J Sched (2012) 15:83–103
DOI 10.1007/s10951-011-0258-5

A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice

Peter Demeester · Burak Bilgin · Patrick De Causmaecker · Greet Vanden Berghe

Published online: 17 November 2011

Abstract Many researchers studying examination timetabling problems focus on either benchmark problems or problems from practice encountered in their institutions. Hyperheuristics are proposed as generic optimisation methods which explore the search space of heuristics rather than direct solutions. In the present study, the performance of tournament-based hyperheuristics for the exam timetabling problem are investigated. The target instances include both the Toronto and ITC 2007 benchmarks and the examination timetabling problem at KAHO Sint-Lieven (Ghent, Belgium). The Toronto and ITC 2007 benchmarks are post-enrolment-based examination timetabling problems, whereas the KAHO Sint-Lieven case is a curriculum-based examination timetabling problem. We drastically improve the previous (manually created) solution for the KAHO Sint-Lieven problem by generating a timetable that satisfies all the hard and soft constraints. We also make improvements on the best known results in the examination timetabling literature for seven out of thirteen instances for the To ronto benchmarks. The results are competitive with those of the finalists of the examination timetabling track of the International Timetabling Competition.

Keywords Exam timetabling · Benchmark instances · Hyperheuristics

P. Demeester (✉) · B. Bilgin · G. Vanden Berghe
Information Technology, KAHO Sint-Lieven, Gebroeders De Smetstraat 1, 9000 Gent, Belgium
e-mail: Peter.Demeester@kahosl.be

B. Bilgin
e-mail: Burak.Bilgin@kahosl.be

G. Vanden Berghe
e-mail: Greet.VandenBerghe@kahosl.be

P. De Causmaecker
Computer Science and Information Technology, Katholieke Universiteit Leuven Campus Kortrijk, Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
e-mail: Patrick.DeCausmaecker@kuleuven-kortrijk.be

1 Introduction

University timetabling was one of the first problems to motivate research on timetabling (Cole 1964). Many academics have their agenda determined by the results to a considerable extent. A course timetable, for example, is strongly constrained by the (limited number of) lecture, tutorial, seminar, and lab rooms, their respective properties, the number, size, and the hierarchical structure of student groups, and the availability of the lecturers. Increasing flexibility to combine different modules results in extra constraints on the students' opportunity to attend all the lectures and exams. Constructing timetables manually thus becomes more complex, and moreover it is a tedious and repetitive process. The quality of the final timetable affects lecturers and students alike, but it can also affect the administrative and technical services on a campus. If, for instance, all lectures scheduled before lunch break finish at the same time, the student restaurant will be flooded by hungry students. Exams that are not spread out sufficiently for all students, can have a serious impact on the number of students that pass their exams.

University timetabling comes in two groups (Schaerf 1999b): course and examination timetabling. In university course timetabling a distinction is made between post-enrolment and curriculum-based course timetabling. These were also the two course timetabling tracks of the second International Timetabling Competition (McCollum et al. 2010). Post-enrolment differs from curriculum-based course timetabling in that the timetable is constructed after the students

have chosen their courses. In the curriculum-based case, the students have to take a set of courses that belong to a certain curriculum. Many research papers address different problems or instances ranging from timetabling competitions (Kostuch 2005; Müller 2008) and (simplified) real world benchmark problems (Carter and Laporte 1996) to very complicated problems from practice that involve room and equipment requirements, flexible student groups, lecturer groups, flexible lecture durations, non-weekly repeating lab sessions, etc. (Beligiannis et al. 2008; Dammak et al. 2008; De Causmaecker et al. 2009; Di Gaspero and Schaerf 2008; Schaerf 1999a; van den Broek et al. 2009).

Examination timetabling is known to the research community as the problem of assigning exams to time slots and rooms in such a way that (1) no students are taking more than one exam at the same time, (2) the study intervals between exams are sufficiently long for each student, and (3) the capacity of rooms is respected. For this problem, competition instances (McCollum et al. 2010) as well as benchmark problems are available. The benchmark problems have been derived from several real world problems, e.g. at the universities of Toronto, Nottingham, Melbourne, etc. (see Qu et al. 2009 for a comprehensive overview).

The problem we introduce in this paper is the examination timetabling problem at the Engineering Department of KAHO Sint-Lieven. We will refer to it as the KAHO problem in the remainder of the paper. The problem under investigation is new. It differs from known examination timetabling problems in that it addresses both oral and written exams. We address this problem with a hyperheuristic approach and compare the timetable with the manual results that were generated in parallel. For the first semester of the academic year 2005–2006, we managed to arrange the exams on weekdays only, whereas the manual planner had not been able to avoid scheduling some exams on Saturdays. The performance of the hyperheuristic algorithm is tested on both the Toronto benchmark instances and the examination timetabling track of the second International Timetabling Competition (ITC 2007).

The contribution of this paper is threefold: (1) the introduction of a complex problem from practice in examination timetabling, (2) a tournament-based hyperheuristic approach consisting of (among other things) the recently introduced late acceptance strategy as a move acceptance criterion, and (3) a performance analysis of the solution approach, including a comparison with the best approaches to the benchmark problems in the recent literature. The claim of this paper is that the hyperheuristic approach can successfully address a complex problem from practice as well as benchmark instances from the scientific literature. There is no one to one match between the three problem descriptions, and the difference between the models is taken care of by the low-level heuristics.

The related literature is reviewed in Sect. 2, whilst a detailed problem description and a model are presented in Sect. 3. We elaborate on the difference between the problem from practice and two profoundly studied benchmark problems for examination timetabling. Section 4 introduces the components of the hyperheuristic approach. In Sect. 5, we discuss experimental results for the KAHO examination timetabling problem and the two benchmarks instances. We conclude the paper and point out some directions for future research in Sect. 6.

2 Related literature

2.1 Examination timetabling

In general, the literature on examination timetabling can be divided into two categories: one concentrates on solving examination timetabling problems from practice, whilst the other focuses on improving benchmark results. Benchmark instances give an unbiased comparison of the search algorithms' performance, on the condition that

- some validation tool that impartially checks the correctness of the obtained results exists,
- and a tool that eliminates the hardware dependency exists. The latter can be achieved, for instance, by providing a software tool that determines the hardware specifications of the researcher's computer and calculates the available computation time to obtain a solution for that specific problem (cfr. for example First International Nurse Rostering Competition 2010; McCollum et al. 2010).

The most studied examination timetabling benchmark is probably the one that was introduced by Carter et al. (1996). This benchmark consists of 13 simplified real world examination time tabling problems of different universities around the world. In contrast to examination timetabling problems from practice, these benchmark instances do not consider rooms and lecturers. It is assumed that rooms have infinite size. These instances are commonly known as the uncapacitated Toronto benchmarks. Until the introduction of the examination timetabling benchmarks of the Second International Timetabling Competition in 2007, the Toronto data sets were the standard test set for examination timetabling. The objective of the problem is twofold: constructing clash-free examination timetables for every student, and spreading out the exams as much as possible, such that students have sufficient study time. If only the no-clash constraint is taken into account, the problem reduces to a graph colouring problem (Burke et al. 2004b): exams correspond to nodes and edges correspond to pairs of exams that have at least one common student. Assigning exams to time slots corresponds to assigning colours to the nodes, such that connected nodes have different colours. Minimizing the number of time slots whilst satisfying the clash free constraint

Table 1 Problem size of the 13 data sets of the Toronto benchmark (version I). The 'conflict' density in the fifth column is the ratio of the number of non-zero elements in the conflict matrix to the total number of conflict matrix elements. In the last column, the minimum number of time slots of a feasible solution is presented

	# students	# exams	# enrolments	conflict density	# time slots
car91	16925	682	56877	0.13	35
car92	18419	543	55522	0.14	32
ear83	1125	190	8109	0.27	24
hec92	2823	81	10632	0.42	18
kfu93	5349	461	25113	0.06	20
lse91	2726	381	10918	0.06	18
pur93	30032	2419	120681	0.03	42
rye92	11483	486	45051	0.07	23
sta83	611	139	5751	0.14	13
tre92	4360	261	14901	0.18	23
uta92	21266	622	58979	0.13	35
ute92	2749	184	11793	0.08	10
yor83	941	181	6034	0.29	21

corresponds to minimizing the number of colours needed to colour all the nodes (or exams). Carter et al. (1996) introduced five basic sorting criteria for assigning exams to the time slots, namely: largest degree, saturation degree, largest weighted degree, largest enrolment, and random ordering.

Different versions of the Toronto instances are available. The instances used in this paper correspond to version I in Qu et al. (2009), which is the most commonly applied version in the literature. The specific properties of the 13 Toronto benchmark instances are summarized in Table 1. The values in the conflict density column (fifth column of Table 1) are the ratios of the total number of non-zero elements to the total number of elements of the 'conflict matrix'. The conflict matrix contains the number of common students for every exam. The conflict density value gives an indication of the number of conflicts between the individual exams. In the last column, the minimum number of time slots needed to obtain a feasible solution is presented.

The 'Benchmark Data Sets in Exam timetabling' website (Qu 2010), provides a collection of several examination time tabling benchmark data instances and a validator to evaluate the solutions of the Toronto benchmark data sets. Initiatives like these are welcomed since they provide a means to check the correctness of the objective function, cf. the discussion on measurability and reproducibility in Schaerf and Di Gaspero (2007).

Although numerous papers have been published on the uncapacitated Toronto benchmarks—we refer to Qu et al. (2009) for an overview—we only discuss those papers that consider the problem in a similar way. Thompson and Dowsland (1998) investigate the characteristics of a robust simulated annealing algorithm. It turns out that next to the cooling schedule, also the way the neighbourhoods are constructed and sampled is important. The Kempe chain-based neighbourhoods (which will be discussed in Sect. 4.1) outperform the other neighbourhoods. Thompson and Dowsland tackle the examination timetabling problem in two stages: the problem is first solved to feasibility and in the second stage, the quality of the obtained solution is improved without making the solution infeasible. Both stages apply different neighbourhoods. The second stage only employs neighbourhoods that do not violate any hard constraints. We will take these findings into account. Kendall and Hussin (2005a) apply a tabu search-based hyperheuristic approach in which the low-level heuristics execute small changes to the current solution. Low-level heuristics like these are called 'perturbative'. The heuristic selection criterion applies a tabu list for selecting a non-tabu heuristic in the next iteration. The tabu list length is equal to the number of heuristics. Although they do not improve on the best results in the literature, Kendall and Hussin demonstrate that it is possible to find good enough solutions with a very general approach that can easily be applied to other problems. A hyperheuristic approach to university timetabling problems, was also applied in Qu and Burke (2009). For tackling the Toronto benchmarks, they employed a graph-based hyperheuristic framework. Actually, high level heuristics such as steepest descent, iterated local search, tabu search, and variable neighbourhood search, are applied for searching sequences of low-level graph colouring heuristics, such as largest (weighted) degree, largest enrolment, colour degree, and saturation degree. The approach reported in Qu and Burke (2009) builds on the authors' experiences in Burke et al. (2007). In the latter paper tabu search is applied as a high level heuristic. In both articles (Burke et al. 2007; Qu and Burke 2009) the two search spaces are investigated: heuristics on the one hand, and solutions on the other hand. In Burke et al. (2010), a hybrid variable neighbourhood search approach to the Toronto benchmarks is described. The variable neighbourhood search is hybridized with a genetic algorithm, which is used for configuring the neighbourhoods. The reported method can improve on one of the data sets of the Toron to benchmarks. Burke and Bykov (2008) introduce the late acceptance strategy and apply it to Carter's examination timetabling benchmarks. For some of the instances, new best results are obtained. We will incorporate late acceptance among other acceptance criteria in our hyperheuristic approach.

The capacitated instances of the Toronto benchmarks were introduced in Burke et al. (1996) and further studied and improved in Abdullah et al. (2007a, 2007b), Di Gaspero and Schaerf (2001), Caramia et al. (2001), Merlot et al. (2003). We refer to Qu et al. (2009) for a review.

Ever since the introduction of the instances of the examination timetabling track of the Second International

Table 2 Problem size of the 12 data sets of the examination timetabling track of the Second International Timetabling Competition. The conflict density column indicates the number of conflicts between the exams. It is the ratio of the number of non-zero elements in the conflict matrix to the total number of conflict matrix elements

	# students	# exams	# rooms	conflict density	# time slots
Instance 1	7891	607	7	0.05	54
Instance 2	12743	870	49	0.01	40
Instance 3	16439	934	48	0.03	36
Instance 4	5045	273	1	0.15	21
Instance 5	9253	1018	3	0.009	42
Instance 6	7909	242	8	0.06	16
Instance 7	14676	1096	15	0.02	80
Instance 8	7718	598	8	0.05	80
Instance 9	655	169	3	0.08	25
Instance 10	1577	214	48	0.05	32
Instance 11	16439	934	40	0.03	26
Instance 12	1653	78	50	0.18	12

Timetabling Competition (McCollum et al. 2010), they have gained increasing popularity. In contrast to Carter's Toronto benchmarks, the new ones include more constraint types, which makes them more interesting and realistic. For instance, the room capacity constraint is now taken into account. The hard constraints are:

- a student cannot take more than one exam per time slot;
- an exam cannot be split over several rooms;
- the room's capacity cannot be exceeded;
- some exams require rooms with special properties;
- the duration of the time slot to which an exam is assigned, should be greater than or equal to the duration of the exam;
- some exams should be scheduled before, after, at the same time as or at different times than other exams.

The following soft constraints should be taken into account:

- two exams taken by the same student should not be scheduled on the same day or in two consecutive time slots;
- exams should be spread out as much as possible;
- exams with different durations should not be assigned to the same room;
- large exams should be scheduled early in the timetable;
- some of the time slots in the examination timetable should be avoided;
- some of the rooms should be avoided for examination.

The problem size of the 12 data sets is presented in Table 2. For more detailed information about the examination timetabling track of the Second International Timetabling Competition, we refer to McCollum et al. (2007, 2010).

The five finalists of the examination track of the competition have applied different approaches, such as:

- a constraint solver, which in the first stage constructs a complete feasible solution, followed by a hill climbing and a great deluge approach for improving the solution (Müller 2008);
- a Greedy Randomized Adaptive Search Procedure, followed by simulated annealing, and integer programming (Gogos et al. 2008);
- a constraint satisfaction solver combined with tabu search and iterated local search (Atsuta et al. 2008);
- Drools Solver (De Smet 2008), which is a combination of tabu search and the JBoss Drools rules engine used for the evaluation, and
- a heuristic method that is inspired by cell biology (Pillay 2008).

After the end of the competition, it was shown that for all 12 instances, feasible solutions could be obtained (McCollum et al. 2009). Also, McCollum et al. (2009) improve on six out of the 12 benchmark instances. Their approach consists of two parts: a feasible solution is constructed first and afterwards the solution is further improved with an extended great deluge algorithm, to which a reheating mechanism is added for escaping from local optima.

The runner-up of the competition, Gogos et al. (to appear), have published a follow-up paper in which they improve the approach applied in the competition. The method consists of two phases: a feasible solution is constructed, which is then further improved. The improvement phase consists of several parts:

- first a steepest descent local search is applied until a local optimum is reached;
- simulated annealing is applied for escaping from the local optimum;
- when no further improvement is obtained, the problem is divided into smaller problems, which are solved with integer programming;
- finally, if the previous stage cannot improve the current solution anymore, the solution is shaken.

Twenty percent of the computation time is spent on the construction phase, and the remainder is spent on the improvement phase. For some of the instances, the best results in the literature are obtained.

Besides the benchmarks, considerable attention has been paid to examination timetabling problems from practice (Ayob et al. 2007; Carter and Laporte 1996; Dimopoulou and Miliotis 2001; Hansen and Vidal 1995; Kendall and Hussin 2005b; Lim et al. 2000; Thompson and Dowsland 1996).

2.2 Hyperheuristics for examination timetabling

Hyperheuristics are high level search and optimization methods (Burke et al. 2003). Instead of operating directly on

a set of candidate solutions, as is the case in meta-heuristics, they operate on a set of (meta-)heuristics. These low-level heuristics can be either perturbative (changing only small parts of the solution) or constructive (constructing a solution). The motivation behind hyperheuristics is to provide a general framework for tackling a wide range of problems. In an ideal situation, the researcher should only adapt the low-level heuristics for solving a different problem than the current one. Another objective of the hyperheuristics research is to generate synergy between heuristics involved in the search, making use of their strengths, avoiding their weaknesses, and taking advantage of their combined capabilities.

Several papers applying hyperheuristics to examination timetabling problems have appeared in the last decade. Some of them are discussed in Sect. 2.1. Pillay and Banzhaf (2009) apply a constructive hyperheuristic to the Toronto benchmarks. The hyperheuristic chooses a combination of low-level heuristics for constructing a solution. Instead of applying the low-level heuristics sequentially, they are combined hierarchically and simultaneously. The low-level heuristics correspond to the five basic graph colouring criteria introduced by Carter et al. (1996). Without any improvement phase, the algorithm generates good quality results. For some data instances of the Toronto benchmarks, the results are better than those obtained by similar graph-based hyperheuristic approaches. Graph-based hyperheuristics for the Toronto benchmarks are also employed in Burke et al. (2005, 2007), Qu and Burke (2005, 2009). These four papers apply five basic graph colouring criteria (Carter et al. 1996) as low-level heuristics for constructing a solution, with different high level heuristics, such as tabu search, variable neighbourhood search, iterated local search, and steepest descent. Kendall and Hussin (2005b) describe the examination timetabling problem at the MARA university in Malaysia. They apply a tabu search-based hyperheuristic approach, selecting the low-level heuristics. The low-level heuristics combine the graph colouring heuristics and perturbative heuristics, swapping or moving random exams. Kendall and Hussin (2005a) apply the same approach to the Toronto benchmarks.

3 Problem description

3.1 The KAHO problem

Unlike the examples discussed in the literature, the KAHO Sint-Lieven examination timetabling problem combines both written and oral exams. Written exams of the same subject should be scheduled in the same time slot for all the students involved, whereas a lecturer can take 20 students at most in the same time slot for oral exams. Both an oral and a written exam should not be assigned to the same room at the same time. Different written exams can be organized in the same room simultaneously, provided the room capacity is not exceeded.

At KAHO Sint-Lieven, the examination capacity of a room ranges from one third to one half of its normal capacity for lectures. The actual ratio depends on the physical properties of the room.

At the beginning of the academic year, students are automatically enrolled for the exams that correspond to their curriculum. This causes—especially in the first year—a high percentage of students not showing up and a poor usage of the available rooms. Exams are assigned to rooms that are larger than needed.

An exam takes at most half a day (from 8h30 till 12h30 or from 14h00 till 18h00) and should only be scheduled on workdays. Students should have at least 3 half days of study time between two consecutive exams. A typical examination period takes 4 weeks of 5 days each, resulting in 40 available time slots.

We solve the data set for the first semester of the academic year 2005–2006. This was the last semester in which exams were scheduled on Saturdays. The manual planner actually needed 24 days (or 48 time slots) for scheduling the exams. Apart from satisfying the above constraints, the aim is to eliminate the exams on Saturdays as well.

The problem includes:

- 336 exams,
- 135 student groups,
- 71 rooms of varying size and
- 40 time slots of 4 hours each.

Note that we do not take the assignment of invigilators into account. In practice, for oral exams, the examiner is invigilator at the same time, whilst for written exams, teaching assistants are invigilating. We do plan to incorporate the assignment of invigilators in future work.

3.2 Constraints and objective

In this section, we identify the different constraints of the KAHO Sint-Lieven examination timetabling problem and distinguish between hard and soft constraints. A solution containing hard constraint violations cannot be executed in practice. A solution satisfying the hard constraints is called a *feasible* solution. Soft constraints do not really need to be satisfied, but if they are, they will improve the quality of the solution. The goal is to search for feasible solutions that satisfy the soft constraints to the highest possible extent.

The *hard constraints* imply:

- Students cannot take two exams at the same time.
- The number of students assigned to a room cannot exceed its capacity.

– All exams should be scheduled within the planning horizon of four weeks.

Although not all the following rules are considered to be *soft constraints* at KAHO Sint-Lieven, we rank them as soft since they do not make the solution 'physically infeasible' when violated. All soft constraints get the value 1 as weight.

– All written exams of the same subject should be scheduled in the same time slot.
– All oral exams should be scheduled such that the maximum number of examinees per time slot is at most 20.
– Lecturers who are responsible for oral exams cannot examine more than one student group at the same time.
– Oral and written exams should not be merged into the same room.
– Students should have at least three slots of study time between two consecutive exams. This corresponds to the spreading out constraint (Corne et al. 1994; Ross et al. 1994).

The objective of the problem is to find a feasible solution that minimizes the weighted sum of the violations of the soft constraints.

3.3 Mathematical formulation

In examination timetabling, E exams need to be scheduled into a limited number of T time slots and R rooms, taking into account the capacity of the rooms, and the restriction that students cannot take more than one exam at the same time, among other constraints. Every student group takes a number of exams. In the examination timetabling problem at hand, exams depend on the student group's curriculum, which is similar to the situation in the curriculum-based university course timetabling case.

The decision variables are

$$x_{rte} = 0 \text{ or } 1 \tag{1}$$

with $(r = 1, \ldots, R;\ t = 1, \ldots, T;\ e = 1, \ldots, E)$, and where $x_{rte} = 1$ if exam e is assigned to room r in time slot t. An exam e is characterized by a subset of the set of all student groups, taking the same course and a lecturer l (with $l = 1, \ldots, L$, with L the number of lecturers). A student group is characterized by s $(s = 1, \ldots, S$, with S the number of student groups).

– Let S_{rt} be the total number of students assigned to room r at time slot t.
– Let W denote the set of all written exams and O the set of all oral exams. Since no exam can be both written or oral, this means that $W \cap O = \emptyset$.
– Let N_e denote the number of student groups taking the same exam e of the same course by the same lecturer.
– Let CAP_r be the capacity of room r.
– Let A denote the conflict matrix. This matrix represents the exams that cannot be scheduled in the same time slot ('conflicting exams'), due to exams that have common students. This is a symmetric $E \times E$ matrix. The matrix elements on the diagonal are 0, whilst the values of the non-diagonal matrix elements correspond to the number of students that the involved exams have in common.

$\forall$ rooms r, time slots t, and exams e:

$$\sum_{m=1}^{E} A_{me} x_{rtm} \leq M(1 - x_{rte}) \tag{2}$$

with M an arbitrary large number.

Equation (2) expresses that no student group can take more than one exam in the same time slot.

$$\forall e, r, t : S_{rt} x_{rte} \leq CAP_r \tag{3}$$

Equation (3) expresses that the number of students assigned to a room cannot exceed the room's capacity.

$$\forall e, r : \sum_{t=1}^{T} x_{rte} = 1 \tag{4}$$

Equation (4) expresses that all exams are scheduled.

Solutions satisfying (2)–(4) are feasible. In order to find feasible solutions that satisfy as many soft constraints as possible, the following objective function (5) is introduced.

$$\text{Min}\left(\sum_{t=1}^{T}\sum_{e\in W} P_{1,te} + \sum_{e\in O} P_{2,e} + \sum_{r=1}^{R}\sum_{e=1}^{E}\sum_{t=1}^{T} P_{3,rte} + \sum_{e=1}^{E}\sum_{r=1}^{R} P_{4,er}\right), \tag{5}$$

with P_1, P_2, P_3, and P_4 the result of the violations of the soft constraints. The penalties corresponding to the violations of the soft constraints are expressed below.

$$\forall e \in W, \forall t : P_{1,te} = \left| \sum_{r=1}^{R} x_{rte} - N_e \right| \tag{6}$$

Equation (6) generates a penalty when the written exams of the same subject are not assigned to one single time slot.

$$\forall e \in O : P_{2,e} = \left| \sum_{r=1}^{R}\sum_{t=1}^{T} x_{rte} - N_e \right| \tag{7}$$

Equation (7) requires that the number of oral exams of the same course should equal the number of student groups taking the course.

$$\forall e_1 \in W, e_2 \in O, \forall r, t : \quad P_{3,rte} = \max\left((x_{rte_1} + x_{rte_2}) - 1, 0\right) \tag{8}$$

Table 3 Differences and similarities between the Toronto and ITC 2007 benchmark and the KAHO Sint-Lieven examination timetabling problem

	KAHO	ITC 2007	Toronto
exam types	oral AND written	written	written
smallest unit	student group	student	student
exam durations	equal	different	equal
exam order	not important	important	not important
room size	important	important	not important

Equation (8) expresses that an oral and a written exam cannot be assigned to the same room in the same time slot.

$$\forall e, r : P_{4,er} = \sum_{t=1}^{T-1} x_{rte} x_{rt+1e} + \sum_{t=1}^{T-2} x_{rte} x_{rt+2e} + \sum_{t=1}^{T-3} x_{rte} x_{rt+3e} \tag{9}$$

Equation (9) requires that for each student group, at least 3 time slots of study time should be assigned between two exams.

3.4 Benchmarks versus KAHO problem

In Table 3 we summarize both the differences and similarities between the KAHO problem and the two best known examination timetabling benchmarks in the literature: the Toronto benchmark and the ITC 2007 data sets. Note that the KAHO problem differs from the other problems in the considered examination types and in the smallest examinee unit. Since the Toronto and ITC 2007 instances only consider written exams, they do not impose special constraints on the maximum number of students that can take the exam at the same time. Contrary to the two benchmark problems, the KAHO problem is a curriculum-based examination timetabling problem. It means that exams at KAHO are organized per student group taking the same curriculum, whilst in the Toronto and ITC 2007 benchmark instances, every student may have an individual examination timetable. In the KAHO case, all exams take 4 hours, whereas in the ITC 2007 case, exams may have different durations. Also, the exam order is important in the ITC 2007 case: large exams should preferably be scheduled in the beginning of the period. The KAHO and the ITC 2007 case take into account the (limited) room size, whilst the Toronto instances consider infinite room sizes. Actually, except for the lack of oral examinations, and the notion of curricula, the ITC 2007 model can be considered as more general than the KAHO model. It foresees exams with different durations, and it prefers exams that are scheduled early in the planning period. If it would be extended with exams based on curricula and with the notion of oral exams, our model would perfectly fit into the ITC model.

4 Solution methods

In this section we introduce the components of the hyperheuristic approach. Section 4.1 introduces the low-level heuristics. The construction of the initial solutions for the three examination timetabling problems is discussed in Sect. 4.2. In Sect. 4.3 we present the four acceptance criteria and the heuristic selection method.

4.1 Low-level heuristics

The hyperheuristics framework is built on top of a local search framework, for which an appropriate solution representation is generated. We mimic the timetabling process of a human planner as closely as possible. The rows of a two dimensional matrix correspond to the rooms and the columns to the time slots. The number of time slots corresponds to the examination period. A room-time slot combination can hold several exams, which can individually be moved to another room-time slot combination. The number of exams in the same room-time slot combination is limited by the room's capacity.

In order to overcome the under-utilization of rooms for the KAHO problem (Sect. 3.1), we introduce a show up factor: this factor represents the percentage of enrolled students that are likely to attend the exam. Throughout this paper we assume that 95% of the students show up. This estimate is thought to be safe for the first year's students.

We distinguish between the heuristics that were developed for solving the capacitated KAHO and ITC 2007 examination problems and good heuristics from the literature (Sect. 2) on the uncapacitated Toronto examination timetabling problem. We address the first set as the capacitated and the latter as the uncapacitated heuristics:

- All the capacitated heuristics rely on one basic move: relocating an exam to another room-time slot combination. In case the capacity of the destination room is insufficient, the content of both matrix elements is swapped; otherwise, the exam is simply moved from the original to the destination position. The heuristics restrict the directions in which the content can be moved. The four heuristics that both capacitated examination timetabling problems have in common are:
  - **Cap**$_1$: This heuristic restricts the moves to random room-time slot combinations. The destination room and time slot need to be different from the original room and time slot.
  - **Cap**$_2$: This heuristic only allows moves in the same time slot to rooms that have enough excess capacity to accommodate the students taking the exam.
  - **Cap**$_3$: A randomly chosen exam is moved to a randomly chosen room, within the original time slot.

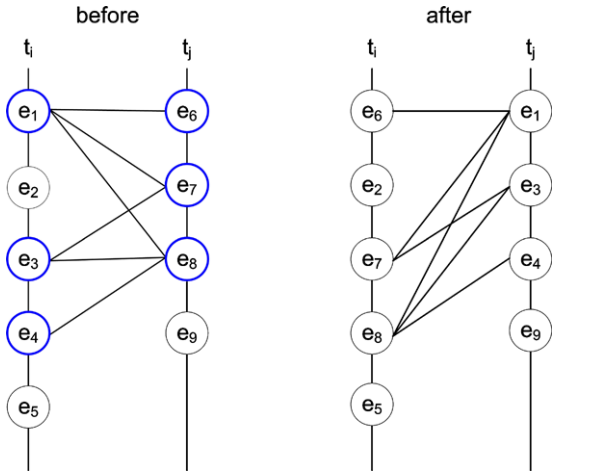

Fig. 1 An example of the Kempe chain heuristic. A move of exam e_1 in time slot t_i to another time slot t_j requires repair moves to maintain feasibility. Exams e_3 and e_4 have to be moved to time slot t_j and exams e_6, e_7, and e_8 have to be moved to time slot t_i

  - **Cap**$_4$: A randomly chosen exam is moved to a randomly chosen time slot, whilst maintaining the original room.

  The ITC 2007 examination timetabling problem has some extra soft constraints. Therefore we constructed an additional set of low-level heuristics that address these constraints in particular:
  - **ITC**$_1$: A randomly chosen exam is moved to another time slot in the same room thereby avoiding time slots that introduce an extra period penalty.
  - **ITC**$_2$: A randomly chosen exam is moved to another room in the same time slot, avoiding rooms that introduce an extra room penalty.
  - **ITC**$_3$: A randomly chosen large exam is moved to a time slot in the beginning of the examination period.

  None of the above heuristics takes special precautions to maintain the feasibility of a solution.
- Concerning the uncapacitated Toronto examination timetabling problem, we apply the *Kempe chain* based heuristics. These low-level heuristics perturb feasible solutions to the uncapacitated examination timetabling problem, without making them infeasible. Suppose a partial solution that corresponds to the left hand side of Fig. 1. If we want to move exam e_1 to time slot t_j, the solution becomes infeasible, since exam e_1 has students in common with exams e_6, e_7, and e_8 that are assigned to time slot t_j. To overcome this, exams e_6, e_7, and e_8 should be moved to time slot t_i. This process is repeated until all the exams that have students in common are assigned to different time slots. The result is depicted at the right hand side of Fig. 1. The Kempe Chain heuristic **Uncap**$_1$ can be seen as moving an exam to another time slot whilst maintaining feasibility by repairing any infeasibilities that may have been introduced. The swap-two-time slots heuristic **Uncap**$_2$, introduced in Burke and Bykov (2008), swaps all exams of two randomly chosen time slots. This heuristic maintains feasibility since swapping exams includes all exams assigned to a time slot.

4.2 Constructing initial solutions

Both the KAHO, the Toronto and the ITC 2007 examination timetabling track data sets are tackled in the same way. The approaches only differ in the applied low-level heuristics and in the way the initial solutions are constructed.

- For the KAHO problem, we do not spend much effort in constructing a good quality initial solution. All the exams are randomly assigned to free room-time slot combinations.
- The same random initialization procedure was applied to the Toronto instances, but too many iterations were required for obtaining a feasible solution for most of the problem instances. For this reason, we decided to make use of the properties of the conflict matrix (Sect. 2) to construct a feasible solution. The initialization phase is described in Algorithm 1. Starting from the conflict matrix, we assign exams to time slots, such that no student takes two exams at the same time. The procedure assigns exams in order of decreasing number of conflicts, which corresponds to the largest degree node colouring heuristic (Carter et al. 1996). This is comparable to what many human planners do. They start assigning the exams that have a lot of constraints, so that the exams with a lower number of constraints are planned in the end. Some noise is added by randomly assigning exams that cause no conflicts with exams that were assigned before to one of the non-conflicting time slots. If no feasible solution can be found, the conflicting exams together with the randomly assigned non-conflicting exams are removed from the solution. After that, the removed exams are reassigned following the same procedure. This process is repeated until a feasible solution is found or until the total number of attempts exceeds a threshold value. In the latter case, the (infeasible) solution with the lowest number of unassigned exams is the new solution, and the unassigned exams are randomly assigned to time slots. The Cap_i $(i = 1, \ldots, 4)$ heuristics are then applied to obtain a feasible solution through perturbations. Although these heuristics do not guarantee that they can make the solution feasible, the experiments always resulted in a feasible solution in a short amount of computation time.
- In the ITC 2007 examination timetabling problem, the exams are ranked in decreasing order according to the number of students taking the exam. Exams are assigned to a room such that both the capacity and duration constraints are satisfied. This process is repeated for the first T exams. The remaining exams are assigned to room-time slot combinations as follows:

Algorithm 1 Pseudocode for initializing the solution of Toronto instances

U_E = list of unassigned exams
C_{E_i} = list of exams that are in conflict with exam E_i
L_E = ordered list of exams, decreasing order of number of exam conflicts
MAXVALUE = 5000
//At the start of the algorithm none of the exams are assigned
$U_E = L_E$
$i = 0$
//Start with the exam that has most conflicts with other exams
for $i < E$ **do**
  //Take the i th element of U_E
  exam $e = U_{E_i}$
  //Find a non-conflicting time slot
  T_e = list of non-conflicting timeslots for e
  if $|T_e| > 0$ **then**
    assign exam e to a randomly chosen time slot from T_e
    remove e from U_E
  end if
  $i = i + 1$
end for
//If there are still unassigned exams
$i = 0$
while $|U_E| > 0$ AND $i <$MAXVALUE **do**
  unassign the assigned exams and try to reassign them together with as many as possible unassigned exams following the procedure described in the above for loop
  $i = i + 1$
end while
if $|U_E| > 0$ **then**
  randomly assign the exams in U_E
  apply the Cap_i heuristics until a feasible solution is obtained
end if

– First, the algorithm checks whether the exam can be assigned to one of the occupied room-time slot combinations. If there is sufficient capacity left and the exam's duration is not greater than the duration of the time slot, the exam is assigned to the first appropriate room-time slot combination that is encountered;
– If none of the occupied room-time slot combinations have sufficient capacity to accommodate the exam, the exam is randomly assigned to an empty room, taking into account that room can accommodate the number of students taking the exam. In addition, the duration of the exam should be less than or equal to the duration of the corresponding time slot.

The construction of the initial solution does not guarantee the feasibility of the solution, since we only take two of the five hard constraints into account. Algorithm 2 presents the initialization phase.

Algorithm 2 Pseudocode for the initialization of the solution of ITC 2007 exam instances

L_E = ordered list of exams, in decreasing order of number of students
U_E = list of unassigned exams
A_E = list of assigned exams
$U_E = L_E$
//First phase: only for the first T exams
$i = 0$
for $i < T$ **do**
  assign exam U_{E_i} to timeslot-room combination such that capacity and duration constraints are satisfied
  remove exam U_{E_i} from U_E
  add exam U_{E_i} to A_E
  $i = i + 1$
end for
//Second phase: for the remaining exams
$i = 0$
while $|U_E| > 0$ **do**
  check whether exam U_{E_i} can be assigned to a room-timeslot combination that is already taken by the exams in A_E, taking into account capacity and duration constraints
  if check is successful **then**
    assign exam U_{E_i} to occupied room-timeslot combination
    remove exam U_{E_i} from U_E
    add exam U_{E_i} to A_E
  else
    assign exam U_{E_i} to random free room-timeslot combination, taking into account capacity and duration constraints
    remove exam U_{E_i} from U_E
    add exam U_{E_i} to A_E
  end if
end while

4.3 Hyperheuristics framework: move acceptance criteria, heuristic selection method and tournament factor

A typical hyperheuristics framework includes a heuristic selection mechanism and move acceptance criteria. The first is a mechanism for selecting a heuristic, to be applied to a single candidate solution. The latter determines whether the resulting candidate solution is accepted or declined. Several well-known meta-heuristic methods have been deployed

within hyperheuristics to serve as selection methods or as move acceptance criteria.

The hyperheuristic framework applied in this paper is influenced by the work of Özcan et al. (2008) and Bilgin et al. (2007). We have extended this hyperheuristic framework with a *tournament factor*. At each iteration, the selected heuristic generates a predefined number, namely the tournament factor, of random moves. The move leading to the candidate solution with the lowest fitness value is selected. Evaluation of a solution by the fitness function results in a value that reflects the solution quality: the lower the value, the better the solution. If the best move is accepted by the acceptance criterion, it modifies the current solution by executing the selected move.

This design enables application to a wide range of problems. This tournament-based hyperheuristic framework has been applied successfully applied to patient admission scheduling (Bilgin et al. 2010). This problem consider patients that need to be assigned to hospital beds taking into account medical and personal constraints. The goal is to assign patients to the room type of their choice, whilst at the same time satisfying their medical needs (Demeester et al. 2010).

We apply the 'simple random' *heuristic selection method*, which randomly chooses a heuristic from a fixed list at each iteration. This is actually the simplest heuristic selection method possible.

We consider the following four *move acceptance criteria* for the experiments:

Algorithm 3 Pseudocode of the simulated annealing acceptance criterion

T = total execution time
R = remaining execution time
C_i = fitness value of candidate solution at ith iteration
C^i_{best} = fitness value of the best solution at ith iteration
P_i = random number between [0,1[at ith iteration
$\delta = C_{i+1} - C_i$
if $\delta \leq 0$ **then**
  Accept
else
  if $P_i < exp^{\frac{-\delta T}{(T-R)}}$ **then**
    Accept
  else
    Reject
  end if
end if

– 'improving or equal', abbreviated as **IE**, which only accepts moves that do not worsen the solution;
– simulated annealing, abbreviated as **SA**, is represented in Algorithm 3. In contrast to Burke et al. (2004a), in which the authors report spending considerable time tuning the problem specific parameters in their simulated annealing algorithm, the acceptance criterion considered here does not contain any parameters that need to be tuned.
– great deluge, abbreviated as **GD**, is presented in Algorithm 4

Algorithm 4 Pseudocode of the great deluge acceptance criterion

T = Total Execution Time
R = Remaining Execution Time
C_i = Fitness value at ith iteration
C_0 = Initial fitness value
D = R/T
if $C_{i+1} \leq C_i$ **then**
  Accept
else
  if $C_{i+1} < C_0 * D$ **then**
    Accept
  else
    Reject
  end if
end if

Algorithm 5 Pseudocode of the steepest descent late acceptance strategy acceptance criterion

initialize list C^*_L: fill complete list of fixed length L with the initial solution's fitness value
C^*_v = fitness value in the list C^*_L at the ith iteration with $v = i \,(\text{mod}) L$
C_i = fitness value of candidate solution at ith iteration
$\delta = C_{i+1} - C_i$
if $\delta \leq 0$ **then**
  replace C^*_v by C_{i+1}
  Accept
else
  if $C_{i+1} \leq C^*_v$ **then**
    replace C^*_v by C_{i+1}
    Accept
  else
    replace C^*_v by C_i
    Reject
  end if
end if

– an adapted version of the late acceptance strategy (Burke and Bykov 2008), abbreviated as **LA**. Similar to tabu search, the late acceptance strategy makes use of a list of a given length L. The acceptance list does not contain properties of tabu moves but the values of the previous

L candidate solutions. Rather than comparing a candidate solution with the current solution, it is compared with the oldest value in the list. When the candidate solution's fitness value is less than the oldest value in the list, the candidate solution is accepted, and its fitness value replaces the oldest value in the list. The late acceptance variant that we introduce in this paper first compares the candidate solution's fitness value with the current solution's fitness value, which is actually a (limited) steepest descent. It is no steepest descent in the strict sense, since only a small part of the neighbourhood is explored. In case the candidate solution is worse than the current solution, the original late acceptance strategy is applied. Algorithm 5 summarizes the steepest descent late acceptance strategy algorithm.

The hyperheuristic presented in this article, resembles the approach discussed in Bai et al. (2007), Bai and Kendall (2005). Bai et al. (2007) apply a simulated annealing hyperheuristic to problems from nurse rostering, university course timetabling and bin packing. Their heuristic selection mechanism starts with random selection, and gradually learns which low-level heuristics perform better than the other ones. As the search proceeds, the better performing low-level heuristics gradually increase their chances of being selected. Experiments show that this method leads to better solutions than some bespoke meta-heuristics. The approach presented in the current paper differs from Bai et al. (2007) in the move acceptance criteria, and in the use of the tournament factor.

5 Experiments

5.1 Settings

Every hyperheuristic variant is executed ten times on each problem instance. Throughout this section, all the quantitative comparisons are assessed using the Student t-test method with a 95% confidence level.

The applications were implemented in Java. All experiments were performed on Intel Core2Duo (3 GHz) PCs running Windows XP Professional SP3, with a Java 1.6 JRE configured to run in server mode with a heap size of 128 MB.

The stopping criteria differ between problems:

- one hour for the KAHO problem;
- for the Toronto benchmarks the stopping criteria range from 1 hour to 12 hours.
- for the ITC 2007 data sets, the stopping criterion depends on the performance of the computer on which the algorithms are executed. In order to guarantee a fair comparison between the different approaches, the competition organizers released a benchmark program that, dependent on the contestant's computer performance, shows the available computation time to find a solution. In our case the stopping criterion was fixed at 300 seconds.

We also compare the effect of the original late acceptance strategy (Burke and Bykov 2008) and the late acceptance strategy combined with the steepest descent algorithm. From the experiments it is clear that the combination of the late acceptance strategy and steepest descent leads to better results for the KAHO examination problem. The experiments show that the length of the acceptance list plays a crucial part in the success of the algorithm. On the other hand, the performance difference between the two late acceptance strategies for the Toronto and ITC 2007 data sets is not that clear. For some of the instances, the original late acceptance strategy outperforms the adapted version and vice versa.

5.2 The KAHO problem

Once a random initial solution is generated, the hyperheuristic adapts the solution with perturbative heuristics (Sect. 4.1) until a feasible solution is obtained. By attributing a high weight (100 in this case) to the violations of the hard constraints in the fitness function, the search is biased towards finding a feasible solution whilst trying to improve the quality. The weights corresponding to the soft constraints have all been set equal to 1. Since none of the low-level heuristics remove exams from the timetable, the third hard constraint of the KAHO problem (Sect. 3.2) is always satisfied.

Table 4 presents the experimental results for the KAHO case. For the hyperheuristic with the simulated acceptance criterion with tournament factor 4 (SA-4), we experimented with different types of low-level heuristics. In the first experiment, we only have applied the Cap_i $(i = 1, \ldots, 4)$ heuristics, whilst in the second experiment, the four Cap_i and two $Uncap_i$ heuristics were applied. From Table 4 it is clear that only applying the Cap_i heuristics leads to better results. Based on these findings, we decided to only employ the Cap_i heuristics for the further experiments. The steepest descent variant of the late acceptance strategy outperforms the original late acceptance strategy for each of the tournament factors in a statistically significant way. The difference between the two best performing approaches, i.e. the simulated annealing acceptance criterion with tournament factor 4 and the combination of the late acceptance strategy and steepest descent with tournament factor 8 (LA-SD-8) is not statistically significant.

Several tests were performed to obtain the ideal list length for the late acceptance criterion. In Fig. 2, the average fitness over 10 runs versus the list length is plotted for the two late acceptance strategies. The best list length for both strategies is 10000. Compared to the other move acceptance criteria (i.e. simulated annealing, 'improving or equal', and great deluge), the dependence of both late acceptance criteria's performance on one parameter is a disadvantage. The

Table 4 Experimental results of the five algorithms applied to the KAHO problem. Each algorithm was run for 10 times, with a computation time of 1 hour. Numbers in italic correspond to the best value obtained with the acceptance criterion

SA with Cap_i heuristics					
Tournament factor	4	8	16	32	64
AVG	*1*	1.1	1.1	2.9	6.1
MIN	*0*	*0*	*0*	*0*	3
STDEV	1.15	1.6	*1.1*	1.79	2.85
SA with Cap_i and $Uncap_i$ heuristics					
Tournament factor	4	8	16	32	64
AVG	21.9	15.5	4.4	4.1	*3.9*
MIN	14	11	1	1	*0*
STDEV	4.72	3.5	2.84	*2.33*	2.38
IE with Cap_i heuristics					
Tournament factor	4	8	16	32	64
AVG	*3.7*	4	5.1	4	5.4
MIN	*0*	*0*	1	1	2
STDEV	*1.89*	2	2.64	2.83	2.27
LA-SD with Cap_i heuristics with list length 10000					
Tournament factor	4	8	16	32	64
AVG	1.6	*0.3*	1.7	3.8	4.1
MIN	*0*	*0*	*0*	1	*0*
STDEV	1.37	*0.45*	1.44	2.11	2.46
LA with Cap_i heuristics with list length 10000					
Tournament factor	4	8	16	32	64
AVG	6.4	5.5	*4.7*	8.4	10.2
MIN	3	*1*	2	3	4
STDEV	3.31	3.63	2	2.88	3.82
GD with Cap_i heuristics					
Tournament factor	4	8	16	32	64
AVG	*1.2*	3.6	2.7	3.1	4.2
MIN	*0*	1	1	*0*	1
STDEV	1.32	1.58	*1.16*	3	2.35

length of the acceptance list influences the algorithm's performance on the KAHO problem. From the experiments it is also clear that the late acceptance strategy combined with steepest descent is less dependent on the list length than the original late acceptance approach.

The experiments show that all algorithms, except the late acceptance strategy, generate examination timetables satisfying all the hard and soft constraints. The timetable is organized in 40 time slots, instead of the 48 time slots that were needed by the manual planner.

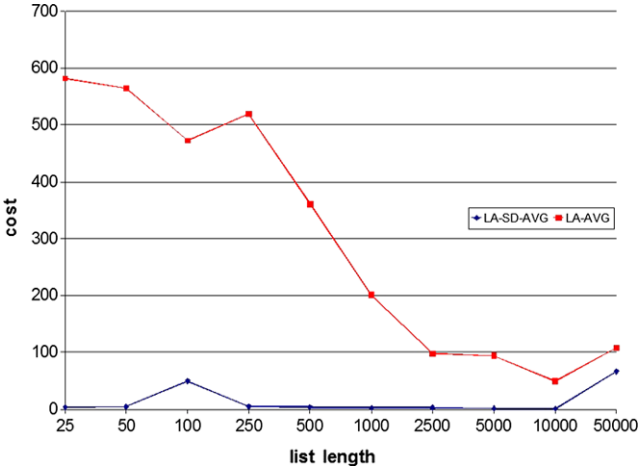

Fig. 2 Average result after ten runs for different values of the list length (tournament factor 16)

5.3 Toronto benchmark problems

The constructed feasible initial solution (Sect. 4.2) is locally perturbated applying only the two heuristics $Uncap_1$ and $Uncap_2$. This corresponds to the findings of Thompson and Dowsland (1998) who also solve the examination timetabling problem in two distinct phases (Sect. 2). The Cap_i heuristics are not used here since they do not maintain feasibility. Preliminary experiments have shown that once a feasible solution becomes infeasible it is really hard to turn it into a feasible one again by only using the Cap_i heuristics. The Toronto benchmark data sets only consider spreading out individual students' exams to the highest possible extent and avoiding student clashes. The $Uncap_i$ heuristics appear to be perfectly suited for realizing this:

- after execution of one of the heuristics the no-clash hard constraint is still satisfied;
- the heuristics manage to spread out the exams at the same time.

Although these two heuristics maintain feasibility (Sect. 4.1), they are not able to turn an infeasible solution into a feasible one. Therefore, we have opted for constructing a feasible solution in a preceding phase and then to improve the solution further in the improvement phase.

In Tables 5 and 6, the results (validated with the function in Qu 2010) for the different move acceptance criteria are presented for each of the 13 instances of the Toronto benchmark problems. Each algorithm finishes after one hour of computation and is executed 10 times. For every algorithm, which is identified by the move acceptance criterion and a

Table 5 Average and minimum fitness and standard deviation for each of the Toronto instances after 10 runs. The acceptance list length for both acceptance strategies is 500 for all experiments. The stopping criterion is one hour

Tournament	4														
Acceptance	SA			LA			LA-SD			GD			IE		
	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV
car91	5.13	5.07	0.05	5.69	5.62	0.05	***5.12***	***5.04***	0.05	5.22	5.1	0.08	5.56	5.47	0.07
car92	***4.06***	***3.99***	0.05	4.39	4.35	0.03	4.17	4.04	0.06	4.16	4.09	0.05	4.6	4.45	0.11
ear83	***33.1***	***32.76***	0.2	33.61	32.99	0.41	35.05	34.39	0.64	33.55	33.06	0.32	38.25	35.44	1.31
hec92	***10.2***	10.09	0.13	10.49	10.27	0.16	10.79	10.45	0.2	10.28	***10.06***	0.14	11.91	11.46	0.34
kfu93	13.51	***13.27***	0.17	***13.47***	13.29	0.14	13.47	13.28	0.08	13.54	13.38	0.13	14.67	14.1	0.34
lse91	***10.49***	***10.21***	0.19	10.75	10.43	0.25	10.85	10.58	0.17	10.59	10.45	0.16	12.11	11.79	0.25
pur93	7.22	7.05	0.1	8.33	8.21	0.06	8.19	8.15	0.03	6.96	***6.79***	0.09	***6.9***	6.8	0.06
rye92	8.61	8.5	0.09	9.27	9.16	0.09	***8.56***	***8.43***	0.07	8.73	8.5	0.1	9.28	8.89	0.19
sta83	***157.05***	***157.03***	0.01	157.1	***157.03***	0.06	157.12	157.05	0.05	157.19	***157.03***	0.15	157.3	157.05	0.25
tre92	***7.95***	***7.82***	0.09	8.24	8.16	0.07	8.45	8.33	0.07	8.22	8.04	0.11	9.14	9.02	0.11
uta92	***3.46***	3.4	0.04	3.72	3.64	0.04	3.49	3.42	0.05	3.48	***3.38***	0.05	3.78	3.7	0.05
ute92	***24.89***	***24.82***	0.07	25.04	24.82	0.31	25.15	24.88	0.23	25.07	24.92	0.13	26.37	25.82	0.5
yor83	***35.21***	***34.7***	0.27	36.23	35.4	0.62	36.96	36.02	0.51	36.33	35.86	0.31	39.58	37.69	1.2

Table 6 Average and minimum fitness and standard deviation for each of the Toronto instances. The acceptance list length is for both acceptance strategies 500 for all experiments. The stopping criterion is 1 hour

Tournament	8														
Acceptance	SA			LA			LA-SD			GD			IE		
	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV	AVG	MIN	STDEV
car91	***5.19***	5.13	0.05	5.59	5.52	0.04	5.27	5.19	0.05	5.28	***5.03***	0.1	5.6	5.45	0.09
car92	4.2	4.09	0.08	4.44	4.33	0.07	***4.16***	***4.06***	0.07	4.32	4.23	0.05	4.71	4.55	0.12
ear83	***33.12***	***32.75***	0.45	34.68	33.26	1.2	34.87	33.4	1.04	33.52	33.02	0.41	37.49	35.3	1.16
hec92	***10.2***	***10.03***	0.16	10.53	10.23	0.16	10.72	10.55	0.11	10.38	10.14	0.21	11.63	11.17	0.32
kfu93	13.84	13.42	0.36	13.7	13.33	0.17	13.68	13.43	0.19	***13.6***	***13.31***	0.22	14.41	13.86	0.38
lse91	10.87	10.63	0.25	10.89	10.32	0.31	10.9	10.57	0.19	10.65	10.44	0.18	12.2	11.61	0.3
pur93	7.2	7	0.09	8.12	8.07	0.04	8.02	7.96	0.04	7.04	6.94	0.06	***6.93***	***6.8***	0.08
rye92	8.62	8.47	0.09	9.11	8.9	0.13	***8.61***	8.5	0.08	8.72	***8.46***	0.13	9.33	9.06	0.2
sta83	***157.04***	**157.03**	0	157.1	157.05	0.06	157.16	157.05	0.11	157.14	**157.03**	0.1	157.26	**157.03**	0.2
tre92	***8.04***	***7.93***	0.08	8.24	8.13	0.1	8.47	8.25	0.14	8.26	8.08	0.14	9.09	8.81	0.24
uta92	3.51	3.46	0.04	3.7	3.64	0.05	***3.5***	***3.45***	0.05	3.5	3.46	0.03	3.84	3.73	0.07
ute92	25.07	24.87	0.24	***25.04***	***24.83***	0.16	25.23	25.04	0.18	25.18	25.02	0.15	26.72	25.95	0.53
yor83	***35.48***	***35.04***	0.52	35.62	35.31	0.36	36.7	35.97	0.52	36.28	35.38	0.53	39.44	38.38	0.71

Table 7 Statistically significant best performing algorithms and corresponding p-values for Table 5 and 6

	Statistically significant best performing algorithm	Corresponding p-values
car91	LA-SD-4, SA-4	1, 0.46
car92	SA-4	1
ear83	SA-4, SA-8	1, 0.88
hec92	SA-4, SA-8, GD-4	1, 0.92, 0.18
kfu93	LA-SD-4, LA-4, SA-4, GD-4, GD-8	1, 0.95, 0.54, 0.16, 0.12
lse91	SA-4, GD-4, GD-8	1, 0.24, 0.08
pur93	IE-4, IE-8, GD-4	1, 0.36, 0.11
rye92	LA-SD-4, SA-4, LA-SD-8, SA-8	1, 0.15, 0.14, 0.16
sta83	SA-8	1
tre92	SA-4	1
uta92	SA-4, GD-4, LA-SD-4	1, 0.32, 0.23
ute92	SA-4, LA-4	1, 0.16
yor83	SA-4, SA-8	1, 0.17

tournament factor, we present the average, the minimum fitness, and the standard deviation. The best fitness for every instance is indicated in italic.

It is important to mention that the parameter setting of the hyperheuristic approach is the same for all the data instances. For instance, the same SA-4 algorithm is applied to every data instance.

The algorithms that performed best in a statistically significant manner are indicated in Table 7. In the last column of Table 7, the corresponding p-value is marked. A p-value of 1 means that the first algorithm in the second column performs best according to the average fitness value. The other p-values are calculated against the best performing algorithm. In general, the best performing move acceptance criterion—tournament factor combination is SA-4. Only for data instances `pur93` and `sta83`, other move acceptance criteria perform better.

In Table 8, we present the results of experiments with longer execution times (ranging from 1 up to 12 hours) of the SA-4 hyperheuristic. In all cases longer running times lead to better results.

In Table 9, we compare the hyperheuristics results with the Toronto benchmark results in the literature. We limit the selected results from the literature to those that have been confirmed accurate by Qu et al. (2009). For seven out of the 13 data instances, we obtain better or equal results.

From Table 9, we can conclude that for the smaller instances, longer running times lead to better or equal quality results compared to the results from the literature. However, for the `rye92` and `pur93` data instances, these results are considerable worse than the best results from literature.

Compared to Kendall and Hussin (2005a), who deploy a more elaborate heuristics selection method and use running times up to four hours, we obtain better results in general,

Table 8 Average and minimum fitness and standard deviation for the SA-4 algorithm

	Stopping criterion	Average	Minimum	Standard deviation
car91	4 h	4.75	4.68	0.05
	12 h	4.64	4.52	0.05
car92	4 h	3.94	3.84	0.05
	12 h	3.86	3.78	0.06
ear83	2 h	33.02	32.82	0.16
	12 h	32.69	32.49	0.13
hec92	1 h	10.2	10.09	0.13
	12 h	10.06	10.03	0.03
kfu93	2 h	13.45	13.06	0.31
	12 h	13.24	12.90	0.2
lse91	2 h	10.38	10.06	0.19
	12 h	10.21	10.04	0.13
pur93	4 h	6.57	6.45	0.07
	12 h	5.75	5.67	0.05
rye92	4 h	8.31	8.18	0.1
	12 h	8.2	8.05	0.12
sta83	1 h	157.05	157.03	0.01
tre92	2 h	7.91	7.73	0.06
	12 h	7.79	7.69	0.07
uta92	2 h	3.37	3.32	0.03
	12 h	3.17	3.13	0.03
ute92	2 h	24.99	24.83	0.24
	12 h	24.88	24.77	0.17
yor83	2 h	35.06	34.79	0.25
	12 h	34.83	34.64	0.14

even after one hour of computation. Possibly, this is partly attributed to the use of other heuristics and different move acceptance criteria.

In Table 10 both late acceptance strategies are compared with Student t-tests. For the larger data instances (`car91`, `car92`, `pur93`, `rye92` and `uta92`), the adapted version of the late acceptance strategy method performs better than the original late acceptance strategy in a statistically significant way. On the other hand, the original late acceptance strategy performs significantly better for the following data instances: `ear83` `hec92`, `tre92`, `ute92` and `yor83`. For the three remaining data instances `kfu93`, `lse91` and `sta83`, there is no statistical significance in the difference between the performance of the two late acceptance strategies.

5.4 The ITC 2007 Examination Timetabling Problem

Starting from the initial (not necessarily feasible) solution, the hyperheuristic for the ITC 2007 problem first tries to

Table 9 Selection of the best results from the literature compared with the best obtained values from the SA-4 hyperheuristics approach (12 hours of computation time)

Instance	SA-4	(Carter et al. 1996)	(Burke and Newall 2003)	(Merlot et al. 2003)	(Burke and Newall 2004)	(Burke et al. 2004a)	(Kendall and Hussin 2005b)	(Yang and Petrovic 2005)	(Eley 2007)	(Burke and Bykov 2008)	(Burke et al. 2010)
car91	4.52	7.10	4.65	5.10	5.00	4.80	5.37	**4.50**	5.20	4.58	4.9
car92	**3.78**	6.20	4.10	4.30	4.30	4.20	4.67	3.93	4.30	3.81	4.1
ear83	**32.49**	36.40	37.05	35.10	36.20	35.40	40.18	33.71	36.80	32.65	33.2
hec92	**10.03**	10.80	11.54	10.60	11.60	10.80	11.86	10.83	11.10	10.06	10.3
kfu93	12.90	14.00	13.90	13.50	15.00	13.70	15.84	13.82	14.50	**12.81**	13.2
lse91	10.04	10.50	10.82	10.50	11.00	10.40	–	10.35	11.30	**9.86**	10.4
pur93	5.67	**3.90**	–	–	–	4.80	–	–	–	4.32	–
rye92	8.05	**7.30**	–	8.40	–	8.90	–	8.53	9.80	7.93	–
sta83	157.03	161.50	168.73	157.30	161.90	159.10	157.38	158.35	157.30	157.03	**156.9**
tre92	**7.69**	9.60	8.35	8.40	8.40	8.30	8.39	7.92	8.60	7.72	8.3
uta92	**3.13**	3.50	3.20	3.50	3.40	3.40	–	3.14	3.50	3.16	3.3
ute92	**24.77**	25.80	25.83	25.10	27.40	25.70	27.60	25.39	26.40	27.79	24.9
yor83	**34.64**	41.70	37.28	37.40	40.80	36.70	–	36.35	39.40	34.78	36.3

Table 10 An X indicates which method performs better in a statistically significant manner. The comparisons are conducted on the results of Table 5 and 6

	4		8	
	LA-SD	LA	LA-SD	LA
car91	X		X	
car92	X		X	
ear83		X		
hec92		X		X
kfu93				
lse91				
pur93	X		X	
rye92	X		X	
sta83				
tre92		X		X
uta92	X		X	
ute92				X
yor83		X		

find a feasible solution, and secondly a feasible solution that satisfies the soft constraints as much as possible. The first part of the search for a feasible solution is further divided into three stages:

- In a first stage, starting from the constructed initial solution, the algorithm tries to obtain a solution that satisfies the hard constraints concerning the room occupancy, the period utilization and room related issues (some of the exams can only be assigned to rooms that do not hold any other exams). The period utilization and room occupancy constraints are already satisfied due to the construction of the initial solution.
- In a second stage, which starts from a solution that satisfies the three hard constraints discussed above, the evaluation of the period related constraint is added to the fitness function. This stage stops when a solution is found that satisfies all four constraints.
- In the final stage, the last hard constraint (two conflicting exams cannot be assigned to the same time slot) is added to the fitness function. The search finishes when a feasible solution is found.

Once a feasible solution is found, the second part of the search begins: the soft constraints are added to the fitness function and the search starts from the feasible solution obtained in the first part. In order to avoid violations of the hard constraints, the weights corresponding to the hard constraints in the fitness function are increased to 1000, whilst the weights of the soft constraints get the value 1.

Similarly to the Toronto approach, the ITC 2007 data sets are solved in two phases. In both the feasibility and the improvement phase, we have experimented with hyperheuristics with different move acceptance criteria (see Table 11). Solving the problem in phases is based on the good

Table 11 Average, minimum, standard deviation and the number of feasible solutions for the 12 data sets obtained by eleven hyperheuristics

	SR-4-HH1				SR-4-HH2				SR-4-HH3			
	AVG	MIN	STD	#F	AVG	MIN	STD	#F	AVG	MIN	STD	#F
1	6471.6	6262	171.57	10	6314.8	***5965***	166.03	10	6451.1	6321	112.64	10
2	550	510	39.21	8	596.3	510	65.69	10	569.67	505	44.43	10
3	26358.67	19104	6285.19	3	–	–	–	0	–	–	–	0
4	–	–	–	0	–	–	–	0	–	–	–	0
5	5446.14	5006	322.02	7	5165.63	***4820***	157.92	8	5360.6	5021	194.89	10
6	29388.33	27490	2628.45	6	28641.43	27880	475.62	7	28279.29	***27445***	512.38	7
7	6276.8	6033	170.82	10	6185.89	***5905***	179.15	9	6315	6079	291.4	10
8	9379.9	9199	155.51	10	9362.8	9089	173.78	10	9507.7	9220	231.61	10
9	1292.8	1211	65.15	10	1315.2	1213	57.01	10	1335.7	1217	72.45	10
10	15945.33	15587	305.64	6	15565.3	15204	244.04	10	15638	15425	157.9	9
11	–	–	–	0	–	–	–	0	–	–	–	0
12	5876.29	5527	257.81	7	5974.22	5679	267.64	9	5839.29	5580	147.29	7

	SR-4-HH4				SR-4-HH5				SR-4-HH6			
	AVG	MIN	STD	#F	AVG	MIN	STD	#F	AVG	MIN	STD	#F
1	6330.2	6060	162.69	10	6490.1	6338	125.31	10	6466.6	6146	162.76	10
2	612.5	515	99.65	10	630.11	***498***	128.64	9	638.88	549	81.94	8
3	23580	23580	0	1	–	–	–	0				
4	–	–	–	0	–	–	–	0	–	–	–	0
5	5323	4855	350.5	8	5397	4986	250.75	6	5485.56	5054	283.47	9
6	28578.13	27605	698.92	8	29146	28590	413.12	5	28922.86	28435	809.18	7
7	6250	6065	168.38	10	6296.8	6039	200.91	10	6350.9	6133	173.56	10
8	9260.9	9038	184.16	10	9408.5	9195	169.31	10	9439.1	9146	182.59	10
9	1255.9	1184	63.63	10	1291.7	1236	34.33	10	1312.5	1203	45.1	10
10	16113.33	15561	464.63	9	15843.4	15594	187.58	5	17072.38	15201	3792.23	8
11	–	–	–	0	–	–	–	0	–	–	–	0
12	5829.14	5483	176.21	7	5865.38	5525	236.87	8	5738.56	***5409***	219.77	9

	SR-4-HH7				SR-4-HH8				SR-4-HH9			
	AVG	MIN	STD	#F	AVG	MIN	STD	#F	AVG	MIN	STD	#F
1	6286.1	6170	102.69	10	6591.3	6243	338.88	10	6667.6	6437	178.55	10
2	580.1	520	45.37	10	590.8	508	57.44	10	773.3	603	77.57	10
3	22966.5	20806	1866.77	4	24206.33	21449	2183.45	6	21437.33	20485	1207.16	3
4	–	–	–	0	–	–	–	0	–	–	–	0
5	6168.2	5391	470.57	5	6839	6480	370.24	4	7790	7639	133.63	3
6	28468.89	27890	525.91	9	28567.5	27665	660.17	6	28000	27530	556.84	3
7	6458	5980	377.79	8	6665.6	6223	475.44	10	7028.7	6658	222.86	10
8	9422.22	9073	193.39	9	9396.3	9011	280.3	10	10220.11	9902	178.35	9
9	1384.4	1272	52.74	9	1320.2	1224	55.95	10	1244	1196	32.84	10
10	15486.9	***15023***	325.5	10	15811.7	15360	304.88	10	15614.78	15079	421.74	9
11	–	–	–	0	–	–	–	0	–	–	–	0
12	5896.63	5449	266.28	8	5944.14	5630	184.59	7	5733	5444	254.88	5

Table 11 (*Continued*)

	SR-4-HH10				SR-4-HH11			
	AVG	MIN	STD	#F	AVG	MIN	STD	#F
1	6809.6	6428	199.65	10	6404.2	6260	126.18	10
2	873.22	689	278.45	9	576.1	526	57.04	10
3	–	–	–	0	20810.4	***18455***	1596.6	10
4	–	–	–	0	–	–	–	0
5	6439	6439	–	1	8354.86	5378	5369.49	7
6	28893.75	27955	1028.27	4	28801.5	27655	926.55	10
7	7004.3	6776	175.23	10	6390.9	5995	229.43	10
8	10211.8	9790	229.39	10	9257.4	***8949***	149.52	10
9	1281.1	***1170***	62	10	1301.6	1202	65.28	10
10	–	–	–	0	15648.9	15145	329.27	10
11	–	–	–	0	–	–	–	0
12	5534	5469	66.89	4	5875.13	5411	372.27	8

Table 12 Description of the 11 move acceptance criteria. The hyperheuristics differ in the move acceptance criteria used in the constructive and the improvement phase and in the available computation time in these two phases

Abbrevations	Description
SR-4-HH1	feasible stage: IE; improvement stage: IE
SR-4-HH2	feasible stage: SA (until 50% computation time); improvement + feasible stage: IE
SR-4-HH3	feasible stage: SA; improvement stage: IE
SR-4-HH4	feasible stage: LA (until 50% computation time); improvement + feasible stage: SA
SR-4-HH5	feasible stage: LA; improvement stage: SA
SR-4-HH6	feasible stage: LA-SA (until 50% computation time); improvement + feasible stage: SA
SR-4-HH7	feasible stage: LA-SD (until 50% computation time); improvement + feasible stage: LA-SD
SR-4-HH8	feasible stage: LA-SD; improvement stage: LA-SD
SR-4-HH9	feasible stage: LA (until 50% computation time); improvement + feasible stage: GD
SR-4-HH10	feasible stage: LA; improvement stage: GD
SR-4-HH11	feasible stage: IE; improvement stage: SA

Table 13 The Student t-test p-values for the different hyperheuristics. The algorithms are compared to the best performing algorithm (indicated with value 1). Algorithms that are not significantly worse are indicated in bold

Instances	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	HH9	HH10	HH11
1	0.01	**0.63**	0	**0.48**	0	0.01	**1**	0.01	0	0	0.03
2	**1**	**0.1**	**0.35**	**0.12**	**0.11**	0.02	**0.16**	**0.11**	0	0.01	**0.29**
3	**0.18**	–	–	–	–	–	**0.09**	**0.08**	**0.32**	–	**1**
5	**0.05**	**1**	0.04	**0.27**	**0.06**	0.01	0	0	0	–	**0.12**
6	**0.41**	**0.1**	**0.46**	**0.23**	0.02	**0.11**	**0.22**	**0.24**	**1**	**0.24**	**0.19**
7	**0.27**	**1**	**0.27**	**0.43**	**0.22**	**0.06**	**0.07**	0.01	0	0	**0.05**
8	**0.09**	**0.16**	0.01	**0.96**	**0.05**	0.03	**0.05**	**0.18**	0	0	**1**
9	**0.05**	0	0	**0.61**	0.01	0	0	0	**1**	**0.11**	0.02
10	0.01	**0.55**	**0.22**	0	0.04	**0.2**	**1**	0.03	**0.47**	–	**0.28**
12	0.03	0.01	0	0.01	0.02	0.1	0.03	0	**0.18**	**1**	**0.11**

results obtained by combining different methods in the ITC 2007 literature (Gogos et al. to appear; Müller 2008). Based on the good results obtained with tournament factor 4 in Sects. 5.2 and 5.3, we focus on the same tournament factor value. An overview of the eleven hyperheuristics is presented in Table 12. Some of the hyperheuristics only differ in the amount of computation time that is spent in finding a feasible solution. Hyperheuristics SR-4-HH2, SR-4-HH4, SR-4-HH6, SR-4-HH7, SR-4-HH9 can spend at most 150 seconds (this is 50% of the available computation time) in the first (feasibility) stage. If no feasible solution is found in this period, the algorithm automatically switches to the improvement phase, whilst at the same time still trying to make the solution feasible. The other hyperheuristics only switch

to the improvement phase when a feasible solution is found. From the obtained results and the t-tests (Table 13), there is no hyperheuristic that significantly performs better than the others over all instances. In order to find the best performing hyperheuristic to all instances, we have ranked the algorithms according to their average and best performance (Table 14). The algorithm that finds the best solution for instance, gets the lowest value, whilst the worst performing algorithm gets a high value. We have also applied the same ranking mechanism to the average values obtained after 10 runs for each instance. The final overall ranking is based on the average of the rankings for all instances. The algorithm with the overall lowest rank can be considered the best performing algorithm. If we only consider the average fitness values for each instance and for each algorithm, we find that the best performing algorithm is SR-4-HH4. On the other hand, if we consider the minimum fitness values, we find that algorithm SR-4-HH11 is the best performing algorithm. The hyperheuristic variants have simulated annealing as the last move acceptance criterion in common. It is safe to say that simulated annealing is a good quality method for these experiments. Hyperheuristic SR-4-HH11 finds most feasible solutions: 95 out of 120 runs (Table 11), whilst SR-4-HH2 finds the best solutions. For two data sets (instance 4 and instance 11), we do not find any feasible solution. Although we do not improve on the best solutions of the ITC 2007 literature, our solutions are competitive with the finalists' results (Table 15).

Table 14 Ranking of the hyperheuristic algorithms according to their average and best performance. The lower the rank, the better the algorithm's performance

Minimum		Average	
Algorithm order	Rank	Algorithm order	Rank
SR-4-HH11	4.0	SR-4-HH4	4.0
SR-4-HH4	4.5	SR-4-HH2	4.9
SR-4-HH2	5.2	SR-4-HH11	5.2
SR-4-HH7	5.3	SR-4-HH3	5.4
SR-4-HH1	5.4	SR-4-HH7	5.4
SR-4-HH6	6.1	SR-4-HH1	5.8
SR-4-HH3	6.5	SR-4-HH9	6.1
SR-4-HH8	6.5	SR-4-HH5	6.6
SR-4-HH9	6.7	SR-4-HH8	7.1
SR-4-HH5	7.2	SR-4-HH6	7.3
SR-4-HH10	8.6	SR-4-HH10	8.2

Our results are in line with the experiments of Bilgin et al. (2007). These authors have conducted numerous experiments with different combinations of move acceptance criteria and several heuristic selection methods on different benchmark instances. Their conclusion was that no single combination of acceptance criteria and selection methods dominates any other combination on all benchmarks. In their experiments, 'improving or equal' resulted in the best average performance, whilst 'choice function' was on average slightly better than the other heuristic selection mechanisms. The choice function heuristic selection method has some built in memory on how well previous applied heuristics performed. The better they have performed, the higher their selection probability at the next iterations. It is very hard to explain why one move acceptance criterion performs better than another one on a particular data instance. In future research, we will take the performance of the different heuristics into account, in a heuristic selection mechanism that has a built in learning mechanism.

We would like to conclude this section with a general advice. The competition organizers released a validation tool

Table 15 Comparison of the solutions obtained with the best performing algorithm on average (SR-4-HH4) with the best solutions from the literature. The best solutions are indicated in bold and italics. Note that (Gogos et al. to appear) do not report on the 4 last instances

Instance	(Müller 2008)	(Gogos et al. 2008)	(Atsuta et al. 2008)	(De Smet 2008)	(Pillay 2008)	(McCollum et al. 2009)	(Gogos et al. to appear)	SR-4-HH4
1	***4370***	5905	8006	6670	12035	4633	4775	6060
2	400	1008	3470	623	3074	405	***385***	515
3	10049	13862	18622	–	15917	9064	***8996***	23580
4	18141	18674	22559	–	23582	***15663***	16204	–
5	2988	4139	4714	3847	6860	3042	***2929***	4855
6	26950	27640	29155	27815	32250	25880	***25740***	27605
7	4213	6683	10473	5420	17666	***4037***	4087	6065
8	7861	10521	14317	–	16184	***7461***	7777	9038
9	***1047***	1159	1737	1288	2055	1071	–	1184
10	16682	–	15085	14778	17724	***14374***	–	15561
11	34129	43888	–	–	40535	***29180***	–	–
12	5535	–	***5264***	–	6310	5693	–	5483

for verifying the obtained solution in order to guarantee correctness. Initiatives like this are highly appreciated: foreseeing a validation tool overcomes problems with changing data sets and wrong objective functions as reported in Qu et al. (2009). Also, the CPU performance tool that was released for the International Timetabling Competition allows a fair comparison among different algorithms, since the resulting computation time will be inversely proportional to the performance of the hardware specifications of the researcher's computer. Some of the papers discussing the Toronto benchmarks give computation times, whereas others do not. Even if computation times are presented, it is still a hard task to compare the approaches, since they were executed on different platforms.

6 Conclusions and future work

We have proposed a hyperheuristic approach to examination timetabling problems that successfully tackles a curriculum-based problem from practice as well as two post-enrolment benchmark instances from the literature.

For the examination timetabling problem from practice, the hyperheuristic generates better solutions, in terms of the number of time slots, than those obtained by the manual planner. Moreover, due to the problem size and its complexity, it takes the manual planner a few weeks to generate a timetable. In addition, the hyperheuristic succeeds to strongly improve on the manual schedule in about one hour on an average desktop computer.

We improved on seven out of thirteen data instances from the Toronto benchmarks. Although the improvement of the benchmark instances was not our primary concern, it shows that focusing on problems from practice can result in competitive algorithms. Concerning the ITC 2007 data sets, our results are competitive, although we could not improve on the best results in the literature.

Since the problems are not completely equivalent, different low-level heuristics had to be applied. Note that these low-level heuristics are rather specific: spreading out exams with the $Uncap_i$ heuristics, or moving exams into an appropriate room with the Cap_i heuristics. The $Uncap_i$ heuristics could not beat the Cap_i heuristics for the KAHO problem, neither were the Cap_i heuristics able to solve the Toronto benchmarks data instances.

We used an extremely simple heuristic selection method, i.e. one that randomly selects a heuristic from a fixed list of heuristics. It obtains in general better results than some of the approaches that employ more 'intelligent' heuristic selection methods. This raises the question whether further improvement can be obtained by merely looking at more sophisticated heuristic selection methods or by concentrating on smarter low-level heuristics. There may be a trade-off in that smarter heuristics are required to reveal the power of a more intelligent heuristic selection mechanism.

Also, we have applied the late acceptance strategy, and a variant, as a move acceptance criterion in the hyper heuristics framework. For the KAHO problem the late acceptance variant combined with steepest descent performs as good as the simulated annealing move acceptance criterion.

Acknowledgements The authors wish to thank the anonymous reviewers for their useful recommendations and comments.

References

Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M. (2007a). Investigating Abuja-Orlins large neighbourhood search for examination timetabling. *OR-Spektrum*, *29*(2), 351–372.

Abdullah, S., Ahmadi, S., Burke, E. K., Dror, M., & McCollum, B. (2007b). A tabu based large neighbourhood search methodology for the capacitated examination timetabling problem. *Operations Research*, *58*, 1494–1502.

Atsuta, M., Nonobe, K., & Ibaraki, T. (2008). *Itc2007 track 1: An approach using a general csp solver* (Technical report).

Ayob, M., Malik, A. M. A., Abdullah, S., Hamdan, A. R., Kendall, G., & Qu, R. (2007). Solving a practical examination timetabling problem: A case study. In O. Gervasi & M. Gavrilova (Eds.), *LNCS: Vol. 4707. Computational science and its applications ICCSA 2007* (pp. 611–624). Berlin: Springer.

Bai, R., & Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In *Operations research/computer science interfaces: Vol. 32. Metaheuristics: Progress as real problem solvers* (pp. 87–108). Berlin: Springer.

Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., & McCollum, B. (2007). *A simulated annealing hyper-heuristic methodology for flexible decision support* (Technical Report NOTTCS-TR-2007-8). School of Computer Science, University of Nottingham.

Beligiannis, G. N., Moschopoulos, C. N., Kaperonis, G. P., & Likothanassis, S. D. (2008). Applying evolutionary computation to the school timetabling problem: The Greek case. *Computers & Operations Research*, *35*(4), 1265–1280.

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam timetabling. In E. K. Burke & H. Rudová (Eds.), *LNCS: Vol. 3867. Revised selected papers of the 6th international conference on practice and theory of automated timetabling (PATAT 2006)*, August/September 2007 (pp. 394–412). Berlin: Springer.

Bilgin, B., Demeester, P., Mısır, M., Vancroonenburg, W., & Vanden Berghe, G. (2010). *One hyperheuristic approach to two timetabling problems in health care* (Technical report). KaHo Sint-Lieven.

Burke, E. K., & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In E. K. Burke & M. Gendreau (Eds.), *Proceedings of the 7th international conference on the practice and theory of automated timetabling*, Montréal, Canada, August 2008.

Burke, E. K., & Newall, J. (2003). Enhancing timetable solutions with local search methods. In E. K. Burke & P. De Causmaecker (Eds.), *LNCS: Vol. 2740. Proceedings of the 4th international conference on practice and theory of automated timetabling (PATAT 2002)* (pp. 195–206). Berlin: Springer.

Burke, E. K., & Newall, J. (2004). Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, *129*, 107–134.

Burke, E. K., Newall, J. P., & Weare, R. F. (1996). A memetic algorithm for university exam timetabling. In E. K. Burke & P. Ross (Eds.), *LNCS: Vol. 1153. Selected papers of first international conference on practice and theory of automated timetabling*, Edinburgh, UK, August/September (pp. 241–250). Berlin: Springer.

Burke, E. K., Hart, E., Kendall, G., Ross, J., & Schulenburg, S. (2003). Hyperheuristics: an emerging direction in modern search technology. In *Handbook of metaheuristics* (pp. 457–474). Berlin: Springer.

Burke, E. K., Bykov, Y., Newall, J., & Petrovic, S. (2004a). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, *36*(6), 509–528.

Burke, E. K., de Werra, D., & Kingston, J. (2004b). Applications to timetabling. In *Handbook of graph theory* (pp. 445–474). New York: Chapman Hall/CRC Press.

Burke, E. K., Dror, M., Petrovic, S., & Qu, R. (2005). Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In B. L. Golden, S. Raghavan, & E. A. Wasil (Eds.), *Conference volume of the 9th informs computing society conference* (pp. 79–91). Berlin: Springer.

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

Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, *206*(1), 46–53.

Caramia, M., Dell'Olmo, P., & Italiano, G. F. (2001). New algorithms for examination timetabling. In *LNCS: Vol. 1982. Proceedings of the 4th international workshop on algorithm engineering* (pp. 230–242). Berlin: Springer.

Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. In E. K. Burke & P. Ross (Eds.), *LNCS: Vol. 1153. Practice and theory of automated timetabling I: selected papers from the 1st international conference* (pp. 3–21). Berlin: Springer.

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

Cole, A. J. (1964). The preparation of examination timetables using a small store computer. *Computer Journal*, *7*, 117–121.

Corne, D., Ross, P., & Fang, H. (1994). Evolutionary computing. In T. C. Fogarty (Ed.), *LNCS: Vol. 865. Fast practical evolutionary timetabling* (pp. 250–263). Berlin: Springer.

Dammak, A., Elloumi, A., Kamoun, H., & Ferland, J. A. (2008). Course timetabling at a Tunisian university: A case study. *Journal of Systems Science and Systems Engineering*, *17*(3).

De Causmaecker, P., Demeester, P., & Vanden Berghe, G. (2009). A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research*, *195*(1), 307–318.

De Smet, G. (2008). *ITC2007—examination track: Practice and theory of automated timetabling* (Technical report).

Demeester, P., Souffriau, W., De Causmaecker, P., & Vanden Berghe, G. (2010). A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine*, *48*(1), 61–70.

Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), *LNCS: Vol. 2079. Practice and theory of automated timetabling III: selected papers from the third international conference* (pp. 104–117). Berlin: Springer.

Di Gaspero, L., & Schaerf, A. (2008). Hybrid local search techniques for the generalized balanced academic curriculum problem. In M. J. Blesa Aguilera, C. Blum, C. Cotta, A. J. Fernndez, J. E. Gallardo, A. Roli, & M. Sampels (Eds.), *LNCS: Vol. 5296. Proceedings of hybrid metaheuristics 5th international workshop (HM 2008)*, Malaga, Spain, October 8–9, 2008 (pp. 146–157). Berlin: Springer.

Dimopoulou, M., & Miliotis, P. (2001). Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, *130*, 202–213.

Eley, M. (2007). Ant algorithms for the exam timetabling problem. In E. K. Burke & H. Rudová (Eds.), *LNCS: Vol. 3867. Practice and theory of automated timetabling (Patat 2006)* (pp. 364–382). Berlin: Springer.

First International Nurse Rostering Competition (2010). https://www.kuleuven-kortrijk.be/nrpcompetition.

Gogos, C., Alefragis, P., & Housos, E. (2008). A multi-staged algorithmic process for the solution of the examination timetabling problem. In E. K. Burke & M. Gendreau (Eds.), *Practice and theory of automated timetabling (PATAT)*, Montréal, Canada, 19–22 August 2008.

Gogos, C., Alefragis, P., & Housos, E. (to appear). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*. doi:10.1007/s10479-010-0712-3.

Hansen, M. P., & Vidal, R. V. V. (1995). Planning of high school examinations in Denmark. *European Journal of Operational Research*, *87*, 519–534.

Kendall, G., & Hussin, N. M. (2005a). An investigation of a tabu-search-based hyper-heuristic for examination timetabling. In G. Kendall, E. K. Burke, S. Petrovic, & M. Gendreau (Eds.), *Multidisciplinary scheduling: theory and applications. Selected papers of the first international conference (MISTA)* (pp. 309–328). Berlin: Springer.

Kendall, G., & Hussin, N. M. (2005b). A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. In E. K. Burke & M. Trick (Eds.), *LNCS: Vol. 3616. Practice and theory of automated timetabling (Patat 2004)* (pp. 270–293). Berlin: Springer.

Kostuch, P. (2005). The university course timetabling problem with a three-phase approach. In E. K. Burke & M. Trick (Eds.), *LNCS: Vol. 3616. Practice and theory of automated timetabling (Patat 2004)* (pp. 109–125). Berlin: Springer.

Lim, A., Chin, A. J., Kit, H. W., & Oon, W. C. (2000). A campuswide university examination timetabling application. In *Proceedings of the 17th national conference on artificial intelligence and 12th conference on innovative applications of artificial intelligence* (pp. 1020–1025).

McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., & Qu, R. (2007). *The second international timetabling competition: Examination timetabling track* (Technical Report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17). Queen's University, Belfast, September.

McCollum, B., McMullan, P. J., Parkes, A. J., Burke, E. K., & Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. In *The 4th multidisciplinary international conference on scheduling: Theory and applications (Mista 09)*, Dublin, Ireland, August 2009 (pp. 424–434).

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

Merlot, L. T. G., Boland, N., Hughes, B. D., & Stuckey, P. J. (2003). A hybrid algorithm for the examination timetabling problem. In E. K. Burke & P. De Causmaecker (Eds.), *LNCS: Vol. 2740. Practice and theory of automated timetabling IV* (pp. 207–231). Berlin: Springer.

Müller, T. (2008). ITC2007 solver description: A hybrid approach. In E. K. Burke & M. Gendreau (Eds.), *Proceedings of the 7th international conference on the practice and theory of automated timetabling*, Montréal, Canada, August 2008.

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, *12*(1), 3–23.

Pillay, N. (2008). *A developmental approach to the examination timetabling problem* (Technical report).

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

Qu, R. (2010). Benchmark data sets in exam timetabling. http://www.cs.nott.ac.uk/~rxq/data.htm.

Qu, R., & Burke, E. K. (2005). Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems. In *MIC 2005: The sixth metaheuristics international conference*, Vienna, Austria.

Qu, R., & Burke, E. K. (2009). Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. *The Journal of the Operational Research Society*, *60*, 1273–1285.

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

Ross, P., Corne, D., & Fang, H. (1994). Improving evolutionary timetabling with delta evaluation and directed mutation. In Y. Davidor, H. Schwefel, & R. Männer (Eds.), *LNCS: Vol. 866. Parallel problem solving from nature—PPSN III international conference on evolutionary computation. The third conference on parallel problem solving from nature*, Jerusalem, Israel, October 9–14 (pp. 556–565). Berlin: Springer.

Schaerf, A. (1999a). Local search techniques for large high-school timetabling problems. *IEEE Transactions on Systems, Man and Cybernetics. Part A. Systems and Humans*, *29*(4), 368–377.

Schaerf, A. (1999b). A survey of automated timetabling. *Artificial Intelligence Review*, *13*(2), 87–127.

Schaerf, A., & Di Gaspero, L. (2007). Measurability and reproducibility in university timetabling research: Discussion and proposals. In E. K. Burke & H. Rudová (Eds.), *LNCS: Vol. 3867. Revised selected papers of the sixth international conference on practice and theory of automated timetabling (Patat 2006)*, Brno, Czech Republic, August/September 2007 (pp. 40–49). Berlin: Springer.

Thompson, J. M., & Dowsland, K. A. (1996). Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, *63*, 105–128.

Thompson, J. M., & Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, *25*, 637–648.

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

Yang, Y., & Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. In E. K. Burke & M. Trick (Eds.), *LNCS: Vol. 3616. Practice and theory of automated timetabling V: selected papers from the fifth international conference*, August 2005 (pp. 377–396). Berlin: Springer.