Journal of Scheduling
https://doi.org/10.1007/s10951-023-00799-1

Multi-neighbourhood simulated annealing for the ITC-2007 capacitated examination timetabling problem

David Van Bulck[1,2] · Dries Goossens[1,2] · Andrea Schaerf[3]

Accepted: 31 October 2023

Abstract
We propose a multi-neighbourhood simulated annealing algorithm for the ITC-2007 version of the capacitated examination timetabling problem. The proposed solver is based on a combination of existing as well as newly proposed neighbourhoods that better exploit the disconnected structure of the underlying conflict graph and that explicitly deal with the assignment of exams to rooms. We use a principled tuning procedure to determine the parameters of the algorithm and assess the contribution of the various neighbourhoods by means of an ablation analysis. The resulting algorithm is able to compete with existing state-of-the-art solvers and finds several new best solutions for a variety of well-known problem instances.

Keywords Capacitated examination timetabling · Room assignment · Simulated annealing · ITC-2007

1 Introduction

Examination timetabling (ETT) is a practical problem that every university faces regularly. Each university has its own version, so that many variations of the ETT problem exist, each with its specific rules, resources, constraints, and objectives. Given the enrolment of students to exams, the most basic version of the problem is to assign exams to periods in such a way that the resulting timetable is 'conflict-free', meaning that no student has to take more than one exam at a time. Finding a conflict-free timetable, however, is far from sufficient as this ignores the spread between exams taken by the same students. Carter et al. (1996) therefore propose to additionally penalize a timetable with a predetermined penalty whenever a student has to take two exams within a given timespan. The resulting problem is commonly referred to as the uncapacitated examination timetabling problem (UETT), and with only one classic problem instance solved to proven optimality so far (see Dimitsas et al., 2022), is known to be very challenging to solve.

In this work, we consider the capacitated examination timetabling (CETT) problem as proposed for the Second International Timetabling Competition (ITC-2007; see McCollum et al., 2010). This problem is more realistic than the UETT and consists of allocating every exam to a single room and period, though allowing exams to share a room as long as the total capacity of that room is respected. Another peculiarity of this problem is that periods have a different length, which precludes the assignment of specific exams to some specific certain periods, given that there is not enough time to run the exam. Other constraints relate to conflicting exams (i.e. exams with at least one student in common) assigned to the same period, rooms with insufficient capacity, and precedence relations between exams. The main objective involves spreading students' exams as much as possible over time, while also avoiding that exams with different lengths are scheduled together in the same room.

Over the years, numerous local search methods have been proposed for CETT. We observe, though, that almost all of them employ neighbourhoods that were originally designed for UETT. Moreover, the literature provides little insights into how different neighbourhoods contribute to the success of current algorithms. This paper therefore presents the following contributions. Firstly, we provide an extensive

✉ David Van Bulck
david.vanbulck@ugent.be

Dries Goossens
dries.goossens@ugent.be

Andrea Schaerf
andrea.schaerf@uniud.it

1 Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium

2 Flanders-Make@UGent - core lab CVAMO, Ghent, Belgium

3 Polytechnic Department of Engineering and Architecture, University of Udine, via delle Scienze 206, 33100 Udine, Italy

Published online: 19 December 2023

overview of existing neighbourhoods and their prevalence in the literature. An efficient implementation of these neighbourhoods is crucial to be competitive on the ITC-2007 benchmarks. Yet, it is not always clear how to implement more complex neighbourhoods like Kempe chains, especially since current state-of-the-art results have all been obtained using source code that has either become inaccessible or has been lost over time. Hence, this paper's first contribution lies in reconstructing the state-of-the-art and making the code for the existing neighbourhoods openly available.[1] Secondly, we introduce two innovative neighbourhoods that make better use of the underlying structure of CETT. One of them explicitly focusses on the assignment of exams to rooms and the other one exploits the decomposed structure of the underlying conflict graph. We combine the newly proposed neighbourhoods and those from the literature within the framework of a multi-neighbourhood simulated annealing (SA) metaheuristic and show how to use a principled procedure to tune its parameters. This approach resulted in several new best found solutions. The starting point for our algorithm is the work by Battistutta et al. (2017), who apply SA to this problem but consider only the two most basic neighbourhoods. A considerably larger set of neighbourhoods has been considered by Bellio et al. (2021), who, however, applied them only to the uncapacitated version of the problem. Finally, to assess the contribution of the individual neighbourhoods, we perform an ablation analysis.

The remainder of this paper is as follows. First, Sect. 2 describes CETT and the problem instances that are available in the literature. Section 3 then provides an overview of existing heuristics, the neighbourhoods employed, and their use in the literature. Next, Sect. 4 introduces the new set of neighbourhoods, and Sect. 5 explains how to integrate them into the framework of SA. Parameters of the algorithm are tuned in Sect. 6, which also contains the ablation analysis. Finally, Sect. 7 presents the computational results, and Sect. 8 concludes the paper with a summary of our findings.

2 Problem description

In this section, we provide a more detailed problem description of CETT, and we introduce the notation used in the remainder of this paper. The planning horizon is divided into a set of non-overlapping periods P, and each period has a certain length, belongs to a day, and possibly has a penalty for scheduling exams in it. Given is also a set of rooms R, with for each room a given capacity in terms of the number of seats available for students (assumed to be the same for all periods), and possibly a penalty for scheduling exams in it. The third and last entity consists of exams E, with for each exam its duration and the set of students that take the exam.

A feasible examination timetable is one which assigns each exam in E to exactly one period in P and exactly one room in R such that the following hard constraints are satisfied.

Conflict free	Exams with at least one student in common are scheduled in different periods. In other words, no student takes more than one exam at a time.
Room capacity	Multiple exams can be assigned to the same period and room, as long as the total number of students enrolled for those exams does not exceed the capacity of the room.
Period length	Exams are only assigned to periods with length greater than or equal to the duration of the exam.
Exam sequence	For some pairs of exams $e_1, e_2 \in E$, $e_1 \neq e_2$, sequence relations are given. In particular, precedence constraints require that e_1 comes before e_2, coincidence constraints that e_1 and e_2 are assigned to the same period, and exclusiveness constraints that e_1 and e_2 are assigned to different periods.

The objective function in CETT is composed of the following six soft constraints, such that each has a penalty weight depending on the problem instance being solved.

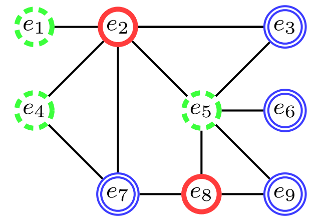
Two in a row	For each pair of exams scheduled in consecutive periods, a penalty equal to the number of students in common.
Two in a day	For each pair of exams scheduled in non-consecutive periods of the same day, a penalty equal to the number of students in common.
Period spread	For each pair of exams scheduled within a given number of periods, a penalty equal to the number of students in common. This spread limit is fixed at global level in the instance.
Front load	For each exam with more than a predetermined number of students scheduled later than a given period in time, a penalty of one.
Exam durations	For each room and period, a penalty equal to the number of distinct exam durations assigned to that period and room more than one.

[1] See https://github.com/davidvanbulck/ITC-2007-CETT.git.

Fig. 1 Relation between examination timetabling, graph colouring, and a conflict free timetable. Periods p_1, p_2, and p_3 correspond to solid, dashed, and double circles, respectively

Student	Exams
u	e_1, e_2
v	e_2, e_3, e_5
w	e_2, e_4, e_7
x	e_5, e_6
y	e_7, e_8
z	e_5, e_8, e_9

(a) Enrolment list



(b) Conflict graph

Slot	Exams
p_1	$\{e_2, e_8\}$
p_2	$\{e_1, e_4, e_5\}$
p_3	$\{e_3, e_6, e_7, e_9\}$

(c) Timetable

Period penalty For each exam scheduled in an undesirable period, a penalty of one.

Room penalty For each exam scheduled in an undesirable room, a penalty of one.

Notice that when the period spread limit is equal to two or more, the presence of two consecutive exams is penalized twice, namely as a 'Two in a row' violation and a 'Period spread' violation.

CETT comes along with a dataset of 12 real-life problem instances (mostly from British universities), which were used in the track on examination timetabling of the International Timetabling Competition 2007 (ITC-2007)[2]. Apart from these problem instances, Özcan and Ersoy (2005) propose a set of 8 real-life problem instances from Yeditepe University; these instances have been translated into the ITC-2007 format by Parkes and Özcan (2010). The Yeditepe instances, however, are somewhat smaller in size and ignore some of the constraints from the ITC-2007 formulation (for a discussion and a detailed overview of instance characteristics, (see Ceschia et al., 2022). All of the previous problem instances plus an additional set of 50 artificial ones from Battistutta et al. (2017) together with the best known solutions, are available from OptHub.[3]

3 Existing heuristics

The field of examination timetabling is relatively young, with pioneering papers dating back to the late 1970s and early '80s (for an overview, see e.g. Carter 1986; Schaerf 1999; Laporte and Desroches 1984). Ever since, the field has conceived considerable attention, which can perhaps be explained by the fact that examination timetabling was one of the first practical applications of graph colouring. Indeed, it is well known that finding a conflict-free timetable is equivalent to colouring the vertices of the associated conflict graph where nodes correspond to exams, colours to periods, and any two exams with at least one student in common need to receive different colours (see Fig. 1). We refer to Aldeeb et al. (2019) for a comprehensive review on recent advancements in uncapacitated timetabling, and focus in the remainder of this section on CETT. In particular, Sect. 3.1 discusses popular techniques to construct an initial solution, whereas Sect. 3.2 discusses commonly employed neighbourhoods as used in metaheuristics for CETT.

3.1 Constructive heuristics

Most of the initialization procedures in the literature on CETT focus on generating a conflict-free timetable by using heuristics from graph colouring (see also Carter et al., 1996) and subsequently consider the assignment of rooms in each period. In particular, the exams are typically listed in some order, and each time using a different random permutation of the periods, the exams of the list are repeatedly assigned to the first feasible period. The following rules have been proposed to sort the exams: (i) largest degree in the conflict graph, (ii) weighted degree in the conflict graph (edge weights correspond to the number of students in common), (iii) saturation degree counting the number of periods in which an exam can be assigned without violating any of the hard constraints (SD, dynamically updated after each assignment), (iv) number of students enrolments, and (v) random order. The rooms are subsequently sorted in increasing order of their residual capacity, and exams are repeatedly assigned to the first feasible room in the list. In the event that the scheduling process is unsuccessful, it is common to simply repeat the procedure or to use some sort of backtracking procedure. Numerous studies have demonstrated the effectiveness of the saturation degree sorting rule (see e.g. Carter et al. 1996; Alsuwaylimi and Fieldsend 2019), and as a result, most heuristics proposed in the literature generate an initial solution using this rule (see e.g. Bykov and Petrovic 2016; Leite et al. 2019). An interesting study that combines the various sorting rules into a hyperheuristic is presented by Pillay (2010).

In an effort to generate initial solutions of higher quality, Alsuwaylimi and Fieldsend (2019) propose the ordering-based scheduling initialization (OBSI). This algorithm prioritizes the scheduling of exams with a substantial number

[2] See also the official competition website at https://www.cs.qub.ac.uk/itc2007/.

[3] See https://opthub.uniud.it/problem/timetabling/edutt/ett/itc-2007-ett.

Fig. 2 Illustration of $Move(e_1, p_3, r_1)$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{\boxed{e_1}, e_4\}$
p_3	$\{e_3, e_6\}$	$\{e_7, e_9\}$

$\rightarrow$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{e_4\}$
p_3	$\{e_3, e_6, \boxed{e_1}\}$	$\{e_7, e_9\}$

Table 1 Overview of improvement heuristics and their neighbourhoods in the context of CETT

	Algorithm	Move	Swap	Kempe	Shake	Path Rel
McCollum et al. (2009)	Great deluge	✓	✓			
Müller (2009)	Hybrid local search	✓	✓			
Gogos et al. (2010)	Scatter search	✓		✓		✓
Gogos et al. (2012)	GRASP	✓		✓		
Burke et al. (2014)	Hyperheuristics	✓	✓	✓	✓	
Alzaqebah and Abdullah (2014, 2015)	Bee colony optimization	✓	✓			
Burke and Bykov (2016)	Adaptive Flex-deluge	✓		✓	✓	
Bykov and Petrovic (2016)	Step-counting hill-climbing	✓	✓	✓	✓	
Burke and Bykov (2017)	Late-acceptance hill-climbing	✓	✓	✓	✓	
Battistutta et al. (2017)	Simulated annealing	✓	✓			
Leite et al. (2019, 2021)	Simulated annealing & Threshold Acceptance	✓		✓		
Rajah and Pillay (2023)	Partial Solution Search	✓	✓	✓	✓	

of conflicts in the early and later portions of the examination timetable, referred to as the front section and the back section, respectively. Subsequently, it tries to schedule the remaining exams into the middle section of the timetable, thereby trying to satisfy the period spread constraints as much as possible (for more details, see Alsuwaylimi and Fieldsend 2019).

3.2 Improvement heuristics

The literature on CETT almost exclusively focusses on the design of metaheuristics based on local search (see Ceschia et al. 2022). The proposed methods range from simulated annealing (e.g. Battistutta et al. 2017; Leite et al. 2019), over Great Deluge (e.g. McCollum et al. 2009) and Scatter Search (e.g. Gogos et al. 2010), to Late-Acceptance Hill-Climbing (e.g. Burke and Bykov 2017). While the list of applied algorithms is too long to enumerate in this paper, some of the more common local search techniques together with the neighbourhoods they use are summarized in Table 1. In the remainder of this section, we explain each of these neighbourhoods in more detail.

3.2.1 Move

Perhaps the most common of all is the neighbourhood $Move(e, p, r)$, which simply reallocates exam $e \in E$ to period $p \in P$ and room $r \in R$ (see Fig. 2). In case only the room changes, the operator is often described in the literature as the 'room move'. Table 1 shows that all considered heuristics from the literature implement this neighbourhood.

3.2.2 Swap

Moving a given exam to a certain period does not always result in another feasible timetable, typically because there is a conflicting exam in that period or there is no room with sufficient residual capacity. The neighbourhood $Swap(e_1, e_2)$ tries to circumvent this by considering two exams $e_1, e_2 \in E$, $e_1 \neq e_2$, and swapping the room and period currently assigned to e_1 and e_2 (see Fig. 3). Perhaps thanks to its simplicity to implement, Table 1 again shows that $Swap$ is very popular in the literature.

3.2.3 Kempe

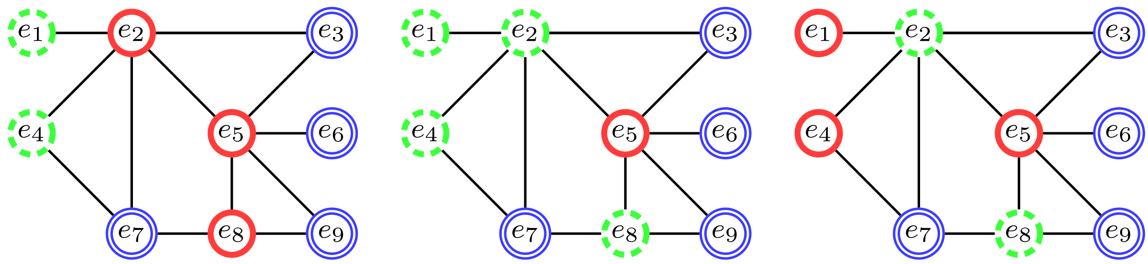
To the best of our knowledge, the $Kempe(e, p, r)$ operator was proposed for the first time in the context of examination timetabling by Thompson and Dowsland (1998). Similar to $Move$, it starts by reallocating exam $e \in E$ to a new room $r \in R$ and new period $p \in P$. However, this time it is assumed that there is at least one conflicting exam in p and that any newly introduced conflict is repaired by using a so-called Kempe chain. To this purpose, it first moves all exams in conflict with exam e, denote them by $C_1 \subseteq E$, from period p to the period originally assigned to exam e (denote this period with s). In turn, all exams in conflict with the exams in C_1 (denote them with C_2) are moved from period s to

Fig. 3 Illustration of $Swap(e_4, e_7)$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{e_1, \boxed{e_4}\}$
p_3	$\{e_3, e_6\}$	$\{\boxed{e_7}, e_9\}$

$\rightarrow$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{e_1, \boxed{e_7}\}$
p_3	$\{e_3, e_6\}$	$\{\boxed{e_4}, e_9\}$



(a) e_5 from p_2 to p_1 (b) e_2 and e_8 from p_1 to p_2 (c) e_1 and e_4 from p_2 to p_1

Fig. 4 Illustration of the Kempe chain associated with $Kempe(e_5, p_2, r)$. The neighbourhood starts from the timetable in Fig. 1 Recall that periods p_1, p_2, and p_3 correspond to solid, dashed, and double circles, respectively

Fig. 5 Illustration of $Shake(p_1, p_2)$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{e_1, e_4\}$
p_3	$\{e_3, e_6\}$	$\{e_7, e_9\}$

$\rightarrow$

Slot	r_1	r_2
p_1	$\{e_5\}$	$\{e_1, e_4\}$
p_2	$\{e_2\}$	$\{e_8\}$
p_3	$\{e_3, e_6\}$	$\{e_7, e_9\}$

period p, and so on until all newly introduced conflicts are solved (see Fig. 4).

The Kempe operator has proven very effective in the context of CETT, but it is not always clear how to deal with the room assignment for exams in C_1 and C_2. Burke and Bykov (2016) propose to use the following simple rule based on the best fit algorithm for bin packing: (i) sort exams in C_1 and C_2 in decreasing size of students, and (ii) for each exam in the list, assign the exam to the smallest room with sufficient residual capacity (or the room with the largest residual capacity if no such room exists).

3.2.4 Shake

The neighbourhood $Shake(p_1, p_2)$ was originally proposed by Di Gaspero (2002), and moves all exams currently assigned to period $p_1 \in P$ to period $p_2 \in P$, $p_1 \neq p_2$, and the other way around. The assignment of rooms remains unaffected by the operator. Figure 5 illustrates the move.

3.2.5 PathRelinking

Different from the previous operators, the $PathRelinking$ (A, B) operator proposed by Gogos et al. (2010) takes as argument two 'reference' timetables A and B. Assuming that A is the better of the two solutions, the operator gradually transforms solution A into solution B by using the $Move$ operator to repeatedly assign a randomly chosen exam in solution A to the period and room assigned to the exam in solution B (see Fig. 6). During this transformation process, all feasible solutions are stored and at the end 'promising' intermediate solutions are returned. The fact that improvement heuristics typically store only one solution at a time perhaps explains why the operator is not often used in the literature (see Table 1).

3.2.6 Kick

The $Kick(e_1, e_2, p)$ operator was proposed by Di Gaspero (2002) in the context of the UETT and is a generalization of the $Swap(e_1, e_2)$ operator. It starts by assigning exam e_1 to the period currently assigned to exam e_2 after which it 'kicks out' e_2 to given period $p \in P$. Its simplicity and use in UETT notwithstanding (see e.g. Bellio et al. 2021), the Kick operator has, to the best of our knowledge, not yet been applied in the context of CETT (and is therefore not included in Table 1). We propose to generalize it to $Kick(e_1, e_2, p, r)$ by not only moving e_1 to the period but also the room currently assigned to e_2, and by kicking out e_2 to the given period p and a given room $r \in R$ (Fig. 7).

Fig. 6 Illustration of the first iteration of $PathRelinking(A, B)$. The operator creates a new solution by starting from solution A and moving the randomly chosen exam e_2 to the period and room currently assigned in solution B

(A)

Slot	r_1	r_2
p_1	$\{e_3, e_6\}$	$\{e_7, e_9\}$
p_2	$\{e_5\}$	$\{e_1, e_4\}$
p_3	$\{\boxed{e_2}\}$	$\{e_8\}$

(B)

Slot	r_1	r_2
p_1	$\{\boxed{e_2}\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{e_1, e_4\}$
p_3	$\{e_3, e_6\}$	$\{e_7, e_9\}$

→

Slot	r_1	r_2
p_1	$\{\boxed{e_2}, e_3, e_6\}$	$\{e_7, e_9\}$
p_2	$\{e_5\}$	$\{e_1, e_4\}$
p_3		$\{e_8\}$

Fig. 7 Illustration of $KickExams(e_1, e_6, p_1, r_2)$

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8\}$
p_2	$\{e_5\}$	$\{\boxed{e_1}, e_4\}$
p_3	$\{e_3, \boxed{e_6}\}$	$\{e_7, e_9\}$

→

Slot	r_1	r_2
p_1	$\{e_2\}$	$\{e_8, \boxed{e_6}\}$
p_2	$\{e_5\}$	$\{e_4\}$
p_3	$\{e_3, \boxed{e_1}\}$	$\{e_7, e_9\}$

4 A new set of neighbourhoods

This section proposes two novel neighbourhoods that try to better exploit the underlying problem structure of CETT. More in particular, Sect. 4.1 proposes a beam search operator to assign exams to rooms, and Sect. 4.2 exploits the fact that the underlying conflict graph of a CETT problem instance often contains multiple disconnected components.

4.1 Beam search for the assignment of exams to rooms

It is striking that all neighbourhoods from Sect. 3.2 but $PathRelinking$ were originally proposed in the context of UETT, where the focus is on period-related constraints and penalizations. They have been straightforwardly adapted to deal with the assignment of exams to rooms. Neighbourhoods focussing solely on room-related violations therefore remain relatively unexplored. In fact, apart from the special cases of $Move$ and $Swap$ operating on exams within the same period (see Müller 2009), the only research that we are aware of is by Gogos et al. (2012). They propose as a last step in their heuristic to further optimize a timetable by solving for each period an integer programming formulation (IP) that optimally reassigns the exams to rooms. A similar idea has been used in the context of course timetabling (see e.g. Chiarandini et al. 2006). While the idea of optimizing the room assignment per period is promising, it seems inefficient to directly use such IP as part of a new neighbourhood as it is likely too time consuming within the context of a local search framework where a neighbourhood is typically visited thousands if not millions of times. Instead, we consider the room assignment of only a subset of the exams assigned to the same period, and further speed up the calculations by using beam search on the full enumeration tree (thus eliminating the use of IP solvers). In particular, the move $Beam(p, \alpha, \beta)$ takes as arguments period $p \in P$, beam depth α, and beam width β. The neighbourhood starts by randomly selecting α exams currently assigned to period p, and sorts these exams in decreasing order of the number of enrolled students. Next, it constructs a search tree where each level in the tree corresponds to an exam and nodes decide on the room assigned to that exam. The result is a room assignment that minimizes the costs for the subset of exams considered. In order to avoid an explosion of the tree, only the β most promising nodes at each level are further expanded to the next level. Moreover, instead of using the bin-packing heuristic from Sect. 3.2.3, we note that the $Beam$ operator can also be used to reassign rooms to the exams in the chains of the $Kempe$ operator. Figure 8 illustrates the move.

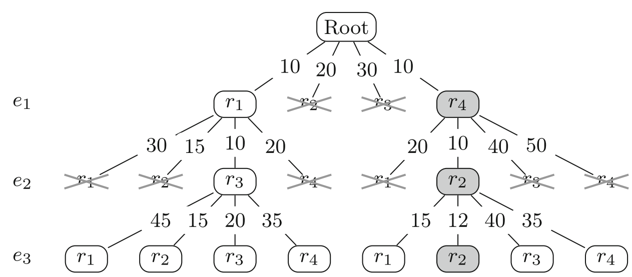
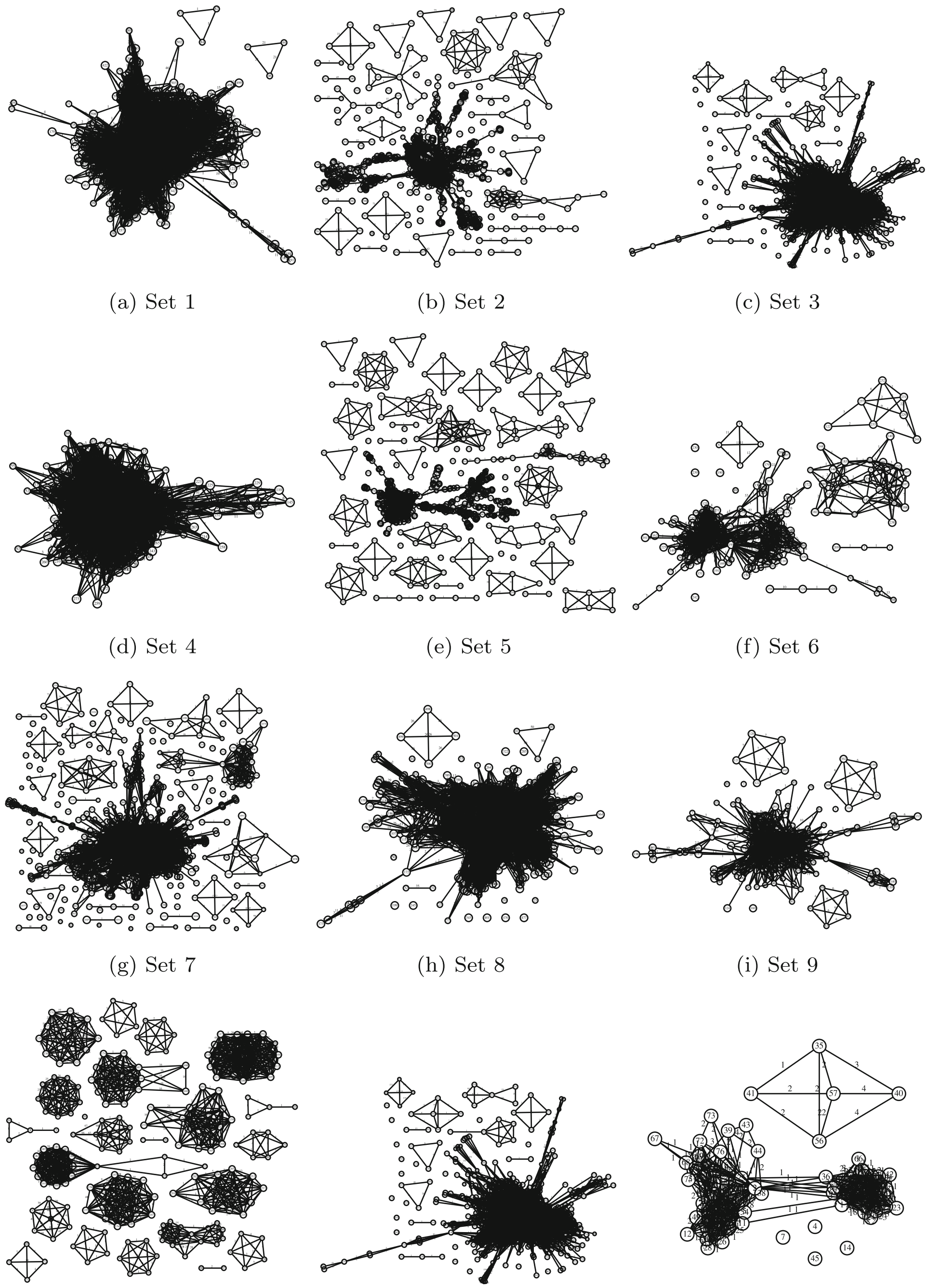


Fig. 8 Illustration of $Beam(2, 3, 2)$. Edge weights represent fictive room assignment costs

4.2 Disconnected component operator

When the conflict graph associated with a problem instance of the UETT is disconnected, Dimitsas et al. (2022) observe



(a) Set 1 (b) Set 2 (c) Set 3
(d) Set 4 (e) Set 5 (f) Set 6
(g) Set 7 (h) Set 8 (i) Set 9
(j) Set 10 (k) Set 11 (l) Set 12

Fig. 9 Visualization of the conflict graphs for the ITC-2007 problem instances

that—without loss of optimality—the problem instance can be split into multiple independent subproblems that each are considerably easier to solve. Although Fig. 9 shows that all but one of the conflict graphs associated with the ITC-2007 problem instances are disconnected, several 'coupling' constraints (e.g. room capacity) cause that CETT problem instances cannot be decomposed in a similar way. Nevertheless, the degree to which the graph is disconnected may still explain how difficult a problem instance is to solve. In this regard, we propose to include the number of disconnected components and variants thereof (e.g. the edge-connectivity of the conflict graph) as instance features on top of those already proposed before in the literature (see Ceschia et al. 2022; Battistutta et al. 2017).

Moreover, the high number of disconnected components in several conflict graphs of Fig. 9 motivated us to think about novel neighbourhoods that specifically exploit the disconnected structure of the conflict graph. In particular, we

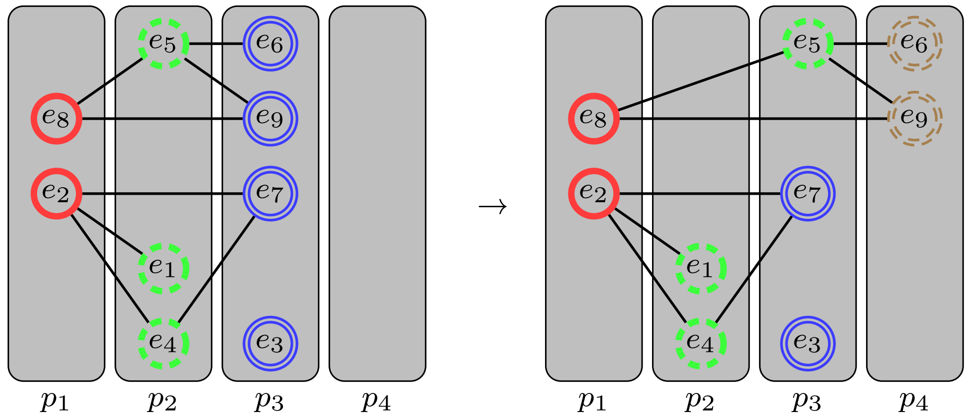


Fig. 10 Illustration of $Component(D_1, p_2, 1)$. To make the underlying conflict graph disconnected, we removed students v and y from the enrolment matrix. The resulting conflict graph has three connected components: $D_1 = \{e_5, e_6, e_8, e_9\}$, $D_2 = \{e_1, e_2, e_4, e_7\}$, and $D_3 = \{e_3\}$ with $P_{D_1} = P_{D_2} = \{p_1, p_2, p_3\}$ and $P_{D_3} = \{p_3\}$

observe that permuting the time slots of a disconnected component affects the period spread costs in a predictable way, without causing additional violations of the conflict-free constraint. To this purpose, let D be a disconnected component in the associated conflict graph, and denote with the ordered set $P_D \subseteq P$ the periods in which exams of D are scheduled. Finally, let k be a strictly negative or positive number such that for all $p \in P_D$ it holds that $p + k \leq |P|$ if k is positive and $p - |k| \geq 1$ if k is negative. If k is positive, the move $Component(D, p, k)$ moves all exams from the disconnected component D scheduled in period $q \in P_D$, $p \leq q$, to period $q + k$. If k is negative, it moves all exams from D scheduled in period q, $q \leq p$, to period $q - |k|$. Note that this operator either increases the spread between exams, or leaves it unchanged (see Fig. 10). To deal with the assignment of rooms, the move makes use of the bin packing heuristic explained in Sect. 3.2.3.

5 Multi-neighbourhood simulated annealing

In order to assess the effectiveness of the existing as well as the new neighbourhoods, this section explains how we combine the neighbourhoods into a multi-neighbourhood SA algorithm. The choice for SA is motivated by its effectiveness for this (Battistutta et al., 2017; Leite et al., 2019) and several other timetabling problems (see e.g. Bellio et al. 2016; Ceschia et al. 2012; Rosati et al. 2022; Bellio et al. 2021). Section 5.1 introduces the considered search space, Sect. 5.2 the initial solution strategy, Sect. 5.3 the neighbourhood relations, and finally Sect. 5.4 the simulated annealing framework.

5.1 Search space

In the spirit of Battistutta et al. (2017), we consider as search space all complete assignments of exams to periods and rooms, including infeasible ones. In order to deal with the hard constraints, the proposed algorithm offers a parameter with three different options. The first option immediately rejects neighbours that result in an increase of the number of violated hard constraints. Hence, when provided with an initial feasible solution, this option explores the feasible part of the search space only (see e.g. Bykov and Petrovic 2016). The second option adds a penalty to the objective function for each violated hard constraint (see e.g. Battistutta et al. 2017). Finally, the third option is a compromise between the first two: *Move*, *Swap*, and *Kick* are allowed to enter the infeasible part of the search space, whereas the other neighbourhoods are immediately rejected in case the number of violated hard constraints increases. As such, high-quality solutions can still be approached from the infeasible as well as feasible part of the search space, while still saving some time in the evaluation of the more expensive neighbourhoods.

5.2 Initial solution generation

We consider three different approaches to construct an initial solution. The first approach is the simplest one and constructs an initial solution by repeatedly assigning each exam to a period of sufficient length and a randomly selected room (ignoring all hard constraints; see e.g. Battistutta et al. 2017). The second and third initialization techniques are based on the saturation degree (SD) and ordering-based scheduling initialization (OBSI) heuristic (see also Sect. 3.1). To speed up the generation of an initial solution, however, we do not consider any restart or backtracking techniques but instead assign exams that cannot be added to the partial timetable without violating any of the hard constraints to a randomly chosen room and period of sufficient length. Although the initial solution is thus complete in the sense that all exams are assigned a room and period, it may violate some of the hard constraints. We note that this is different from several methods in the literature that assume a feasible starting solution is given (see e.g. Burke and Bykov 2016; Bykov and Petrovic 2016).

5.3 Neighbourhood selection

The proposed metaheuristic makes use of the following six neighbourhoods introduced earlier in this paper: $Move$, $Kick$, $Kempe$, $Beam$, $Shake$, and $Component$. The metaheuristic is parametrized with a selection probability for each of the six operators, and in addition contains a parameter to control the fraction of $Kick$ moves that correspond to $Swap$. Moreover, for the $Move$ operator we additionally include a bias parameter that controls for the fraction of moves and kicks that leave the room assignment unchanged (the evaluation of such moves is typically faster). Similarly, for the $Kick$ operator we include a bias parameter that determines the percentage of neighbours sampled for which exams e_1 and e_2 belong to the same period. At each iteration, the selection probabilities are used to select a neighbourhood from which we subsequently select a neighbour with uniform probability.

5.4 Simulated annealing

After selecting a neighbour, the associated change in the objective function (referred to as Δ) is computed and its acceptance is determined by the well-known Metropolis criterion. In other words, improving moves (i.e. $\Delta < 0$) and sideways moves (i.e. $\Delta = 0$) are always accepted, whereas deteriorating moves are accepted only with probability $e^{-\Delta/T}$ where T is a temperature parameter controlled by the algorithm. The temperature is initialized with value T_0 at the beginning of the search and is subsequently reduced by multiplying it with a cooling rate $0 < \gamma < 1$ every m iterations until it reaches the end temperature T_f. The value of m is automatically determined such that the total time of the algorithm does not surpass a predefined time limit.

Our version of the simulated annealing algorithm additionally makes use of a so-called cut-off mechanism (see Johnson et al. 1989). Based on the assumption that the number of moves accepted at early temperatures is important rather than the number of moves performed, the cut-off mechanism speeds-up the algorithm during early iterations by reducing the temperature until either m moves have been performed or $\delta \cdot m$ moves have been accepted, where $0 < \delta < 1$ is the cut-off factor.

6 Programming by optimization

During the development of the algorithm, we avoided premature commitment to certain design choices by including these choices as parameters of the algorithm (a paradigm known as programming by optimization, see Hoos 2012). Section 6.1 provides an overview of these parameters as well as the method we used to select their final values. Next, Sects. 6.2 and 6.3 provide more insights into which of the parameters are most critical to achieve the final performance of the algorithm.

6.1 Parameter tuning

An overview of all parameters can be found in Table 2. The first group of parameters regulates the initialization method, the restrictions on the search space, and the penalty weight for violated hard constraints. The second group of parameters regulates the selection and configuration of the neighbourhoods. Next, parameters are included for the depth and width of the enumeration tree in the $Beam$ neighbourhood and for the decision whether to reassign rooms in the Kempe chain using the beam operator (in which case an appropriate beam width is selected) or the bin packing heuristic from Sect. 3.2.3. Finally, the third group of parameters configures the Simulated Annealing mechanisms by setting the start and expected end temperature, and the cooling and cut-off rate.

In order to tune the parameters of the algorithm, we used the `irace` package which performs an iterated F-racing procedure consisting of the following three steps (see also López-Ibáñez et al. 2016). First, a number of parameter configurations are sampled from a particular distribution. Second, the best configurations are determined by means of racing: at each step of the race the candidate solutions are tested on a single instance, after which the candidate configurations that perform statistically worse are discarded. Third, the parameter configurations that survived after the last step of the race are used to update the sampling distributions.

In total, `irace` was provided with a budget of 10,000 runs of the algorithm. As training data, we used the set of 50 artificial problem instances proposed by Battistutta et al. (2017). This resulted in six parameter configurations that, according to `irace`, are statistically performing equally well on the training data. To choose a final configuration, we ran the algorithm ten times on each of first eight problem instances (instance 9-12 were not available to the participants of the competition), and selected the configuration with the best mean performance. The last two columns of Table 2 provide the range of possible parameter values that we considered, and the selected values.

6.2 Sampling distributions

The use of `irace` provides us with a final set of parameter configurations, but since `irace` acts like a black box, it does not provide much insights into which of the algorithm parameters are critical, let alone why.

To get a better understanding of the parameter choices, Fig. 11 plots the frequency for each of the parameters as sampled by `irace` during the last iteration of the tuning process. These graphs give us a first indication of how important the different parameters are: if a plot resembles a uniform distri-

Table 2 Overview of the parameters of the algorithm

Parameter	Considered values	Selected value
Initialization method	{Random, SD, OBSI}	OBSI
Search space restriction	{Feasible, Infeasible, Mixed}	Feasible
Hard constraint penalty	[50, 500]	279
Probability $Move$	$p_{Move} = 1 - p_{Kick} - p_{Kempe} - p_{Shake} - p_{Beam} - p_{Component}$	0.51
Room bias $Move$	[0, 0.5]	0.17
Probability $Kick$ (p_{Kick})	[0.25, 0.5]	0.43
Swap bias $kick$ ($p_{SwapBias}$)	[0.7, 1]	0.93
Room bias $Kick$	[0, 0.5]	0.04
Probability $Kempe$ (p_{Kempe})	[0, 0.2]	0.03
Kempe room assignment	{Bin packing, Beam}	Bin packing
Kempe beam width	[1, 5]	-
Probability $Shake$ (p_{Shake})	[0, 0.1]	0.09
Probability $Beam$ (p_{Beam})	[0, 0.1]	0.01
Beam depth (α)	[3, 10]	7
Beam width (β)	[2, 5]	2
Probability $Component$ ($p_{Component}$)	[0, 0.10]	0.03
Start temperature (T_0)	[200, 1000]	749
Expected end temp. (T_f)	[0.05, 1.5]	0.72
Cooling rate (γ)	[0.95, 0.99]	0.97
Cut-off rate (δ)	[0.10, 0.20]	0.20

bution its parameter is perhaps not so important, whereas a distribution with several ‘spikes’ may hint that some parameter values work better than others.

With regard to the first group of parameters, we observe that `irace` has a mild preference to explore the feasible part of the search space only. Perhaps, this can best be explained by the fact that substantial computation time can be saved by aborting the evaluation of moves as soon as the number of hard constraint violations increases, thus allowing more time to evaluate interesting moves near the end of the search. Furthermore, it seems that `irace` prefers to start from a random solution, although all final eight best performing configurations made use of OBSI. This is interesting, since most of the contributions in the literature make use of the SD method.

Looking at the neighbourhood selection probabilities, we observe that `irace` prefers to include all neighbourhoods, selecting $Move$ and $Swap$ ($p_{Kick} * p_{SwapBias}$) most often. It is interesting to see that $Kempe$ is not so often selected, which can perhaps best be explained by the computational cost in evaluating the associated moves and the fact that the much simpler $Swap$ and $Kick$ operators may already solve some of the conflicts. On the other hand, $Shake$ and $Component$ are selected quite often, perhaps thanks to their ability to significantly alter a solution without introducing new conflicts thus enabling the algorithm to escape local optima.

6.3 Ablation analysis

In order to find out which neighbourhoods are most critical to the performance of the algorithm, we conduct an ablation analysis (see Fawcett and Hoos 2016; Bellio et al. 2021). This analysis transforms a source configuration into a target configuration by iteratively assigning to the parameters of the source configuration the values assigned in the target configuration. During each iteration of this process, we select the configuration that exhibits the best performance as the starting point for the subsequent iteration.

Our source configuration includes the $Move$ neighbourhood only ($p_{Move} = 1$), while the target configuration is the best found configuration by `irace` (i.e. the last column of Table 2). In other words, we iteratively reactivate neighbourhoods by setting their selection probabilities to the target configuration value from `irace`. As such, left-out neighbourhoods in any of the intermediate configurations from source to target contribute to the selection probability of $Move$. In order to determine the best configuration at each iteration, we compute the reduction in the median relative gap with regard to the performance of the source configuration using 10 repetitions for each of the ITC-2007 problem instances. Intermediate configurations are denoted by the neighbourhoods that are reactivated: S_w, K_i, K_e, S_h, B and C stand for $Swap$, $Kick$, $Kempe$, $Shake$, $Beam$ and $Component$, respectively. For example,

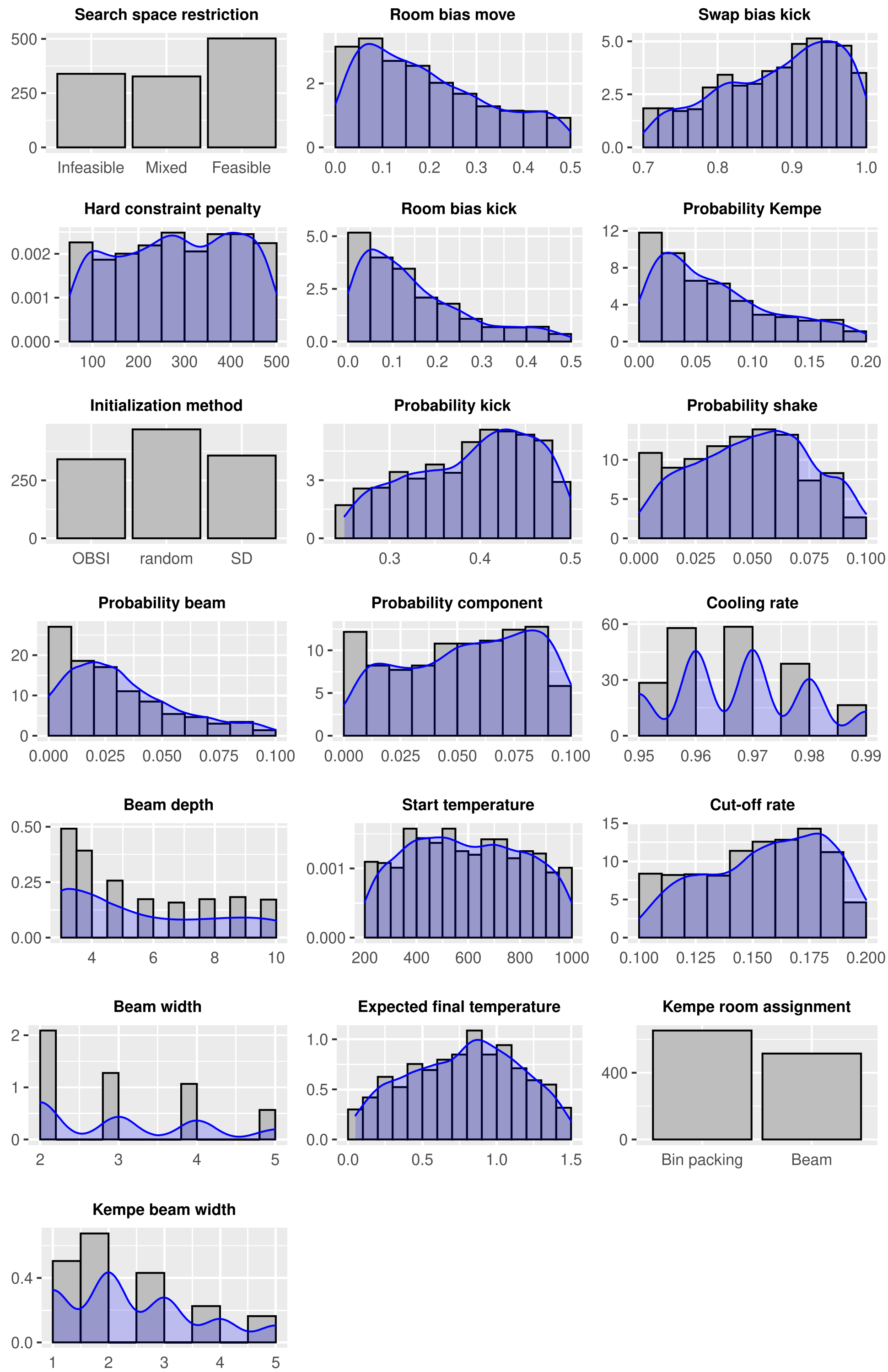
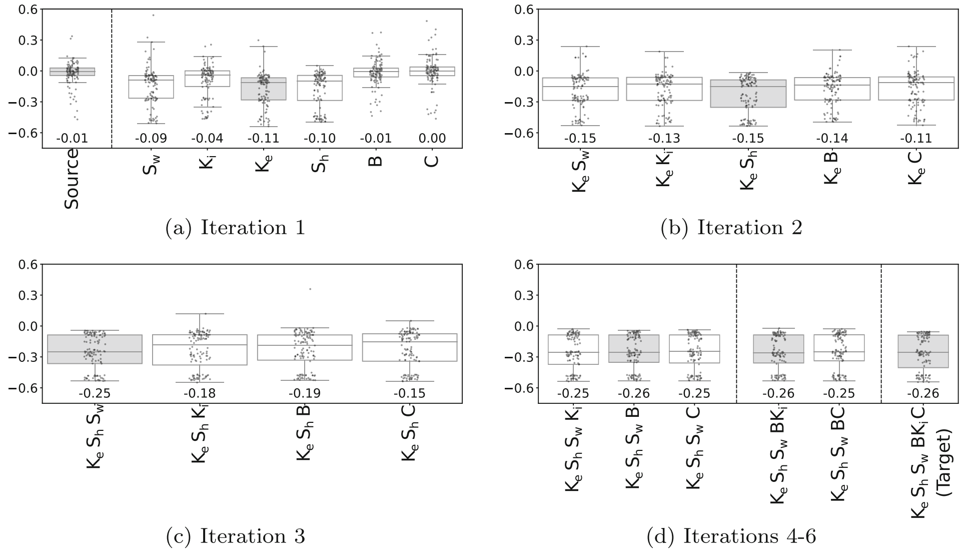


Fig. 11 Frequency of parameters as sampled by `irace`

$K_e S_h$ indicates that the moves *Kempe* and *Shake* are reactivated with probabilities given in Table 2, and the moves *Swap*, *Kick*, *Beam* and *Component* are inactive. Note that all parameters other than the neighbourhood probabilities remain unchanged over the course of the analysis.

Starting by analysing the contribution of adding a single neighbourhood to the source configuration, Fig. 12 shows that the contribution of *Kempe* is largest. Following closely are the *Shake* and *Swap* neighbourhoods, which are reactivated in iteration 2 and 3, respectively. The stand-alone contribution for each of the three neighbourhoods in iteration

Fig. 12 Ablation analysis. Numbers below the boxplots show the median reduction in the relative gap when reactivating the neighbourhoods of that configuration. The box plot of the best performing configuration of each iteration is shown in grey



1 is about 10%, while iteration 3 shows that the combined inclusion yields a median improvement in solution quality of about 25%. This suggests a strong level of complementarity among these neighbourhoods. Especially the large contribution of *Shake* was somewhat unanticipated, and is interesting since some of the contributions from the literature do not consider this neighbourhood (see Table 1) which provides an interesting opportunity to further improve those algorithms.

In the last three iterations, the ablation analysis sequentially reactivates *Beam*, *Kick*, and *Component*. However, these operators do not seem to have a profound impact on the overall performance, even though a reduction in the relative gap of just 1% can still be quite substantial in absolute terms. Furthermore, it is important to note that the ablation analysis solely relies on the ITC-2007 problem instances, and the contribution of the neighbourhoods could vary for different instances. For example, considering the synthetic training instances introduced by Battistutta et al. (2017), Fig. 11 already hinted the added value of the *Component* neighbourhood. We also expect that the significance of *Beam*, *Kick*, and *Component* will increase with an extended runtime. Indeed, it is likely that the algorithm has not yet fully converged within the limited runtime permitted by the ITC-2007 competition (as discussed in Sect. 7). In such a scenario, the algorithm has not completely exploited the potential of the more basic operators, and hence introducing more sophisticated neighbourhoods aimed at facilitating deeper exploration or search diversification might not be justifiable.

7 Experimental results

All software was implemented with C++ and compiled using g++ (v. 11.3). The experiments were run on a Red Hat Enterprise Linux (RHEL) 8 machine equipped with one single core on an Intel Xeon E5-2660 CPU running at 2.60 GHz. Using the official ITC-2007 processing speed benchmarking tool allowed to run the simulated annealing algorithm with a time limit of 264 s. To respect the ITC time limit, for each instance independently, we first ran the algorithm with few evaluations and measured the computation time. We then adjusted the maximum evaluations to fit within the ITC limit, preventing any significant time overrun. In order to facilitate the benchmarking and development of new neighbourhoods for CETT, all source code used in this paper is publicly available at https://github.com/davidvanbulck/ITC-2007-CETT.git.

Table 3 first shows a comparison between our results and those of the algorithms that adhere to the ITC-2007 competition rules: that is, the execution time of the algorithm does not exceed the time limit provided by the ITC-2007 benchmarking tool (on an individual instance basis), it is not assumed that an initial feasible solution is available, and the average and best-out-of-ten results are reported. Had we participated at that time, it is clear that our solver would have won the competition (recall that the ITC benchmarking tool accounts for the improvements in processor speed). Indeed, for all but one of the instances, our solver outperforms others in terms of the best as well as the average solutions found. Given that the parameters of our solver were tuned on the artificial instances only, thus not overfitting to the ITC-2007 instances like the other algorithms, these results are especially encouraging.

Table 4 provides an overview of the best solutions found by other solvers from the literature and the best known lower bounds (column ‘LB’, retrieved from Ceschia et al. 2022). Although these solvers do not strictly adhere to the ITC-2007 rules (e.g. they violate the ITC time limit, or start from a given feasible solution), we note that the running time for all solvers is more or less in the order of the ITC-2007 time limit. The only exception to this is the solver by Gogos et al. (2010) who

Table 3 Comparison with results according to the ITC-2007 rules

	Competition finalists															
	Mueller		Gogos		De Smet		Atsuta		Pillay		Leite et al. (2019)		Leite et al. (2021)		Us	
No	Aver	Best	Aver	Best	Aver	Best	Aver	Best	Aver	Best	Aver	Best	Aver	Best	Aver	Best
1	4575	4370	6064	5905	6671	6670	9084	8006	12819	12035	5232	5050	5314	5089	**3868**	**3797**
2	414	400	1049	1008	623	623	3669	3470	3926	2886	405	395	410	395	**398**	**385**
3	10789	10049	14134	13771	✗	✗	19367	17669	19812	15917	9940	9574	10092	9711	**9167**	**8628**
4	21639	18141	20667	18674	✗	✗	26347	22559	25729	23582	12993	12299	13011	12235	**12138**	**11422**
5	3321	2988	4229	4139	3858	3847	4920	4638	11176	6860	3490	3115	3470	3184	**2779**	**2676**
6	27809	26585	28078	27640	28155	27815	29935	29155	34029	32250	26009	25750	26136	25975	**25650**	**25510**
7	4396	4213	6760	6572	5432	5420	11004	10473	19669	17666	4535	4308	4515	4369	**3881**	**3768**
8	7950	7742	10809	10521	✗	✗	14870	14317	16721	15592	**7603**	7506	7671	**7488**	7834	7743
9	1085	1030	1204	1159	1288	1288	1936	1737	2277	2055	1018	977	1024	985	**969**	**939**
10	18581	16682	✗	✗	14778	14778	15580	15085	20333	17724	13632	13449	13704	13487	**13566**	**13353**
11	34129	34129	49861	43888	✗	✗	✗	✗	44277	40535	31792	30112	33938	32189	**27659**	**26789**
12	6403	5535	✗	✗	✗	✗	5542	5264	7179	6310	5205	5148	5233	5148	**5199**	**5132**

Results of the competition finalists are retrieved from the official competition website. The lowest average and best results are indicated in bold, while a cross mark means that no feasible solution was found

Table 4 Comparison with best found results

No.	LB	OptHub	Battistutta et al. (2017)	Burke and Bykov (2016)	Bykov and Petrovic (2016)	Burke and Bykov (2017)	Gogos et al. (2010)	Us			
								ITC	ITCx10	ITCx100	Tuning
1	0	3488	3752	3691	3647	3787	4128	3650	3339	**3085***	
2	10	**380**	385	385	385	402	**380**	**380**	**380**	**380**	
3	670	**7041**	8175	7359	7487	7378	7769	8407	7744	7147	
4	1620	11806	13681	11329[†]	11779[†]	13278	13103	11084	10648	**10448***	
5	0	**2327**	2544	2482	2447	2491	2513	2510	2397	2341	
6	22875	25145	25560	25265	25210	25461	25330	25330	25175	**25120***	
7	0	3424	3517	3608	3563	3589	3537	3659	3393	**3225***	
8	1250	7356	7505	**6818**[†]	6614[†]	6701[†]	7087[†]	7461	7233	7077	
9	0	904	941	902[†]	924	997	913	1000	899	**898***	
10	0	**12878**	13530	12900	12931	13013	13053	13339	13263	13264	
11	3970	22465	26114	22875	23784	22959	24369	25755	23357	22343	**22186***
12	2030	**5095**	5153	5107	5097	5234	**5095**	**5095**	**5095**	**5095**	

New best solutions are denoted with an asterisk, and have been uploaded to OptHub. (†) These solution have been lost before being validated and are therefore not reported on OptHub

Table 5 Results for the Yeditepe dataset

	Müller (2009)	Muklason et al. (2017)		Rajah and Pillay (2023)		Us		
	Best (10)	Med	Best (21)	Avg	Best (30)	Avg	ITC (100)	Tuning
yue20011	62	68	56	50	**47**	49	**47**	
yue20012	125	161	122	109	102	105	**97***	
yue20013	**29**	29	**29**	29	**29**	29	**29**	
yue20021	70	111	76	55	47	50	**46***	
yue20022	170	212	162	151	129	128	121	**119***
yue20023	70	61	**56**	**56**	**56**	56	**56**	
yue20031	223	206	143	130	99	97	**79***	
yue20032	440	479	434	385	359	329	305	**296***

Numbers in parentheses provide the total number of random seeds considered. New best solutions are denoted with an asterisk and have been uploaded to OptHub

do not consider any hardware or time limit, and who instead focus on finding new best solutions. Following Gogos et al. (2010), we also report our algorithm's best solutions derived from 500 runs. These runs employ time limits of one time (ITC), ten times (ITCx10), and one hundred times (ITCx100) the official ITC-2007 time limit. Furthermore, any improved solutions discovered during algorithm development and tuning are recorded in the last column of Table 4. Given the increased runtime granted to our algorithm, it would be unfair to use the last three columns of Table 4 to directly compare the performance of our algorithm against previous state-of-the-art work. Nevertheless, the results highlight the potential of our solver. Notably, we've established new best-known solutions for half of the problem instances (all validated and uploaded to OptHub), while matching the best-known solution for two other instances. These results are remarkable, given that tens of algorithms have been proposed for CETT in the past.

Finally, the outcomes for the Yeditepe problem instances are outlined in Table 5. As these problem instances turn out somewhat easier to solve, we conduct a reduced number of 100 runs per problem instance, solely adhering to the original ITC-2007 time limit. Across all examined problem instances, our algorithm demonstrates consistent or even improved performance in terms of both average and best solution quality. Impressively, for 5 out of the 8 problem instances, we achieve new best solutions.

8 Conclusion

This study focused on solving the capacitated examination timetabling problem (CETT) proposed in the ITC-2007 competition. An extensive overview of existing neighbourhoods provided the foundation for the development of two new neighbourhoods, which explicitly optimize the assignment of exams to rooms and exploit the presence of disconnected components in the underlying conflict graph. These neighbourhoods were then integrated into a simulated annealing framework with parameters tuned using `irace`. In addition, an ablation analysis revealed the significant impact of including an often overlooked neighbourhood from the literature. Despite the plethora of algorithms proposed for CETT, the proposed heuristic found new best solutions for half of the ITC-2007 problem instances.

For future work, variants of the neighbourhood that exploit the presence of disconnected components, such as shuffling instead of moving the periods in which exams are scheduled, can be developed. Additionally, it would be interesting to explore how this neighbourhood could be adapted to deal with weakly connected subgraphs.

Funding David Van Bulck is a postdoctoral research fellow funded by the Research Foundation Flanders (FWO) [12AXE24N]

References

Aldeeb, B. A., Al-Betar, M. A., Abdelmajeed, A. O., Younes, M. J., AlKenani, M., Alomoush, W., Alissa, K. A., & Alqahtani, M. A. (2019). A comprehensive review of uncapacitated university examination timetabling problem. *International Journal of Applied Engineering, 14*, 4524–4547.

Alsuwaylimi, A. A., & Fieldsend, J. E. (2019). A new initialisation method for examination timetabling heuristics. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1636–1643). https://doi.org/10.1109/SSCI44817.2019.9002989.

Alzaqebah, M., & Abdullah, S. (2014). An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling. *Journal of Scheduling, 17*, 249–262.

Alzaqebah, M., & Abdullah, S. (2015). Hybrid bee colony optimization for examination timetabling problems. *Computers & Operations Research, 54*, 142–154.

Battistutta, M., Schaerf, A., & Urli, T. (2017). Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annals of Operations Research, 252*, 239–254.

Bellio, R., Ceschia, S., Di Gaspero, L., Schaerf, A., & Urli, T. (2016). Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *Computers & Operations Research, 65*, 83–92.

Bellio, R., Ceschia, S., Di Gaspero, L., & Schaerf, A. (2021). Two-stage multi-neighborhood simulated annealing for uncapacitated examination timetabling. *Computers and Operations Research, 132*, 105300.

Burke, E. K., & Bykov, Y. (2016). An adaptive flex-deluge approach to university exam timetabling. *INFORMS Journal on Computing, 28*(4), 781–794.

Burke, E. K., & Bykov, Y. (2017). The late acceptance hill-climbing heuristic. *European Journal of Operational Research, 258*, 70–78.

Burke, E. K., Qu, E., & Soghier, A. (2014). Adaptive selection of heuristics for improving exam timetables. *Annals of Operations Research, 218*, 129–145.

Bykov, Y., & Petrovic, S. (2016). A step counting hill climbing algorithm applied to university examination timetabling. *Journal of Scheduling, 19*, 479–492.

Carter, M. W. (1986). OR Practice—A survey of practical applications of examination timetabling algorithms. *Operations Research, 34*, 193–202.

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

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research, 39*, 1615–1624.

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2022). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research, 308*, 1–18.

Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling, 9*, 403–432.

Di Gaspero, L. (2002). Recolour, shake and kick: A recipe for the examination timetabling problem. In *Proc. 4th Int. Conf. on the Practice and Theory of Automated Timetabling, PATAT* (pp. 404–407).

Dimitsas, A., Nastos, V., Valouxis, C., Alefragis, P., & Gogos, C. (2022). A proven optimal result for a benchmark dataset of the uncapacitated examination timetabling problem. In *Proc. 13th Int. Conf. Pract. Theory Autom. Timetabling, PATAT* (vol. 3, pp. 30–46).

Fawcett, C., & Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics, 22*, 431–458.

Gogos, C., Goulas, G., Alefragis, P., Kolonias, V., & Housos, E. (2010). Distributed scatter search for the examination timetabling problem. In B. McCollum, E. K. Burke, & G. White (Eds.), *8th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2010)* (pp. 211–223). Belfast: PATAT.

Gogos, C., Alefragis, P., & Housos, E. (2012). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research, 194*, 203–221.

Hoos, H. H. (2012). Programming by optimization. *Communications of the ACM, 55*, 70–80.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning. *Operations Research, 37*, 865–892.

Laporte, G., & Desroches, S. (1984). Examination timetabling by computer. *Computers & Operations Research, 11*, 351–360.

Leite, N., Melício, F., & Rosa, A. C. (2019). A fast simulated annealing algorithm for the examination timetabling problem. *Expert Systems with Applications, 122*, 137–151.

Leite, N., Melício, F., & Rosa, A. C. (2021). A fast threshold acceptance algorithm for the examination timetabling problem. In *Handbook of operations research and management science in higher education* (pp. 323–363).

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43–58.

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009)* (pp. 424–434).

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

Muklason, A., Parkes, A. J., Özcan, E., McCollum, B., & McMullan, P. (2017). Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing, 55*, 302–318.

Müller, T. (2009). ITC2007 solver description: A hybrid approach. *Annals of Operations Research, 172*, 429–446.

Özcan, E., & Ersoy, E. (2005). Final exam scheduler—FES. *IEEE Congress on Evolutionary Computation, 2*, 1356–1363.

Parkes, A. J., & Özcan, E. (2010). Properties of Yeditepe examination timetabling benchmark instances. In *Proc. of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010)* (pp. 531–534).

Pillay, N. (2010). Evolving hyper-heuristics for a highly constrained examination timetabling problem. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT'10)* (pp. 336–346).

Rajah, C., & Pillay, N. (2023). An improved structure-based partial solution search for the examination timetabling problem. In R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. M. Zurada, & Rutkowski, L. (Eds.), *Artificial Intelligence and Soft Computing* (pp. 314–326). Springer.

Rosati, R. M., Petris, M., Di Gaspero, L., & Schaerf, A. (2022). Multi-neighborhood simulated annealing for the sports timetabling competition ITC2021. *Journal of Scheduling, 25*, 301–319.

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

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