in a hybrid approach, the Kempe chain move heuristic and swapping timeslots proved to be the best heuristics to use in a hybridisation. Similarly, it was proved that ordering the exams using Saturation Degree and breaking any ties using Largest Weighted Degree produces the best results. Based on these observations, an iterative hybrid approach is developed to adaptively hybridise these two heuristics in two stages. In the first stage, random heuristic sequences are generated and applied to the problem. The heuristic sequences are automatically analysed. The heuristics repeated in the best sequences are fixed while the rest are randomly changed in an attempt to find the best heuristic sequence. The approach is tested on the Toronto benchmark and the exam timetabling track of the second International Timetabling Competition, to evaluate its ability to generalise. The hyper-heuristic with low-level improvement heuristics approach was found to generalise well over the two different datasets and performed comparably to the state of the art approaches.

1 Introduction

For more than 40 years exam timetabling has become one of the most studied domains in the AI and OR research communities. This is due to its importance in many academic institutions worldwide. However, much of the research has been aimed at developing methodologies that produce the best quality timetables for a single problem [24]. 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 [7]. In this case the low-level heuristics represent the search space. The low-level heuristics can be categorised as heuristics which construct a timetable or heuristics which perform certain moves to improve a constructed timetable. This paper presents a random iterative hyper-heuristic approach which uses improvement low-level heuristics. This approach was tested on the Toronto benchmark and the second International Timetabling Competition (ITC2007) exam timetabling problems. It was found to generalise well over

the two datasets. Furthermore, very competitive results were produced against other approaches in the literature.

The following section presents a brief description of the benchmarks. Section 2.3 and 2.4 provide an overview on different approaches developed in the exam timetabling domain. Furthermore, section 2.5 concentrates on hyper-heuristic approaches. A random iterative hyper-heuristic to improve timetables is proposed in section 3. An adaptive methodology to select low-level heuristics and the results obtained are presented in section 4. The future extensions of this work are summarised in section 5.

2 Exam Timetabling

2.1 The Toronto Benchmark

An exam timetabling problem consists of a set of exams to be allocated to a given set of timeslots. The generated timetable must satisfy the hard constraints of the problem. The hard constraints are the requirements that cannot be violated, e.g. no one student must be scheduled to sit two exams during the same period. A timetable which meets all the hard constraints given is called a feasible timetable. A timetabling problem can also have a set of soft constraints. Soft constraints are constraints that can be violated and determine the quality of the timetable generated, e.g. there must be a certain number of periods between two exams sat by the same student. Therefore, high quality timetables contain the least number of soft constraint violations. The Toronto benchmark problem is well known in the exam timetabling community since it was firstly introduced by Carter et al. [11] in 1996 . Over the years, a slightly different version was made available and used to test some approaches in the literature. The characteristics of the two versions of this dataset are presented in table 1. The problem has one hard constraint where conflicting exams cannot be assigned to the same time slot. In addition, a soft constraint is present where conflicting exams should be spread throughout the timetable as far as possible from each other. The goal here is to minimise the sum of proximity costs given as:

$$\sum_{i=0}^{4}(w_i \text{ x } n)/S$$

where

- $w_i = 2^{4-i}$ is the cost of assigning two exams with i time slots apart. Only exams with common students and are four or less time slots apart are considered as violations
- n is the number of students involved in the conflict
- S is the total number of students in the problem

Since then this problem has been used to test and compare many approaches in the literature. Recently, a more constrained set of benchmarks was made available as part of the International Timetabling Competition (ITC2007) [17]. The next section describes the ITC2007 dataset in detail.

Problem	Exams I/II/IIc	Students I/II	Enrolments I/II	Density	Time Slots
car91	682	16925	56877	0.13	35
car92	543	18419	55522	0.14	32
ear83 I	190	1125	8109	0.27	24
ear83 II	189	1108	8057	0.27	24
hec92 I	81	2823	10632	0.42	18
hec92 II	80	2823	10625	0.42	18
kfu93 I	461	5349	25113	0.06	20
lse91	381	2726	10918	0.06	18
sta83 I	139	611	5751	0.14	13
sta83 II	138	549	5417	0.19	35
tre92	261	4360	14901	0.18	23
uta92 I	622	21266	58979	0.13	35
uta92 II	638	21329	59144	0.12	35
ute92	184	2750	11793	0.08	10
yor83	181	941	6034	0.29	21
yor83 II	180	919	6002	0.3	21

Table 1. Characteristics of the two versions of the Toronto Benchmark datasets

2.2 The International Timetabling Competition (ITC2007) dataset

The ITC2007 exam timetabling track could be considered as a complex and a more practical dataset in comparison to the Toronto benchmark. This is due to the larger number of constraints it contains. A full description of the problem and the evaluation function can be found in [17]. In addition, the characteristics which define the instances are summarised in table 2. The problem consists of the following:

- A set of timeslots covering a specified length of time. The number of timeslots and their durations are provided.
- A set of exams which should be allocated to the timeslots.
- A list of the students enrolled in each exam.
- A set of rooms with different capacities.
- A set of additional hard constraints (e.g. exam X must be after exam Y or exam A must use Room R).
- A set of soft constraints and their associated penalties.

In comparison to the Toronto benchmark, the ITC2007 dataset has more than one hard constraint. The hard constraints are as follows:

- No student sits 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.
- Additional hard constraints should be all satisfied.

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 timeslots 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 timeslots, which have an associated penalty, are used.

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

Table 2. Characteristics of the ITC2007 dataset

2.3 Exam timetabling approaches for the ITC2007 dataset

A three phased approach was developed by Muller [18] to solve the problems in the ITC2007 exam timetabling track. The first phase consists of an Iterative Forward Search algorithm to find a feasible solution. Hill climbing is used to find the local optima in the second phase. Finally, a Great Deluge Algorithm is applied to further explore the search space.

Gogos et al. [14] proposed a method which used a GRASP (Greedy Randomised Adaptive Search Procedure). In the construction phase, five orderings of exams based on various criteria are generated. 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 used. Finally, room allocations are arranged using integer programming in the third phase.

Atsuta et al. [3] used a constraint satisfaction solver incorporating tabu search and iterated local search. The solver differentiates between the constraints and their corre-

sponding weights during computation to improve performance. De Smet [12] also incorporated local search techniques in a solver called Drools. Drools is an Open-Source Business Rule Management System (http://www.jboss.org/drools/).

Pillay [19] introduced a biological inspired approach which mimics cell behaviour. The exams are initially ordered using the saturation degree heuristic and scheduled sequentially in the available "cells" i.e. timeslots. If more than one timeslot is available, the slot which causes the least overall constraints violation is chosen. Rooms are chosen using the best fit heuristic. If a conflict occurs before all the exams are scheduled, the timetable is rearranged to lower the soft constraints violation. This is described as cell division. If the overall soft constraint violation is not improved without breaking of hard constraints, cell interaction occurs. The timeslots 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. [16] proposed a two phased approach where an adaptive heuristic is used to achieve feasibility during the first phase. The second phase improves the solution through the employment of a variant of the Great Deluge Algorithm.

2.4 Exam timetabling approaches for the Toronto Benchmark

An approach which uses a sequential construction method, employed by Caramia et al. [10], to assign exams in the least number of timeslots was able to produce the best quality timetables for four of the Toronto benchmark instances. It uses a greedy scheduler to obtain a feasible solution. A penalty decreaser and trader are then applied to improve the quality of the constructed solution. Burke et al. [6] introduced an approach which combines a variable neighbourhood search with a genetic algorithm which produced the best quality solution for one of the Toronto instances. In addition Burke et. al [5] proposed a method where a hill-climber compares the candidate solution with a solution produced a couple of iterations back instead of the current solution. This was called the "late acceptance criteria" and it produced the best quality solutions for another four instances. Yang et. al [26] employed Case-Based Reasoning to choose graph-heuristics to construct initial solution which were improved using a Great Deluge algorithm. This approach produced the best quality solution for one of the instances. The results obtained by these approaches are presented in section 4.1.

Recently, some new methods were investigated to automatically find the best heuristic to solve a set of instances. This has led to the introduction of Hyper-heuristics. The next section summarises some of the hyper-heuristic methods applied to the exam timetabling domain.

2.5 Hyper-heuristics in exam timetabling

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

A Tabu search was developed by Burke et al. [8] to optimise a search space of heuristic sequences comprised of two or more low-level heuristics. This work was extended in later research by Qu et al. [23] to construct heuristic sequences which produce feasible timetables. The combinations are then analyzed to find distribution patterns of low-level heuristics, based on which the heuristic sequences are adaptively adjusted to construct better timetables. In addition, hybridisations of the graph based hyper-heuristic with local search methods was investigated in [22].

Asmuni et al. [1] used fuzzy logic to combine two out of three graph colouring heuristics. The idea was to combine the two heuristics into a single value which calculates the difficulty of allocating an exam to a timeslot. The exams are ordered using this value and are scheduled in order. Furthermore, the approach was extended to tune the fuzzy rules instead of keeping them fixed [2].

Ersoy et al. [13] developed an approach called the hyperhill-climber where a hyper-heuristic is embedded in a memetic algorithm. The aim of this hyper-heuristic was to select the best hill-climber to apply or decide the best order in which hill-climbers are executed. In addition, Pillay et al. [20] created another approach where genetic programming was used to evolve hyper-heuristics.

Biligan et al. [4] presented different heuristic selection methods and acceptance criteria for hyper-heuristics in exam timetabling. Finally, a different method of combining heuristics was presented by Pillay et al. [21]. The low-level heuristics were combined hierarchically and applied simultaneously instead of being applied sequentially.

3 A Hyper-heuristic with low-level improvement heuristics

Several low-level heuristics can be used to improve a timetable with varying quality. The different low-level heuristics used could be considered as different methods for escaping from local optima. However, the order in which exams are moved and the type of move performed plays an important role in finding the best quality solution. An initial feasible solution is constructed using the Largest Degree heuristic where the exams in the ordering are assigned randomly to a timeslot causing the least penalty. Our objective is to analyse the performance of the different low-level heuristics used to minimise the penalty incurred from a constructed solution. In addition, we test the effect of using different orderings for the exams causing penalties in the solution. Finally we develop an adaptive approach which orders the exams causing violations and automatically selects the best heuristic to use for each exam to produce an improvement.

3.1 The low-level heuristics

In this paper we investigate the effect of using different low-level heuristics or neighbourhoods to improve timetables. A combination of two improvement low-level heuristics is used in our approach. The following is a list of the heuristics investigated:

1. Move Exam (ME): This heuristic selects an exam and reassigns it to the timeslot causing the least penalty.

2. Swap Exam (SE): This heuristic selects an exam and tries swapping it with a scheduled exam leading to the least penalty timetable.
3. Kempe Chain Move (KCM): This is similar to the SM heuristic but is more complex as it involves swapping a subset of conflicting exams in two distinct timeslots. This neighbourhood operator proved success when it was previously used in [**?**] and [25].
4. Swap Timeslot (ST) : This heuristic selects an exam and swaps all the exams in the same timeslot with another set of exams in a different timeslot. After testing all the timeslots, the swap producing the least penalty timetable is applied.

3.2 The random iterative hyper-heuristic

The study presented in this paper takes a similar approach to that presented in [23] where a random iterative hyper-heuristic generates heuristic sequences of different quality to solve the benchmark problem mentioned in section 2.1. Instead of using the heuristic sequences to construct solutions, they are used here to improve constructed feasible solutions by rescheduling exams causing penalties. Figure 1 presents the pseudo-code of this random iterative hyper-heuristic. The process starts by constructing an initial feasible solution. Since the initial solution constructed affects the improvement process, a random largest degree graph colouring heuristic which orders exams according to the number of conflicts each exam has with others is used [**?**]. This allows us to compare our approach to other approaches in the literature which use a similar method in construction. At every iteration, the exams causing violations in the constructed solution are identified and a random sequence of moves is generated. A move is the application of one of the low-level heuristics described in section 3.1. The sequence of moves is then applied to the sequence of exams as they are unscheduled one by one. Only moves that improve the current solution are accepted. If a move does not improve the solution, it is skipped and the exam stays in its current position. A sequence is discarded if an improvement is not obtained after the whole sequence is employed.

This approach was applied to four instances (hec92 I, sta83 I,yor83 I AND tre92) of the Toronto benchmark exam timetabling problems described in section 2.1 for off-line learning of the best heuristic hybridisations and the order of execution leading to the best improvement. After running this process for (ex50) times, where e is the number of exams causing soft constraint violations in the constructed solution, a set of sequences and the penalties of their corresponding solutions are obtained for further investigation on the effectiveness of the different heuristics used. Finally, an adaptive approach was developed and applied to the Toronto benchmark. Furthermore, to test the generality of the approach, it was applied to the ITC2007 exam timetabling track. The approach is presented in section 4.

3.3 Analysis of hybridising improvement low-level heuristics

In order to clearly observe the effect of the different low-level heuristics in improving solutions, the heuristic sequences generated consist of two heuristics. We use the Kempe chain move heuristic as the basic heuristic in the sequences as it has proved to be successful in previous work [**?**,25]. The Kempe chain move involves swapping a

construct a feasible solution using LD with random time-slot assignment
create a random ordered list of the exams contributing to the overall penalty incurred
for $i = 0$ to $i = e \times 50$ *e: number of exams causing penalty*
    for $n = 1$ to $n = e$
    initialise heuristic sequence h = [KCM KCM … KCM KCM] \\ h has size e
    h = randomly change n heuristics in h to SM, SS or ST
    construct a solution c using h
    if solution c is feasible
        save h and the penalty of its corresponding solution c

Fig. 1. The pseudocode of the random iterative hyper-heuristic with low-level improvement heuristics

subset of exams in two distinct timeslots making sure that a hard constraint violation does not occur. The rest of the heuristics (ME, SE and ST) are randomly hybridised into the list of KCM.

The random sequences are generated with different percentages of hybridisation by inserting n ME, SE or ST, n = [1,..,e] in the sequences. For each hybridisation of KCM with either ME, SE or ST, fifty samples are obtained for each amount of hybridisation.

We applied this approach to four instances of the Toronto benchmark exam timetabling problems [11]. Table 3 presents the results obtained using ME, SE and ST in a hybridisation with KCM as well as a comparison against using KCM only.

	hec92 I	yor83 I	sta83 I	tre92
KCM without hybridisation Best	13.50	43.84	160.43	8.99
KCM with ME Best	12.03	43.84	157.48	8.91
KCM with SE Best	12.03	42.37	157.75	8.75
KCM with ST Best	**11.30**	**41.79**	**157.27**	**8.57**

Table 3. Results using KCM without a hybridisation and with several different moves.

It is observed that using a Kempe chain only produces the worst results. After introducing other heuristics in a hybridisation with the Kempe chain moves, better results were obtained. Another observation from table 3 is that swapping timeslots and performing Kempe chain moves produces the best improvement for all the problems. One possible reason may be that swapping timeslots allows the search to be more diverse and to sample different areas of the search space to find good solutions faster. In addition, no obvious trends could be obtained on the amount of ST hybridisation within the best heuristic sequences. However, it is observed in all the sequences leading to the best timetables that the ST heuristic is randomly distributed within the sequence and the percentage of hybridisation is less than 50%.

3.4 Variations of Orderings of the exams causing a penalty

To analyse the effect of ordering the unscheduled exams causing a soft constraint violation in a previous solution, we decided to test different orderings while using the Kempe Chain and swapping timeslot hybridisation stated in the previous section. After the exams causing violations are identified, they are ordered first before being reassigned to a timeslot. Several orderings can be used to help guide the search as follows:

- Largest Degree (LD) : The exams are ordered decreasingly according to the number of conflicts each exam has with others.
- Largest Weighted Degree (LWD) : The exams are ordered similarly to LD but the exams are weighted according to the number of students involved in the conflict.
- Saturation Degree (SD) : The exams are ordered increasingly according to the number of remaining timeslots available to assign them without causing conflicts. In the case where ties occur, LWD is used as a tie breaker. From our previous work it was shown that SD produces the best results when LWD is used to break ties in the ordering [9].
- Largest Penalty (LP) : The exams are ordered decreasingly according to the penalty they incur in the current solution.
- Random Ordering (RO) : The exams are ordered randomly.

Table 4 presents the results of applying different orderings to the unscheduled exams, then running a random heuristic sequence of KCM and ST to assign them in better timeslots.

	hec92 I	yor83 I	sta83 I	tre92
KCM with ST + RO Average	11.99	42.63	159.74	8.91
KCM with ST + RO Best	11.60	41.33	158.46	8.64
KCM with ST + LD Average	12.15	42.09	159.39	9.00
KCM with ST + LD Best	11.32	39.69	157.76	8.66
KCM with ST + LWD Average	12.06	42.08	159.74	9.02
KCM with ST + LWD Best	11.39	39.69	157.49	8.66
KCM with ST + LP Average	12.69	42.10	163.32	8.91
KCM with ST + LP Best	12.50	39.69	159.50	8.51
KCM with ST + SD tb LWD Average	12.69	41.74	159.21	8.90
KCM with ST + SD tb LWD Best	**11.19**	**39.47**	**157.18**	**8.49**

Table 4. Results of hybridising KCM with ST using different orderings of the exams causing a soft constraint violation. The notation X tb Y means heuristic Y is used to break ties in heuristic X

As shown in table 4, we found that using SD and breaking any ties in the ordering using LWD produced the best results. This is because SD orders the unscheduled exams according to the number of timeslots available to assign them without causing conflicts. Therefore, the chances of moving exams at the top of the SD list and finding better

timeslots for them becomes higher. Ordering the exams according to the penalty they incur proved to be the second best ordering after SD. Another observation is that LD and RO performed randomly when applied.

4 Adaptive Selection of Low-level Heuristics for Improving Exam Timetables

Figure 2 presents the initialisation stage of the adaptive approach. The exams causing a penalty are first identified and are unscheduled. They are then put in a list and ordered using SD. Random heuristic sequences are generated using KCM and ST to reschedule the exams. The sequences are then applied to the ordered exams and the corresponding solutions are saved.

```
construct a feasible solution using LD with random time-slot assignment
create a list of the exams contributing in the overall penalty incurred ordered by SD
for i = 0 to i = e × 10, e: number of exams causing penalty
        for n = 1 to n = e
        initialise heuristic sequence h = {KCM KCM … KCM KCM}
        h = randomly change n heuristics in h to ST
        construct a solution using h
        if solution c is feasible
                save h and the penalty of its corresponding solution c
```

Fig. 2. The pseudocode of the initialisation stage of the adaptive hyper-heuristic with low-level improvement heuristics

The above observations indicate that the best solutions were obtained when ordering the exams causing violations using SD, and rescheduling them using either a Kempe-chain move or swapping timeslots. It was also observed that the heuristic sequences resulting in the best solutions used the same move for the majority of the exams (i.e. the same heuristic appears in the same position in the majority of the sequences). Therefore, we developed an intelligent approach that performs an analysis to the best 5% of the sequences produced to generate a new set of sequences. The new set of sequences obtained better results for all the problem instances. The adaptive approach was tested and showed to be effective and comparable with the best approaches in the literature.

Figure 3 presents the pseudo-code of the approach which hybridises ST with KCM in two stages. The process is presented as follows:

1. In the first stage, the best 5% of heuristic sequences are collected and analysed. If the same heuristic is used in more than 75% of the heuristic sequences, then it is stored. Otherwise the heuristic is neglected and the position is randomly assigned as KCM or ST.

2. n x 5 sequences for the large problems (uta92 I, uta92 II, car91 and car92) and n x 10 sequences for the small problems are generated, respectively. The new sequences are then applied to the problem.

```
construct initial heuristic sequences  // see Fig.2
collect the best 5 % of the heuristic sequences
for i = 0 to i < number of exams causing penalty
{
        count = 0
        for j = 0 to j < number of sequences
        {
                if heuristicSequence[i][j] = KCM
                count ++
        }
                if count > 0.75 * number of exams causing penalty
                        then finalHeuristicSequence[i] = KCM
                else if count < 0.25 * number of sequences
                        then finalHeuristicSequence[i] = ST
                else
                        finalHeuristicSequence[i] = empty
}
for i = 0 to i < n // n = number of empty positions * 5 for large problems
                   // or  number of  empty  positions * 10 for small problems
{
        for j = 0 to j < number of exams causing penalty
        {
                if finalHeuristicSequence[j] = empty
                        finalHeuristicSequence[j] = KCM or ST
        }
        construct a solution using finalHeuristicSequence[j]
        if a better solution is obtained
                save finalHeuristicSequence[j] and the penalty of its corresponding
                solution
}
```

Fig. 3. Adaptive generation of heuristic sequences hybridising KCM and ST

4.1 The Toronto Benchmark Results

We tested this approach on the Toronto benchmark exam timetabling problems and present the results in tables 5 and 6. The average computational time across the instances is also presented for 30 runs on a Pentium IV machine with a 1 GB memory.

	hec92 I	yor83 I	ear83 I	sta83 I	car92	car91	uta92 I	ute92	lse91	tre92	kfu93
AIH Average	12.69	41.74	38.98	159.21	4.49	5.39	3.56	27.97	11.45	8.90	15.54
AIH Best	11.19	39.47	35.79	157.18	4.31	5.19	3.44	26.70	10.92	8.49	14.51
Time(s)	397	1683	1692	759	41954	97961	61284	641	1466	4293	2745

Table 5. Results from the the adaptive improvement Hyper-heuristic (AIH) approach on the Toronto Benchmark dataset.

	hec92 II	yor83 II	ear83 II	sta83 II	uta92 II
AIH Average	12.43	50.49	41.98	35.00	3.54
AIH Best	11.35	49.72	39.60	32.57	3.45
Time (s)	498	1374	2792	1888	81316

Table 6. Contd. Results from the the adaptive improvement Hyper-heuristic (AIH) approach on the Toronto Benchmark dataset.

The best results stated in the literature are presented in table 7. These include the hill-climbing with a late acceptance strategy implemented by Burke et al. [5], the variable neighbourhood search incorporating the use of genetic algorithms used by Burke et al. [6], the sequential construction method developed by Caramia et al. [10] and the Case-Based Reasoning approach employed by Yang et al. [26]. These algorithms are described in section 2.4.

Problems	AIH Best	Burke(2008) Best [5]	Burke(2010) Best [6]	Caramia(2008) Best [10]	Yang(2005) Best [26]
hec92 I	11.19	10.06	10.00	**9.20**	10.83
sta83 I	157.18	157.03	**156.90**	158.20	158.35
yor83 I	39.47	**34.78**	34.90	36.20	36.35
ute92	26.70	24.79	24.80	**24.40**	25.39
ear83 I	35.79	32.65	32.80	**29.30**	33.70
tre92	8.49	**7.72**	7.90	9.40	7.92
lse91	10.92	9.86	10.00	**9.60**	10.35
kfu93	14.51	**12.81**	13.00	13.80	13.82
car92	4.31	**3.81**	3.90	6.00	3.93
uta92 I	3.44	3.16	3.20	3.50	**3.14**
car91	5.19	4.58	4.60	6.60	**4.50**

Table 7. Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to the best approaches in the literature on the Toronto Benchmark

The results obtained indicate the generality of our approach to different constructed timetables regardless of the size. We also make a comparison with other hyper-heuristics which produced the best results in the literature in table 8. In comparison with the graph-based hyper-heuristic in [8], our approach performs better in all the cases reported. In addition, it performs better in 8 out of 11 cases in comparison with the hyper-heuristics investigated in [21] and [22]. Finally, it performs better in 10 out of 11 cases compared to the Tabu search hyper-heuristic investigated in [15]. Only the problems presented in table 8 were compared to other results since the results for the other instances in table 1 were not reported in the literature.

Problems	AIH Best	Kendall(2004) Best [15]	Burke(2007) Best [8]	Pillay(2009) Best [21]	Qu(2009) Best [22]
hec92 I	**11.19**	11.86	12.72	11.85	11.94
sta83 I	**157.18**	157.38	158.19	158.33	159.00
yor83 I	**39.47**	-	40.13	40.74	40.24
ute92	**26.70**	27.60	31.65	28.88	28.30
ear83 I	**35.79**	40.18	38.19	36.86	35.86
tre92	8.49	**8.39**	8.85	8.48	8.60
lse91	**10.92**	-	13.15	11.14	11.15
kfu93	**14.51**	15.84	15.76	14.62	14.79
car92	4.31	4.67	4.84	4.28	**4.16**
uta92 I	3.44	-	3.88	**3.40**	3.42
car91	5.19	5.37	5.41	**4.97**	5.16

Table 8. Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to other hyper-heuristics approaches in the literature on the Toronto Benchmark

4.2 The International Timetabling Competition (ITC2007) Results

To test the generality of our approach, we applied it to the ITC2007 exam timetabling dataset. The initial solution is constructed by ordering the exams according to their saturation degree. The exams are assigned a random timeslot in the situation where more than one timeslot is available. After a feasible solution is constructed the Adaptive Improvement Hyper-heuristic was applied to the constructed solution. To allow a fair comparison with the reported competition results, the approach was run for the same amount of time using 11 distinct seeds for each instance. Table 9 presents the results we obtained in comparison with the best in the literature. The description of the approaches used for comparison are presented in section 2.3. We do emphasise that the objective here is not to beat the best reported results but to demonstrate the generality of our approach to different problems with different constraints. A dash in the table means that no feasible solution was achieved.

The Extended Great Deluge in [16] obtained the best results for 5 out of the 8 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 8 instances were reported in [18] using a three phased approach. The GRASP used in [14] produced the second best results.

In comparison to the Constraint Based Solver developed in [3], our approach performed better in 3 out of the 8 instances. The approach using the Drools solver in [12] obtained feasibility for only 5 instances. Our approach outperformed it as we were able to gain feasibility for all the 8 instances. This demonstrates the generality of our approach to solving exam timetabling problems. Finally, our approach performed better on 6 of the 8 instances in comparison with the biologically inspired approach proposed in [19].

Instances	AIH Best	McCollum(2009) Best [16]	Muller(2008) Best [18]	Gogos(2008) Best [14]	Atsuta(2008) Best [3]	De Smet(2008) Best [12]	Pillay(2008) Best [19]
Exam 1	6235	4633	**4370**	5905	8006	6670	12035
Exam 2	2974	405	**400**	1008	3470	623	3074
Exam 3	15832	**9064**	10049	13862	18622	-	15917
Exam 4	35106	**15663**	18141	18674	22559	-	23582
Exam 5	4873	3042	**2988**	4139	4714	3847	6860
Exam 6	31756	**25880**	26950	27640	29155	27815	32250
Exam 7	11562	**4037**	4213	6683	10473	5420	17666
Exam 8	20994	**7461**	7861	10521	14317	-	16184

Table 9. Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to the best approaches in the literature on the ITC2007 dataset

5 Conclusions

The study presented in this paper implements a hyper-heuristic approach which adaptively adjusts heuristic combinations to achieve the best improvement for constructed timetables. An investigation is made on the low-level heuristics used and the order in which exams causing soft constraint violations are rescheduled. The analysis is performed on a set of four benchmark instances of differing difficulty in an off-line learning process. It is shown that, of the heuristics tried, the best to combine with Kempe chains is swapping timeslots. In addition, better solutions are produced when ordering the exams causing a soft constraint violation using Saturation Degree and breaking any ties with Largest Weighted Degree. Based on the output of the learning process, an adaptive approach which analyzes and adjusts some randomly generated sequences is implemented and applied to the rest of the instances. Furthermore, the approach is applied to a more constrained dataset. The hyper-heuristic was tested and it produced very competitive results compared to other approaches in the literature on both datasets.

Future research directions include performing improvements during the timetable construction stage instead of performing the improvements at the end of the construction. Using hybridisations of more than two low-level heuristics could also be investigated. Finally, the approach investigated in this paper can be applied to course timetabling.

References

1. H. Asmuni, E.K. Burke, J. Garibaldi, and B. McCollum. Fuzzy multiple ordering criteria for examination timetabling. In E.K. Burke and M. Trick, editors, *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, volume 3616 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2004.
2. H. Asmuni, E.K. Burke, J. Garibaldi, B. McCollum, and A.J. Parkes. An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables. *Computers and Operations Research*, 36(4):981–1001, 2009.

3. M. Atsuta, K. Nonobe, and T. Ibaraki. Itc2007 track 1: An approach using general csp solver. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.
4. B. Biligan, E. Ozcan, and Korkmaz E.E. An experimental study on hyper-heuristics and exam timetabling. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference PATAT 2006*, volume 3867 of *Lecture Notes in Computer Science*, pages 394–412, 2007.
5. E.K. Burke and Y. Bykov. A late acceptance strategy in hill-climbing for examination timetabling problems. In *Proceedings of the conference on the Practice and Theory of Automated Timetabling(PATAT)*, 2008.
6. E.K. Burke, A. Eckersley, B. McCollum, S. Petrovic, and R. Qu. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research (EJOR)*, 206:46–53, 2010.
7. E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger, editors, *Handbook of Meta-Heuristics*, pages 457–474. Kluwer, 2003.
8. E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176:177–192, 2007.
9. E.K. Burke, R. Qu, and A. Soghier. Adaptive tie breaking and hybridisation in a graph-based hyper-heuristic for exam timetabling problems. *under review at European Journal of Operational Research*, 2009.
10. M. Caramia, P. Dell Olmo, and G.F. Italiano. Novel local-search-based approaches to university examination timetabling. *Informs Journal of Computing*, 20(1):86–99, 2008.
11. M.W. Carter, G. Laporte, and S.Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of Operational Research Society*, 74:373–383, 1996.
12. G. De Smet. Itc2007 - examination track. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.
13. E. Ersoy, E. Ozcan, and Uyar S. Memetic algorithms and hill-climbers. In P. Baptiste, G. Kendall, A.M. Kordon, and F. Sourd, editors, *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications Conference(MISTA2007)*, pages 159–166, 2007.
14. C. Gogos, P. Alefragis, and E. Housos. A multi-staged algorithmic process for the solution of the examination timetabling problem. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, 2008.
15. G. Kendall and N. Mohd Hussin. An investigation of a tabu search based hyper-heuristic for examination timetabling. In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, editors, *Selected Papers from MISTA 2005*, pages 309–328. Springer, 2005.
16. B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and S. Abdullah. An extended great deluge approach to the examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009), pp. 424-434, 10-12 August , Dublin, Ireland*, 2009.
17. B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, L. Di Gaspero, A. J. Parkes, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal of Computing, doi: 10.1287/ijoc.1090.0320*, 2008.
18. T. Muller. Itc 2007 solver description: A hybrid approach. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.
19. N. Pillay. A developmental approach to the examination timetabling problem. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.

20. N. Pillay and W. Banzhaf. A genetic programming approach to the generation of hyper-heuristic systems for the uncapicitated examination timetabling problem. In Neves et al., editor, *Progress in Artificial Intelligence*, volume 4874 of *Lecture Notes in Artificial Intelligence*, pages 223–234, 2007.
21. N. Pillay and W. Banzhaf. A study of heuristic combinations for hyper-heuristic systems for the uncapicitated examination timetabling problem. *European Journal of Operational Research*, 197:482–491, 2009.
22. R. Qu and E.K. Burke. Hybridisations within a graph-based hyper-heuristic framework for university timetabling problems. *Journal of Operational Research Society*, 60:1273–1285, 2009.
23. R. Qu, E.K. Burke, and B. McCollum. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392–404, 2009.
24. R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee. A survey of search methodologies and automated approaches for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.
25. J.M. Thompson and K.A. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annuals of Operations Research*, 63:105–128, 1996.
26. Y. Yang and S. Petrovic. A novel similarity measure for heuristic selection in examination timetabling. In *Practice and Theory of Automated Timetabling: Selected papers from the 5th International Conference .Lecture Notes in Computer Science*, volume 3616, pages 377–396, 2005.