Journal of Scheduling
https://doi.org/10.1007/s10951-022-00722-0

Bi-criteria simulated annealing for the curriculum-based course timetabling problem with robustness approximation

Can Akkan[1] · Ayla Gülcü[2] · Zeki Kuş[3]

Accepted: 24 January 2022

Abstract
In the process of developing a university's weekly course timetable, changes in the data, such as the available time periods of professors or rooms, render the timetable infeasible, requiring the administrators to repair or update the timetable. Since such changes almost always occur, it would be a sensible approach to identify a robust initial timetable, that is, one that can be repaired by making a limited number of changes, while still maintaining a high solution quality. This article formulates the problem as a bi-criteria optimization one, in which robustness is a stochastic objective, and the goal is to identify a good approximation to the Pareto frontier. It is assumed that multiple data changes, or disruptions, of multiple types can occur. The solution approach is a multi-objective simulated annealing (MOSA) algorithm, where a surrogate measure is used to approximate the robustness objective. Inspired by the concept of slack in machine and project scheduling, ten alternative measures of slack and a total of thirty surrogate measures are defined. Preliminary computational experiments are used to narrow the list of promising ones first to eight and then to two measures, which are then tested within a MOSA algorithm. Computational experiments show that one of these measures, when implemented in a multi-start MOSA algorithm, consistently provides the best Pareto frontier.

Keywords Course timetabling · Simulated annealing · Robustness · Fitness approximation

1 Introduction

As in most scheduling problems, the process of course timetabling has a dynamic characteristic. In a university where *pre-enrollment* timetabling is used, first, an initial timetable, S_0, is prepared based on a set of constraints, which is then announced to the university staff, giving them some time to submit requirement changes, due to new constraints or data corrections. We refer to such changes in constraints as *disruptions*. The timetable is then re-optimized taking these disruptions into account, while ensuring that the changes to S_0 are kept to a minimum. The resultant timetable is announced, and the students enroll in courses based on this timetable. Several types of disruptions are discussed in the literature (McCollum 2007; Müller et al. 2005; Kingston 2013; Phillips et al. 2017; Lindahl et al. 2019; Yasari et al. 2019; Lemos et al. 2020). Some disruptions could simply change feasibility of certain periods for some lectures, whereas others may affect either the availability or the capacity sufficiency of rooms for some lectures. In some cases, some new courses could be added, some others could be removed or canceled, some new faculty may arrive and others may leave, and some courses that are already in one curriculum (say economics) could be added to another one (say business).

Problems in which constraints change over time are known as dynamic optimization problems, which fall into the category of optimization in uncertain environments. Metaheuristics are often used for solving these problems (e.g., see Jin and Branke 2005 for a survey of evolutionary algorithms).

This work was supported by TÜBİTAK grant 217M475.

✉ Can Akkan
can.akkan@sabanciuniv.edu

Ayla Gülcü
ayla.gulcu@eng.bau.edu.tr

Zeki Kuş
zeki.kus@stu.fsm.edu.tr

1 Sabancı Business School, Sabancı University, Tuzla, Istanbul 34956, Turkey

2 Software Engineering Department, Faculty of Engineering and Natural Sciences, Bahçeşehir University, Istanbul, Turkey

3 Department of Computer Engineering, Fatih Sultan Mehmet University, Beyoglu, Istanbul 34445, Turkey

Published online: 06 April 2022

Recently, there has been increasing interest in modeling and solving dynamic combinatorial optimization problems. One such problem closely related to timetabling is the graph coloring problem. Hardy et al. (2018) develop heuristics for the dynamic graph coloring problem where edges are added/removed over time, randomly. They look into how information about the likelihood of future edge changes can be used to produce more robust colorings.

We say a timetable is *robust* if, when disrupted, its feasibility can be restored without significantly lowering its quality in terms of the objective function while keeping it relatively stable. Stability is measured by the number of events whose assigned periods are different in the disrupted and repaired solutions, i.e., the Hamming distance between the arrays storing these period assignments. In obtaining the repaired solution, we assume an upper limit, which is a function of the size of the disruption, is placed on this distance between the solutions.

It is important to distinguish the robustness to changes in data and constraints as described above, from robust optimization research in which uncertainty is defined by a range (or radius) around the value(s) taken by the decision variable(s). This is quite a sensible approach for engineering design optimization because the optimized characteristics of the product (i.e., the decision variables) could take values within a margin of these optimized values due to the inherent variability of the production processes. For instance, the thickness of the side walls of a car tire could deviate from the optimum design specifications and therefore, up to a certain level of this deviation, the performance of the tire should still be close to the optimum. Thus, for such problems, it is common to define robustness as the worst-case performance or standard deviation of the performance when the decision variables vary within a radius of their intended values. Often, meta-heuristics and algorithms that automatically design meta-heuristics are being used to solve such problems. A recent example of the latter is Hughes et al. (2021) for robust optimization of production plans in batch production, where the uncertainty is in the implementation of a given plan, which takes place within a radius around the intended plan. Again dealing with a problem where disturbances affect the values of the decision variables, Fei et al. (2019) use the archive sample approximation method which reuses previous function evaluations stored in an archive. They use the Wasserstein distance metric to approximate the possible benefit of a potential sample location on the estimation error and propose new sampling strategies based on this metric. Both Hughes et al. (2021) and Fei et al. (2019) test their algorithms on a set of artificial benchmark problems, which are continuous functions of one or more decision variables. Fei et al. (2019) also test their algorithm on a real-world, but simple, air foil design optimization problem.

We formulate the problem of identifying a robust timetable as a bi-criteria optimization problem where one objective is the quality of the solution measured as a function of the violated soft constraints (i.e., the penalty function), denoted by P, and the second one is a function that measures the robustness of the timetable, denoted by R. We express the robustness objective as minimizing $E(R(S, Y_S))$, the expected value of a disruption measure $R(S, Y_S)$, where S is a given solution and Y_S is the random variable representing the disruptions. We assume multiple number and types of disruptions can affect a given timetable, and there is no prior knowledge of the probability of any specific disruption occurring. (Thus, any disruption is equally likely.)

Recently, in the context of production planning, a benefit of formulating the optimization problem as a bi-criteria one (with one of the objectives measuring robustness) has been identified by Diaz et al. (2017). They looked into the interactions between the number of objectives, choices of sample size and robustness measure within the context of evolutionary robust optimization in production planning (formulated as a version of multi-dimensional knapsack problem). They report that including a robustness objective along with the primary objective of the problem (profit of the production plan) and finding a Pareto frontier yielded, on the average, better primary objective values than those found by a single objective approach.

Since there is no closed-form expression for $E(R(S, Y_S))$, one may consider using the sample average approximation (SAA) approach (Kleywegt et al. 2002) of $E(R(S, Y_S))$ to turn the stochastic optimization problem into a deterministic one by minimizing the sample average of $R(S, y)$ for $y \in \mathcal{Y}$, where $\mathcal{Y}$ is a random sample of disruptions. Gülcü and Akkan (2020) use this approach, where a sample of 20 disruptions are used within a multi-objective simulated annealing (MOSA) algorithm. However, for the version of the problem addressed here SAA is not practical. Unlike in Gülcü and Akkan (2020) where existence of multiple disruptions of a single type is assumed for each disruption scenario, here four different disruption types are assumed that would require a much larger sample of disruption scenarios for it to be representative enough. Their algorithm is a hybrid MOSA algorithm complemented with local search that is used to find good solutions for each disruption scenario in the sample. These solutions are maintained in a solution network through the MOSA algorithm. Thus, any significant increase in the size of the sample of disruption scenarios significantly increases the running time and the memory requirement of the hybrid MOSA.

Many practical robust optimization problems have a similar challenge: requirement of significant computational effort to evaluate the quality of a given solution. In their survey, Jin and Branke (2005) state that when the fitness function is very expensive to evaluate, or an analytical fitness function is not available, fitness functions are often approximated. They emphasize that the error made by this approximation could be systematic (i.e., with nonzero mean), and still it would perform well, as long as the relative ranking of the solutions is accurate. These approximate fitness functions, also known as meta-models or surrogates, are trained using a set of solutions with known (true) fitness values. The training may be off-line (i.e., before the search algorithm is run), online (i.e., during the search algorithm) or both. Algorithms with surrogate models mostly work with a budget limit on the number of (or CPU time used by) true fitness evaluations.

Most of the work on surrogate models, such as kriging, radial basis functions and polynomial regression, are for continuous unconstrained optimization problems (Jin 2011). A recent example using surrogate models for constrained combinatorial optimization is provided by Wang and Jin (2020), where authors use random forest (RF) models. An important characteristic of the problems they address is that both fitness and constraint satisfaction of the solutions require estimation. They determine that the performance of the RF models degenerates on high-dimensional combinatorial optimization problems. They argue that this weakness can be mitigated by dimension reduction techniques. Bartz-Beielstein and Zaefferer (2017) provide a discussion of strategies for dealing fitness estimation for discrete optimization problems. They argue that for permutation decision variables similarity-based methods, rather than surrogate models, can be used. For instance, in k-nearest neighbor (k-NN) models similarity measures are used to determine the "distance" between solutions and the fitness of a solution is estimated by using the fitness values of these neighbors (e.g., see Brownlee and Wright 2015).

An alternative to using surrogate models, which is taken here, is to define a domain-specific approximate fitness measure that can be calculated very quickly within a search heuristic and use it to evaluate all solutions throughout the search algorithm, as opposed to a fraction of these solutions.

In the work reported here, we first develop and test some robustness measures based on certain characteristics of slack in a given timetable and then implement the most promising ones in a MOSA algorithm. Figure 1 depicts a small example to provide some intuition regarding how slack could be related to robustness. We assume there are 3 courses (the first one with two events and the other with one) to be scheduled over four time periods with two rooms (r1 and r2). The time and room assignments of these events are marked with an "X." As an input to the timetabling problem, some periods are unavailable for some courses (typically due to teacher unavailability). Once a timetable is created, the available periods could have different characteristics. Some of the available periods can be fully committed (i.e., all rooms are busy at that period). If one wants to reschedule an event to an uncommitted period, the move may not be feasible due to some hard-constraint violations, often referred to as conflicts (e.g., the event's teacher may already be teaching another course at that period). Uncommitted and conflict-free periods are examples of slack, and clearly their numbers and distribution should have an effect on the ease of repairing the timetable. In timetable 1, only course 3 has a conflict-free period to which its event can be rescheduled, whereas in timetable 2, course 2 and course 3 have one conflict-free period each. In timetable 1, if we move the event of course 2 to period 4, we would have to reschedule the event of course 3 at period 4 to avoid the hard-constraint violation.

To the best of our knowledge, this is the first attempt to measure slack in a course timetable and use it as a surrogate measure of robustness or flexibility of the timetable. The specific timetabling problem we address is the curriculum-based university course timetabling problem (CB-CTP) of International Timetabling Competition, 2007 (ITC-2007).

The rest of the paper is organized as follows: In the next section, a formal definition of the problem is provided, including the ITC-2007 timetabling problem, the definitions of the types of data changes (so-called disruptions) and the robustness measure. Then, in Sect. 3 the measures of slack in the timetable are presented. Section 4 presents the multi-objective simulated annealing (MOSA) algorithm designed to work with a slack measure that is used as a surrogate measure of robustness. Section 5 presents the computational experiments done to identify the most promising slack measures among the ones explained in the previous section, and the results of the computational experiments run for testing the performance of the MOSA algorithm using the two most promising slack measures. Finally, in Sect. 6 some concluding remarks are shared. An appendix section presenting the IP model that is used in calculating the robustness measure is provided in the end.

2 Problem definition

2.1 The curriculum-based course timetabling problem

Among the variants of the university course timetabling problem, two main variants, namely the curriculum-based (CB-CTP) and post-enrolment (PE-CTP) course timetabling problems, have been widely studied (see Chen et al. 2021 for a recent survey). We have chosen to use the CB-CTP definition and instances developed for ITC-2007 (see McCollum et al. 2010), as these instances have become widely used benchmarking instances. In course timetabling, a solution is an assignment of a period (day and time slot) and a room to all weekly lectures of each course which satisfies all of the *hard constraints*, to arrive at a schedule which is repeated every week.

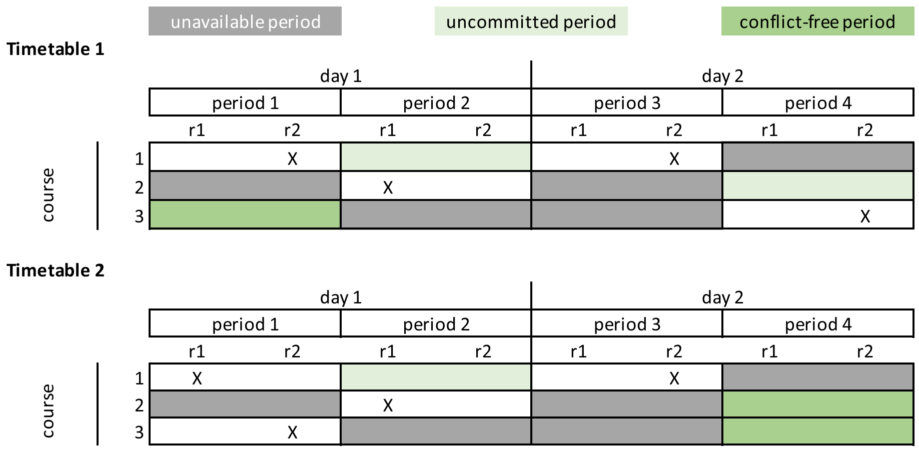

Fig. 1 Example depicting how characteristics of slack measures can be related to robustness

The hard constraints of the CB-CTP of ITC-2007 are *Lectures* (all lectures of a course must be scheduled to distinct periods), *Conflicts* (lectures of courses in the same curriculum or taught by the same teacher must be scheduled in different periods), *Availabilities* (if the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period) and *RoomOccupancy*. (Two lectures cannot take place in the same room in the same period.) There are also *soft constraints*, which can be violated. The soft constraints of CB-CTP of ITC-2007 are *RoomCapacity* (for each lecture, the number of students taking the course must be less than or equal to the number of seats of all the rooms that host its lectures), *MinimumWorkingDays* (the lectures of each course must be spread into the given minimum number of days), *CurriculumCompactness* (lectures belonging to a curriculum should be in consecutive periods) and *RoomStability*. (All lectures of a course should be given in the same room.)

The objective function, referred to as the *penalty* and denoted by P, is computed as the weighted sum of the violation of the soft constraints. Specifically, for the *RoomCapacity* constraint, each student above the capacity counts as 1 point of penalty. For the *MinimumWorkingDays* constraint, each day below the minimum counts as 5 points of penalty. For the *CurriculumCompactness* constraint, each isolated lecture in a curriculum counts as 2 points of penalty. Finally, for the *RoomStability* constraint, each distinct room used for the lectures of a course, but the first, counts as 1 point of penalty.

2.2 Disruption scenarios

In a typical pre-enrollment timetabling process, before the timetable is announced to the students for them to register to their course, it goes through two stages of optimization. First, an initial timetable is prepared based on a set of constraints provided by the professors and administrators. This timetable is announced to the staff, giving them some time to submit changes in constraints. The timetable is then re-optimized, and the students enroll in courses based on this timetable. As discussed in Phillips et al. (2017), many different types of changes in constraints are possible before enrollment, such as new courses being added, others being canceled; some faculty arriving or leaving; certain periods ceasing to be feasible for some professors; or capacity of some rooms becoming insufficient for some lectures due to an increase in the number of students.

Here, we use the disruption scenarios that have been first defined by Akkan et al. (2020). The disruptions affect the feasibility of the periods for lectures and availability or capacity sufficiency of the rooms. By assuming disruptions that affect the two main limited resources in timetabling, we believe we introduce sufficient variety and complexity. It is assumed that four types of disruptions make up a scenario, namely IP, CP, CS and RP disruptions.

The purpose of IP disruptions is to represent a situation when an instructor i learns she cannot teach at a period p to which one of her lectures is scheduled. Hence, this disruption type is specified by the tuple $\langle i, p\rangle$. For each disruption $\langle i, p\rangle$, unavailability constraints for all courses of instructor i at period p are added.

The purpose of CP disruptions is to represent a situation when an instructor learns he cannot teach during a consecutive set of periods and, to make up for this unavailability, offers a set of consecutive periods which he designated as unavailable for the initial timetabling. This disruption is specified by a tuple $\langle c, \mathcal{P}^-, \mathcal{P}^+\rangle$ for course c. Given the set of available periods for course c, $\mathcal{P}(c)$, $\mathcal{P}^- \subseteq \mathcal{P}(c)$ is a set of consecutive periods on the same day that become unavailable for course c and at least one of these periods is used by course c in the initial timetable, S_0. $\mathcal{P}^+ \subseteq \mathcal{P} \setminus \mathcal{P}(c)$ is a set of consecutive periods that become available for course c such that $|\mathcal{P}^+| \leq |\mathcal{P}^-|$.

In some universities, before students' official registration, trial registrations or surveys are carried out to judge demand for courses. CS disruption is introduced to represent a situation in which the initially planned capacity of a course is increased due to an estimated increase in demand. It is assumed that the planned number of students for a course is increased beyond the capacity of the room assigned to at least one lecture of that course. (Recall that room capacity is a soft constraint.) This disruption is specified by a tuple $\langle c, s\rangle$, where s is the new number of students for course c. All events of this course are included in the set of room-disrupted events, since, due to room stability constraints, it is quite likely to have most, if not all, events of the course to be scheduled at the same room. Even if some events of the course are currently assigned to a different room with enough capacity, adding all events of the course in this set results in a looser distance constraint on the repaired solution. Having a looser distance constraint leads to a larger solution space for the repaired solution. This is important because violation of the *RoomCapacity* constraint costs 1 point of penalty for *each* student above the capacity.

Classrooms are often used for purposes other than classes, such as seminars and faculty meetings. The purpose of an RP disruption is to represent such a situation in which a classroom becomes unavailable for scheduling classes for the duration of the semester. It is assumed that the availability of the room is lost for one or two consecutive periods on the same day. This disruption is specified by $\langle r, p, d\rangle$, where p is the first period that room r becomes unavailable, and d is the number of periods that become unavailable.

A set of disruptions of these types is referred to as a *disruption scenario*. All disruptions in a given disruption scenario are aggregated in two sets of disrupted lectures. Lectures e whose assigned periods in S_0 become infeasible due to IP and CP disruptions are denoted as $\mathcal{E}^P$ (the set of *disrupted-period* lectures) with size δ^p. Lectures e whose assigned rooms in S_0 become either infeasible due to RP disruptions or have insufficient capacity due to CS disruptions are denoted by $\mathcal{E}^R$ (the set of *disrupted-room* lectures) with size δ^r. Then, the set of disrupted lectures, $\mathcal{E}^D$, equals $\mathcal{E}^P \cup \mathcal{E}^R$ and the number disrupted lectures, δ, equals $|\mathcal{E}^D|$.

2.3 Robustness measure

As discussed in Sect. 1, the robustness objective is expressed as minimizing $E(R(S, Y_S))$, the expected value of a disruption measure $R(S, Y_S)$, where S is a given solution and Y_S is the random variable representing the disruptions. Since there is no closed-form expression for $E(R(S, Y_S))$, we estimate it by the sample average of $R(S, y_n)$ for $y_n \in \mathcal{Y}$, where $\mathcal{Y}$ is a random sample of disruption scenarios.

Let $\mathcal{F}(y_n)$ be the set of all solutions that are feasible with respect to a disruption scenario y_n and $D(S_0, S_1)$ be the Hamming distance between the arrays storing the assigned periods for all events of these two solutions. (Hence, $D(S_0, S_1)$ is equal to the number of lectures that are assigned to different periods in these two solutions.) Then, we define the following neighborhood set for a given solution S_0 and disruption scenario y_n with δ_n^p disrupted-period and δ_n^r disrupted-room lectures:

$$\mathcal{N}(S_0, y_n) = \{S : D(S_0, S) \le f(\delta_n^p, \delta_n^r);\ S \in \mathcal{F}(y_n)\} \tag{1}$$

Thus, if solution S_0 is disrupted by scenario y_n, then switching to any solution in $\mathcal{N}(S_0, y_n)$ would restore feasibility by rescheduling at most $f(\delta_n^p, \delta_n^r)$ lectures to a different period, where $f : (\mathbb{N}, \mathbb{N}) \to \mathbb{N}$. If there had been only period-disruptions, the *radius* $f(\delta_n^p, \delta_n^r)$ could be a multiple of δ^p, since every period-disrupted lecture must be moved to a different period. However, we also assume room-disruptions can occur, which may be repaired by moving a lecture to a different room without changing its period. Furthermore, the number of lectures affected by room-disruptions could be relatively large, as in the case of CS disruptions that may lead to the need for rescheduling all lectures of the corresponding course, some of which might be forced to a different period. So, given δ^p and δ^r, we define $f(\delta^p, \delta^r) = f^p\delta^p + f^r\delta^r,\ \ f^p > 1, f^r > 0$. For the computational results reported in Sect. 5, we set $f^p = 2$, $f^r = 0.25$. Then, we define the robustness measure $R(S_0, y_n)$ as:

$$R(S_0, y_n) = \min_{S \in \mathcal{N}(S_0, y_n)} \Phi_i(S, S_0), \text{ where} \tag{2}$$

$$\Phi_n(S, S_0) = P_{\text{ave}} \cdot 1_{D(S,S_0) > \delta_n^p} + (P(S) - P(S_0))^+ \tag{3}$$

where $x^+ := \max(0, x)$ and P_{ave} is the average per lecture penalty for a randomly generated sample of solutions, and calculated for each problem instance separately. The solution sample, denoted by $\mathcal{S}$, is the union of the initial populations (each comprised of 40 solutions) of 30 runs of the MOGA algorithm of Akkan and Gülcü (2018). Thus, $P_{\text{ave}} = (1/(|\mathcal{S}||\mathcal{L}|)\sum_{S\in\mathcal{S}} P(S)$, where $\mathcal{L}$ is the set of lectures (see Table 1). P_{ave} should be seen as a penalty term added so that solutions which only reschedule period-disrupted events to different periods are favored. Therefore, in addition to quality robustness measured by $(P(S) - P(S_0))^+$, $R(S_0, y_n)$ also incorporates a measure of solution robustness. Solution robustness is further ensured by the constraint $D(S_0, S) \le f(\delta_n^p, \delta_n^r)$ in defining $\mathcal{N}(S_0, y_n)$. If $\mathcal{N}(S_0, y_n) = \emptyset$, then $R(S_0, y_n)$ is set to a large value, B.

Finally, the sample average $\overline{R}(S, y) = (1/N)\sum_{n=1}^{N} R(S, y_n)$ is calculated as an estimate of $E(R(S, Y_S))$, given a set of randomly generated sample of disruption scenarios, $y = \{y_1, y_2, \ldots, y_N\}$, using a reasonably large N. For the computational results, N was set to 100 (after a convergence test). The following example depicts how $R(S, y_n)$ is calculated.

Example 1 We will assume a small timetabling problem in which we need to schedule 3 courses, over a two-day long "week," each day having 2 periods. For each course c, we are

Table 1 P_{ave} values used for each instance

ITC	P_{ave}	ITC	P_{ave}	ITC	P_{ave}	ITC	P_{ave}	Inst	P_{ave}
1	3.295	6	3.018	11	1.822	16	2.097	21	1.644
2	2.803	7	2.906	12	5.291	17	2.102		
3	1.783	8	1.570	13	2.049	18	2.105		
4	1.588	9	1.458	14	1.756	19	1.831		
5	9.614	10	1.969	15	1.806	20	4.738		

Table 2 Example 1: The timetabling instance

c	$\mathcal{E}(c)$	$\mathcal{P}(c)$	NS(c)	$T(c)$	CU(c)	MW(c)	r	$K(r)$
1	1, 2	1,2,3	25	tA	cA	2	r1	20
2	3	2,4	15	tB	cA	1	r2	40
3	4	1,4	30	tB	cB	1		

given the set of events $\mathcal{E}(c)$, the set of available periods $\mathcal{P}(c)$, the number of students NS(c), the teacher of the course $T(c)$, the curriculum it belongs to CU(c) and the minimum working days the course has to be scheduled in, MW(c) (see Table 2). This problem has a total of 80 feasible solutions. Each solution i is represented by the array storing the assigned period for event e, $A_i(e)$, for $e = 1, \ldots, 4$, and the room each event is assigned, $R_i(e)$.

The robustness of solution S_9, $A_9(e) = \{1, 3, 2, 4\}$ $R_9(e) = \{r1, r1, r2, r2\}$, with penalty $P_9 = 26$, is defined by the sets of disruptions of each disruption type. The set of IP disruptions for S_9 is $\{\langle 1, 1\rangle, \langle 1, 3\rangle, \langle 2, 2\rangle, \langle 3, 4\rangle\}$. The set of CP disruptions is $\{\langle 1, \{1\}, \{4\}\rangle, \langle 1, \{1, 2\}, \{4\}\rangle, \langle 1, \{3\}, \{4\}\rangle, \langle 2, \{2\}, \{1\}\rangle, \langle 2, \{2\}, \{3\}\rangle, \langle 3, \{4\}, \{1\}\rangle\}$. The set of RP disruptions is $\{\langle r1, 1, 1\rangle, \langle r1, 2, 1\rangle, \langle r1, 1, 2\rangle, \langle r1, 3, 1\rangle, \langle r1, 4, 1\rangle, \langle r1, 3, 2\rangle, \langle r2, 1, 1\rangle, \langle r2, 2, 1\rangle, \langle r2, 1, 2\rangle, \langle r2, 3, 1\rangle, \langle r2, 4, 1\rangle, \langle r2, 3, 2\rangle\}$. Finally, the set of CS disruptions is $\{\langle 1, 21\rangle, \langle 1, 22\rangle, \ldots, \langle 1, 40\rangle\}$, since courses whose first lecture is not assigned to the largest capacity room are exposed to this disruption (see Sect. 5.1). Disruption scenarios are formed by combinations of these disruptions that result in at least 3 disruptions such that at most 1 disruption of type RP and at most 2 disruptions of each of the other types occur.

Of course, even for this small example there would be a very large number of possible disruption scenarios, so here we demonstrate the robustness calculation for a single scenario, say y_1, comprised of IP disruption $\langle 3, 4\rangle$, CS disruption $\langle 1, 33\rangle$, and RP disruption $\langle r1, 2, 1\rangle$. Thus, $\delta_1^p = 1$ and $\delta_1^r = 2$, and assuming $f^p = 2$ and $f^r = 0.5$, we get $f(\delta_i^p, \delta_i^r)$ to be 2. Of the 80 feasible solutions for the problem at hand, $\mathcal{N}(S_0, y_1)$ has the 16 solutions listed in Table 3, where $K^+(e)$ denotes the room capacity violation of event e, $W(c)$ is the number of work days for course c. For this example, we set $P_{\text{ave}} = (1/80)\sum_i P(S_i) = 19.5$, using all 80 solutions, rather than a sample of solutions. Then, $\Phi_1(S_i, S_0)$ are calculated as shown in Table 3, which results in $R(S_9, y_1) = 0$.

For the computational experiments, the calculation of $R(S, y_n)$ is done by solving the integer programming (IP) model presented in Appendix. Gurobi is used as the solver, and if it fails to find a feasible solution, $R(S, y_n)$ is set to B, as discussed above, and computational experiments revealed that for the problem instances solved here $B = 1200$ is a sufficiently large penalty.

3 Slack-based surrogate measures

The idea of using some form of slack as a surrogate measure of robustness of schedules is not new. Some of the early examples of research done in this vein are Leon et al. (1994) for job shop scheduling with disruptions in the form of machine failures; Lambrechts et al. (2008) for the resource-constrained project scheduling problem where resource breakdown disruptions occur and one measure of slack is resource slack and the other is inserted time buffers; Hazır et al. (2010) for the discrete time–cost trade-off problem in project scheduling; and Vansteenwegen and Oudheusden (2006) for train scheduling with the goal of minimizing the propagation of delays. In all of these problems, there is a well-established concept of resource slack which can be created by the planner, such as unused time on a machine or a limited resource (such as man-hours or budget) available in a given period that is in excess of the requirement.

To the best of our knowledge, the idea of slack has not been studied for course timetables in the context of creating robust timetables. The two limited resources of a course timetable are available time periods and classrooms available at each period. We first define different measures of slack and, then given a slack measure, use a summary statistic of that measure as an estimator of robustness of the given timetable. The notation that is used to define the underlying timetabling problem, the disruptions and a given solution are summarized in Table 4. The example instance whose data are provided in Tables 5 and 6 is used to demonstrate the calculations of the robustness measures.

The defined slack measures can be classified into three groups. The first group provides measures of slack for each

Table 3 Example 1: Calculation of R for a disruption scenario

$(A_i(e), R_i(e))$				$K^+(e)$				$W(1)$	$P(S_i)$	$D(S_i, S_9)$	$\Phi_1(S_i, S_9)$
(1, r1)	(3, r1)	(2, r2)	(1, r1)	13	13	0	10	2	36	1	10
(1, r1)	(3, r2)	(2, r2)	(1, r1)	13	0	0	10	2	24	1	0
(1, r1)	(3, r1)	(2, r1)	(1, r1)	13	13	0	10	2	36	1	10
(1, r1)	(3, r2)	(2, r1)	(1, r1)	13	0	0	10	2	24	1	0
(1, r1)	(3, r1)	(2, r2)	(1, r2)	13	13	0	0	2	26	1	0
(1, r1)	(3, r2)	(2, r2)	(1, r2)	13	0	0	0	2	14	1	0
(1, r1)	(3, r1)	(2, r1)	(1, r2)	13	13	0	0	2	26	1	0
(1, r1)	(3, r2)	(2, r1)	(1, r2)	13	0	0	0	2	14	1	0
(1, r1)	(3, r1)	(4, r2)	(1, r1)	13	13	0	10	2	36	2	29.5
(1, r1)	(3, r2)	(4, r2)	(1, r1)	13	0	0	10	2	24	2	19.5
(1, r1)	(3, r1)	(4, r1)	(1, r1)	13	13	0	10	2	36	2	29.5
(1, r1)	(3, r2)	(4, r1)	(1, r1)	13	0	0	10	2	24	2	19.5
(1, r1)	(3, r1)	(4, r2)	(1, r2)	13	13	0	0	2	26	2	19.5
(1, r1)	(3, r2)	(4, r2)	(1, r2)	13	0	0	0	2	14	2	19.5
(1, r1)	(3, r1)	(4, r1)	(1, r2)	13	13	0	0	2	26	2	19.5
(1, r1)	(3, r2)	(4, r1)	(1, r2)	13	0	0	0	2	14	2	19.5

period. Let $X(p, r)$ be equal to 1 if room r is used at period p by some lecture, 0 otherwise. Then, assuming the rooms are indexed in increasing capacity, define $\mathbf{RSA}[p]$, room-based slack at period p, as

$$\mathbf{RSA}[p] = \begin{cases} \frac{1}{\text{Rm}-\rho(p)} \sum_r \sum_{q>r} 1_{(X(p,r)=1, X(p,q)=0)} & \text{if Rm} > \rho(p) \\ \text{Rm} - 1 & \text{if Rm} = \rho(p) \end{cases} \tag{4}$$

Let's denote the term $\sum_{q>r} 1_{([X(p,r)=1, X(p,q)=0)}$ in Eqn. 4 as $\rho(p, r)$. $\rho(p, r)$ gives for every room used at a period the number of unused larger capacity rooms in the same period. If $\rho(p) = \text{Rm}$, we set $\mathbf{RSA}[p] = \text{Rm} - 1$ because that is the largest possible value for $\mathbf{RSA}[p]$ if $\text{Rm} > \rho(p)$. It is quite reasonable to have decreasing marginal benefit in increasing $\rho(p, r)$, so an alternative slack measure could make use of an exponential utility function as $\mathbf{RSU}[p] = \frac{1}{\text{Rm}-\rho(p)} \sum_r \sum_{j=1}^{\rho(p,r)} e^{-j}$ for a given period p. Alternatively, we can assume there is utility in having at least one larger capacity room for a lecture scheduled at a given period, say at (p, r). In that case, we would have $\rho(p, r) > 0$. Then, another slack measure can be defined as $\mathbf{RSB}[p] = \frac{1}{\text{Rm}-\rho(p)} \sum_r 1_{\rho(p,r)>0}$ for a given period p.

The second group of estimators make use of slack measured for each room. For room r, letting $\pi(r) = \sum_t 1_{X[t,r]=0}$ denote the number of unused periods in the same room, we define $\mathbf{PSU}[r] = \sum_{j=1}^{\pi(r)} e^{-j}$ as a slack measure for a given room r. The next measure is a more finely grained version of $\mathbf{PSU}[r]$, where the periods are subdivided into daily sets. Let the day of a given period p be denoted by $D(p)$. Then, $\xi(d, r) = \sum_{p:D(p)=d} 1_{X[p,r]=0}$ represents the number of unused periods on day d at room r, and we define $\mathbf{DSU}[d, r] = \sum_{j=1}^{\xi(d,r)} e^{-j}$ as a slack measure for a given room r on day d. Then, $\mathbf{DSU}[r]$ is simply obtained by concatenating the arrays $\mathbf{DSU}[d, r]$ for all d. Table 7 summarizes the calculation of these room-indexed slack measures.

The third group of measures use course-specific availability of periods. We first let,

$$\begin{aligned} \mathcal{Y}(p) &= \text{ the set of courses scheduled at period } p. \\ K^{\max}(p) &= \max_{r \in \mathcal{FR}(p)} \{K(r)\} \\ \mathcal{C}(c) &= \text{ the set of conflicting courses for course } c \end{aligned}$$

Note that courses in the same curriculum or taught by the same teacher are referred to as conflicting courses. We then define the following sets,

$$\begin{aligned} \mathcal{UP}(c) &= \{p : p \in \mathcal{P}(c), \rho(p) > 0\} \\ \mathcal{UP}^+(c) &= \{p : p \in \mathcal{UP}(c), K^{\max}(p) > NS(c)\} \\ \mathcal{CP}(c) &= \{p : p \in \mathcal{UP}(c), \mathcal{Y}(p) \cap \{\mathcal{C}(c) \cup c\} = \emptyset\} \\ \mathcal{CP}^+(c) &= \{p : p \in \mathcal{UP}^+(c), \mathcal{Y}(p) \cap \{\mathcal{C}(c) \cup c\} = \emptyset\} \end{aligned}$$

Thus, $\mathcal{UP}(c)$ is the set of uncommitted periods, which is the set of available periods with free rooms for course c, and $\mathcal{UP}^+(c)$ gives the set of uncommitted periods for course c, having at least one free room with sufficient capacity. On the other hand, $\mathcal{CP}(c)$ gives the set of conflict-free periods among uncommitted periods for course c and $\mathcal{CP}^+(c)$ gives the set of conflict-free and uncommitted periods for course

Table 4 Notation for problems, disruptions and solutions

$\mathcal{E}(c)$	: the set of events for course c
$E(c)$	$= \lvert\mathcal{E}(c)\rvert$
$\mathcal{P}(c)$	: the set of available periods for course c
$\mathcal{P}$	: the set of all available periods
$\text{NS}(c)$	: number of students for course c
$T(c)$	: teacher of course c
$\text{CU}(c)$	: the curriculum course c belongs to
$\text{MW}(c)$	: the minimum working days course c has to be scheduled in
$K(r)$	: capacity of room r
$K^{\max}$	: maximum room capacity
$D(p)$	: day of period p
D	: the number of days
Rm	: the number of rooms
$\mathcal{E}^P$	: The set of *disrupted-period* lectures; caused by IP and CP disruptions
$\mathcal{E}^R$	: The set of *disrupted-room* lectures; caused by RP and CS disruptions
$\mathcal{E}^D$	$= \mathcal{E}^P \cup \mathcal{E}^R$, the set of disrupted lectures
δ^p	$= \lvert\mathcal{E}^P\rvert$
δ^r	$= \lvert\mathcal{E}^R\rvert$
δ	$= \lvert\mathcal{E}^D\rvert$
$P(S)$	: penalty of solution S
A_i	: array of periods to which each event of solution S_i is assigned
R_i	: array of rooms to which each event of solution S_i is assigned
$D(S_i, S_j)$	: Hamming distance between arrays A_i and A_j
$f(\delta^p, \delta^r)$	: maximum allowed distance between a disrupted and a repaired solution
$R(S, y_n)$	: robustness of solution S for a disruption scenario y_n
$\overline{R}(S, y)$	$= (1/N)\sum_{n=1}^{N} R(S, y_n)$, estimated robustness of solution S
$X(p, r)$	: 1 if room r is used at period p by some event, 0 otherwise.
$\mathcal{FR}(p)$	: the set of unused (free) rooms in period p
$\rho(p)$	: the number of unused rooms in period p, $\lvert\mathcal{FR}(p)\rvert$
$\rho(p, r)$	: for room r used at period p, the number of unused larger capacity rooms in period p.
$\pi(r)$	: the number of unused periods in room, r
$\xi(d, r)$	: the number of unused periods on day d at room r

c, having at least one free room with sufficient capacity. Given these sets, we can define two arrays, $\mathbf{C}[c] = |\mathcal{CP}(c)|$ and $\mathbf{R}[c] = |\mathcal{CP}^+(c)|$. Rather than simply counting the number of periods in these sets, an alternative is to calculate an exponential utility function that gives decreasing marginal utility with increasing number of periods in these sets. These two arrays are defined as $\mathbf{UC}[c] = \sum_{j=1}^{|\mathcal{CP}(c)|} e^{-j}$, and $\mathbf{UR}[c] = \sum_{j=1}^{|\mathcal{CP}^+(c)|} e^{-j}$. Furthermore, one can argue that the value of $\mathbf{C}[c]$ depends on the number of events of course c, $E(c)$, so we defined an additional array $\mathbf{CL}[c]$, as $\mathbf{CL}[c] = \mathbf{C}[c]/E(c)$. Table 8 depicts the calculation of the measures that use the course-specific availability of periods.

Given these ten slack measuring arrays, whose definitions are summarized in Table 9, we calculate the average, standard deviation and the coefficient of variation (standard deviation divided by average) of each array as surrogate measures of the robustness of the given timetable. These three summary statistics for a given slack measure $\mathbf{L}$ are denoted as $\overline{L}$, SD_L, CV_L, respectively. Variability or the distribution of any slack measure across time, rooms or courses could potentially have a significant impact on the robustness of the timetable. For instance, if only a small number of courses have most of the conflict-free periods, the standard deviation of the values in array $\mathbf{C}[c]$ would be large. In that case, the likelihood of repairing the timetable with minimum increase in penalty would be low, since the disruption is more likely to affect a course with no or few conflict-free periods, which could require a costly repair. On the other hand, if the conflict-free periods are evenly distributed, a domino effect due to rescheduling events is less likely to happen, and furthermore, it is easier to find a low penalty repair due to having a larger

Table 5 Example 2: Course-related data

c	$\lvert\mathcal{E}(c)\rvert$	$\mathcal{P}(c)$	NS(c)	CU(c)	$T(c)$
1	3	{1, 2, 4, 6, 7, 8, 9}	26	A	tA
2	3	{3, 4, 5, 6, 10, 11, 12}	30	A	tC
3	2	{1, 2, 4, 7, 8, 9, 11, 12}	28	B	tF
4	3	{1, 4, 5, 6, 8, 9}	25	E	tB
5	3	{2, 3, 4, 5, 6, 7, 9, 11, 12}	30	B	tD
6	4	{1, 2, 3, 6, 7, 10}	46	E	tB
7	4	{2, 3, 9, 12}	70	H	tA
8	3	{1, 2, 9, 10}	20	G	tC
9	4	{1, 4, 5, 6, 7, 8, 9, 10, 11, 12}	32	F	tD
10	2	{1, 2, 4, 5, 6, 7, 8, 9}	70	E	tB
11	4	{1, 2, 3, 4, 5, 6, 8, 9, 10, 11}	45	C	tE
12	3	{2, 3, 6, 8, 11, 12}	20	F	tD
13	4	{3, 5, 6, 7, 9, 10}	28	G	tF
14	4	{1, 4, 5, 6, 11, 12}	35	D	tG
15	3	{1, 3, 4, 9, 11}	40	C	tE
16	2	{1, 6, 7, 8, 9, 10, 12}	70	H	tC
17	3	{3, 4, 8, 9, 11}	40	B	tF
18	3	{2, 3, 5, 6, 7, 8, 9}	45	D	tG

Table 6 Example 2: Room capacity data

r	1	2	3	4	5	6	7	8
$K(r)$	30	30	30	50	50	50	80	100

number of alternative periods to move the effected event(s). CV could prove to be a good measure since, in addition to low variability of slack, high slack quantities are likely to produce robust timetables. Thus, robust timetables are likely to have a small CV for slack. Using CV of slack as a measure of robustness is also investigated in Hazır et al. (2010) for robust project scheduling.

4 Multi-objective simulated annealing (MOSA) algorithm

The main characteristic feature of the SA algorithms is the probabilistic acceptance rule which is designed to occasionally accept solutions with inferior objective value, and as the iterations increase, this probability of accepting inferior moves declines exponentially using a parameter referred to as the temperature. (For a survey of MOSA and SA algorithms, see Suman and Kumar 2006.) This design is built on the premise that the objective function value is a deterministic function of the decision variables with no noise or error. Since in our implementation of MOSA the two objectives are penalty P and a surrogate measure of the robustness measure $\overline{R}$, it is necessary to take into account the approximate nature of the second objective. Before we go into how this challenge is addressed, we provide a brief overview of the design of the acceptance rule and the maintenance of an archive of solutions that yield an approximation to the optimal Pareto frontier in MOSA algorithms.

In one of the earliest studies of MOSA algorithms, Serafini (1994) presents nine alternative rules for the computation of the probability of accepting a new solution based on scalar ordering, Pareto ordering and cone ordering approaches. Using a scalarizing function (the weighted sum of objectives) that aggregates the multi-criteria information into a single criterion is quite common (e.g., Ulungu et al. 1999 and Czyzk and Jaszkiewicz 1997).

Suppapitnarm et al. (2000) present a MOSA algorithm that introduces a new acceptance probability formulation based on an annealing schedule with multiple temperatures (one for each objective). The algorithm also periodically executes a “return-to-base” option which restarts the search using one of the non-dominated solutions selected randomly from the archive.

Suman (2003) proposes a Pareto domination-based MOSA (PDMOSA) algorithm which uses the strength Pareto fitness assignment procedure described in Zitzler and Thiele (1999) to compute the acceptance probability. In this approach, a given solution has better fitness if it is dominated by fewer solutions in a set of potentially Pareto optimal solutions. An improved version of PDMOSA in which a self-stopping criterion has been introduced can be found in Suman (2005). The idea of a dominance-based approach is also used in Smith et al. (2008). Bandyopadhyay et al. (2008) propose another MOSA algorithm that uses an archive of solutions and dominance-based acceptance probabilities, which they call Archive-based Multi-objective Simulated Annealing (AMOSA). They take into account the domination status of the new solution with the current solution, as well as those in the archive and also use a measure of the amount of domination between two solutions. The algorithm has a novel archive management approach, where the archive size is managed with two limits, HL and SL, and when the archive size reaches SL, a clustering algorithm reduces it down to HL. The final Pareto frontier produced by the algorithm has at most HL solutions.

All the MOSA literature discussed above assumes the objective functions can be evaluated quickly, which is clearly not the case for the robustness objective here. In a survey of fitness approximation in evolutionary computation, Jin (2005) discusses four different levels of fitness approximation, one of which is the functional approximation, where an explicit alternative objective function is used. Authors point out that it is quite difficult to design an approximate model that converges to the true optimum. One possible solution to alleviate this weakness is to use both models, the approximate function and the original function. Authors state that since the surrogate measures in general have biased error, their accuracy cannot be improved by resampling the approximate fitness function. They suggest addressing the issue of error by using the true fitness function instead of the approximation for some of the solutions. In that case, the critical decision is finding the right balance between cheap but erroneous approximate fitness evaluations and expensive but accurate true fitness evaluations.

Since it is computationally very expensive to calculate $\bar{R}(S, y)$ using a reasonably large sample of disruption scenarios, we opted for calculating $\bar{R}(S, y)$ only for the solutions in the final archive and using a surrogate measure throughout the MOSA algorithm. All surrogate measures proposed here

Table 7 Example 2: Room-indexed slack measures

r	$\pi(r)$	PSU(r)	$\xi(., r)$	DSU$(1, r)$	DSU$(2, r)$	DSU$(3, r)$
1	3	0.553	$[1, 0, 2]^T$	0.368	0	0.503
2	5	0.578	$[0, 1, 4]^T$	0	0.368	0.571
3	4	0.571	$[0, 1, 3]^T$	0	0.368	0.553
4	6	0.580	$[0, 3, 3]^T$	0	0.553	0.553
5	4	0.571	$[0, 1, 3]^T$	0	0.368	0.553
6	5	0.578	$[1, 0, 4]^T$	0.368	0	0.571
7	5	0.578	$[1, 1, 3]^T$	0.368	0.368	0.553
8	7	0.581	$[2, 3, 2]^T$	0.503	0.553	0.503

Table 8 Example 2: Course-indexed slack measures

c	$\mathcal{UP}(c)$	$\mathcal{CP}(c)$	$\mathcal{CP}^+(c)$	**C**$[c]$	**R**$[c]$	**CU**$[c]$	**RU**$[c]$
1	{1, 2, 4, 6, 7, 8, 9}	{6}	{6}	1	1	0.368	0.368
2	{3, 4, 5, 6, 10, 11, 12}	{11, 12}	{11, 12}	2	2	0.503	0.503
3	{1, 2, 4, 7, 8, 9, 11, 12}	{11, 12}	{11, 12}	2	2	0.503	0.503
4	{1, 4, 5, 6, 8, 9}	{}	{}	0	0	0	0
5	{2, 3, 4, 5, 6, 7, 9, 11, 12}	{12}	{12}	1	1	0.368	0.368
6	{1, 2, 3, 6, 7, 10}	{10}	{10}	1	1	0.368	0.368
7	{2, 3, 9, 12}	{}	{}	0	0	0	0
8	{1, 2, 9, 10}	{}	{}	0	0	0	0
9	{1, 4, 5, 6, 7, 8, 9, 10, 11, 12}	{10, 12}	{10, 12}	2	2	0.503	0.503
10	{1, 2, 4, 5, 6, 7, 8, 9}	{}	{}	0	0	0	0
11	{1, 2, 3, 4, 5, 6, 8, 9, 10, 11}	{8, 10, 11}	{8, 10, 11}	3	3	0.553	0.553
12	{2, 3, 6, 8, 11, 12}	{12}	{12}	1	1	0.368	0.368
13	{3, 5, 6, 7, 9, 10}	{}	{}	0	0	0	0
14	{1, 4, 5, 6, 11, 12}	{11, 12}	{11, 12}	2	2	0.503	0.503
15	{1, 3, 4, 9, 11}	{11}	{11}	1	1	0.368	0.368
16	{1, 6, 7, 8, 9, 10, 12}	{7}	{7}	1	1	0.368	0.368
17	{3, 4, 8, 9, 11}	{11}	{11}	1	1	0.368	0.368
18	{2, 3, 5, 6, 7, 8, 9}	{8, 9}	{8, 9}	2	2	0.503	0.503

Table 9 Definitions of slack measuring arrays

RSA$[p]$	$: \begin{cases} \frac{1}{\mathrm{Rm}-\rho(p)} \sum_r \sum_{q>r} 1_{(X(p,r)=1, X(p,q)=0)} & \text{if } \mathrm{Rm} > \rho(p) \\ \mathrm{Rm}-1 & \text{if } \mathrm{Rm} = \rho(p) \end{cases}$
RSU$[p]$	$= \frac{1}{\mathrm{Rm}-\rho(p)} \sum_r \sum_{j=1}^{\rho(p,r)} e^{-j}$
RSB$[p]$	$= \frac{1}{\mathrm{Rm}-\rho(p)} \sum_r 1_{\rho(p,r)>0}$
PSU$[r]$	$= \sum_{j=1}^{\pi(r)} e^{-j}$
DSU$[r]$	$= [$**DSU**$[1,r] \ \ldots$ **DSU**$[d,r] \ \ldots$ **DSU**$[D,r]]$ where **DSU**$[d,r] = \sum_{j=1}^{\xi(d,r)} e^{-j}$
C$[c]$	$= \lvert\mathcal{CP}(c)\rvert$
R$[c]$	$= \lvert\mathcal{CP}^{+}(c)\rvert$
UC$[c]$	$= \sum_{j=1}^{\lvert\mathcal{CP}(c)\rvert} e^{-j}$
UR$[c]$	$= \sum_{j=1}^{\lvert\mathcal{CP}^{+}(c)\rvert} e^{-j}$
CL$[c]$	$=$ **C**$[c]/E(c)$

are approximate measures rather than unbiased estimators with random noise, and this should work fine since what is needed is to obtain a relative rank of the solutions, as discussed next.

At each iteration of MOSA algorithm, a new solution S' is compared to the current solution S taking into account both the penalty and the given surrogate measure, M, as the robustness objective. For computing the acceptance probability, we use Smith's Pareto dominance rule (Smith et al. 2008), which works as follows. Let $\mathcal{A}$ denote the current potentially Pareto optimal set, or archive. Then, we set $\widetilde{\mathcal{A}} = \mathcal{A} \cup S \cup S'$. Let $r^M(S)$ be the domination count of solution S in $\widetilde{\mathcal{A}}$ (i.e., the number of solutions in $\widetilde{\mathcal{A}}$ dominating solution S plus 1) when surrogate measure M is used. So, if S is not dominated by any solution in $\widetilde{\mathcal{A}}$, $r^M(S)$ equals 1. Smith's Pareto dominance rule calculates the acceptance probability as $\min\{1, \exp(-\Delta/T_{cur})\}$, where $\Delta = (|r^M(S')| - |r^M(S)|)/|\widetilde{\mathcal{A}}|$. Due to the approximate nature of M, a solution S with $r^M(S) = 1$ based on (P, M) could have a larger $r(S)$ based on $(P, \overline{R})$. Thus, our MOSA algorithm uses a cutoff K on $r^M(S)$ in making the acceptance and archiving decisions for the generated solutions. As stated above, at the end of the MOSA algorithm, $\overline{R}(S, y)$ is calculated for all $S \in \mathcal{A}$ to obtain a good approximation to the Pareto frontier defined by objectives $(P, \overline{R})$ and the performance comparison of the surrogate measures is done based on the Pareto frontier defined by $(P, \bar{R})$.

The pseudo-code of the MOSA algorithm is given in Algorithm 1, where acceptance and archiving decisions are made in $ChooseNextandUpdate()$. Algorithm 2 provides the pseudo-code of $ChooseNextandUpdate()$ function. Note that since a given surrogate measure is used in the implementation of MOSA, previously used notation, $r^M(S)$ is simplified as $r(S)$. There are two checks carried out for the new solution S': First, $r(S')$ is compared with $r(S)$, and then, $r(S')$ is compared with the cutoff K, resulting in four possible cases. When $r(S') \leq K$, archive $\mathcal{A}$ is updated by

Algorithm 1: MOSA Algorithm

```
1  {$T_f$, $cr$, $nbr\_iter$, $p_{acc}$} ← set_params()
2  {$T_0$, $nbr\_out$, $nbr\_inn$} ←
     compute_params($T_f$, $cr$, $nbr\_iter$, $p_{acc}$)
3  $initialize(S)$
4  $\mathcal{A} \leftarrow \{S\}$, $T_{cur} \leftarrow T_0$
5  for $o\_counter \leftarrow 0$ to $nbr\_out$ do
6      for $i\_counter \leftarrow 0$ to $nbr\_inn$ do
7          $S' \leftarrow RandomFeasMove(S)$
8          $S \leftarrow ChooseNextandUpdate(S, S', \mathcal{A}, T_{cur})$
9      end
10     $T_{cur} \leftarrow T_{cur} * cr$
11 end
```

inserting S' (cases 1 and 3); otherwise, the archive remains unchanged. The solution from which the next local move ($RandomFeasMove()$) is made, is determined differently for each of the four cases. In case 3, S' becomes that solution because its rank is better than $r(S)$ and it is good enough to enter the archive. In case 4, since $r(S') > K$, a solution, a^*, is selected randomly from the archive and the choice is made between a^* and S'. When $r(S)$ is better than $r(S')$, the choice is made between S and S' (cases 1 and 2). When inserting a new solution into the archive, domination count of all solutions in the archive is updated, and any solution, S, with $r(S) > K$ is removed from the archive.

Table 10 summarizes the first 10 iterations of MOSA for a simple example, where it is assumed that smaller M values are associated with more robust solutions. Solutions are represented with their (P, M) values. Figure 2 shows the archive for the iterations in which the archive is updated. It is assumed that $(1, 7)$ is the initial randomly generated solution. In iteration 6, when solution $(7, 6)$ is generated and inserted into the archive, $r(s)$ for solution $(8, 8)$ exceeds the cutoff $K = 5$ and it leaves the archive. In iteration 7, S' is $(9, 10)$ with $r(s)$ equal to 6, and thus, case 2 applies. Let's assume $(9, 10)$ is selected to be the next iteration's S. Then, S' of iter-

Algorithm 2: *ChooseNextandUpdate*

```
Input: $S$ (current solution), $S'$ (new solution), $\mathcal{A}$
1  if $r(S) < r(S')$ then
2      if $r(S') \le K$ then
           // Case 1: worse domination count of
           new but still within cutoff
3          updateArchive($S'$, $\mathcal{A}$)
4          $p_{acc} \longleftarrow$computeAcceptPr($S$, $S'$)
5          if $rnd01 < p_{acc}$ then $S \longleftarrow S'$
6      end
7      else
           // Case 2: worse domination count of
           new and above cutoff
8          $p_{acc} \longleftarrow$computeAcceptPr($S$, $S'$ )
9          if $rnd01 < p_{acc}$ then $S \longleftarrow S'$
10     end
11 end
12 else
13     if $r(S') \le K$ then
           // Case 3: same or better domination
           count of new and within cutoff
14         updateArchive($S'$, $\mathcal{A}$)
15         $S \longleftarrow S'$
16     end
17     else
           // Case 4: same or better domination
           count of new but above cutoff
18         $a^* \longleftarrow$ selectRandomArchiveSol($\mathcal{A}$)
19         $p_{acc} \longleftarrow$computeAcceptPr($a^*$, $S'$)
20         if $rnd01 < p_{acc}$ then $S \longleftarrow S'$
21         else $S \longleftarrow a^*$
22     end
23 end
24 return $S$
```

ation 8 is (8, 9) with rank 6, and case 4 applies. We assume a^* selected from the archive is (4, 3), and it is the one chosen to be S for the next iteration. In iteration 9, solution (3, 4) is generated and is added to the archive, since case 3 applies in that iteration. Finally, in iteration 10, S' is generated to have (5, 6), which results in case 1 to be applied with (5, 6) moving to the archive.

5 Computational experiments

For the computational experiments, we selected five ITC-2007 instances that are the most constrained (thus potentially difficult) instances in terms of conflict intensity, teacher availability and room occupancy (Bonutti et al. 2012). Specifically, ITC5 and ITC12 are the timetabling instances with the highest conflict intensity; ITC2 and ITC5 are the top two in terms of lowest teacher availability; and finally, ITC1 and ITC7 have the highest room occupancy.

The experiments were done in two stages in which the results of the first stage were used to narrow down the large set of surrogate measures to a short list of promising ones. Then, in the second stage, two of the most promising surrogate measures were implemented in the MOSA algorithm and their performances were evaluated.

As discussed in Sect. 2.3, the calculation of $\overline{R}(S, y)$ requires a randomly generated sample of disruption scenarios, $y = \{y_1, y_2, \ldots, y_N\}$ for each solution S. Hence, below, we first discuss how the disruption scenarios were generated. Then, in Sect. 5.2 we discuss how the solutions that were used in stage one of the analysis were selected. Stage one comprised of a correlation analysis between the surrogate measures and $\overline{R}$ (Sect. 5.3) and an accuracy analysis for the archiving rule based on $r(s)$ and K (Sect. 5.4), which assesses the likelihood of an archived solution being on the Pareto frontier defined by $(P, \overline{R})$. Finally, the results of the second stage are reported in Sect. 5.5, assessing the quality of the approximation frontiers found by the MOSA algorithm using two of the most promising surrogate measures.

5.1 Generating the disruption scenarios

Recall that a disruption scenario is comprised of a combination of four types of disruptions: IP, CP, RP, and CS. For a disruption scenario, let N_{ip}, N_{cp}, N_{rp} and N_{cs} denote the number of IP, CP, RP and CS disruptions, respectively. In creating a scenario, N_{rp} is drawn from $DU(0, 1)$, while the others are each drawn from $DU(0, 2)$ so that the total number of disruptions in the disruption scenario is at least 3.

For each IP disruption ($\langle i, p \rangle$), first an instructor i is chosen randomly. Given i, first a course of that instructor is chosen randomly, and then, a lecture of the selected course is chosen randomly. The period p at which the selected lecture is scheduled in the timetable is designated as unavailable for instructor i. Period p becomes unavailable for all courses of instructor i. It is assumed that there can be at most one IP disruption for any instructor i.

For generating a CP disruption ($\langle c, \mathcal{P}^-, \mathcal{P}^+ \rangle$), again first an instructor is selected randomly, and then, a course c of that instructor is chosen randomly. Let p denote the period of the first lecture of course c. A random length of consecutive available periods for course c on the same day as p and including p is determined. These periods form the set $\mathcal{P}^-$. Then, the number of unavailable periods for course c at each day is calculated. If none of the days has at least $|\mathcal{P}^-|$ unavailable periods, the day with the maximum number of unavailable periods is selected. Otherwise, if there is at least one day with $|\mathcal{P}^-|$ or more unavailable periods, one such day is randomly selected. Given the chosen day, a set of consecutive unavailable periods starting with the first unavailable period of that day are assigned to $\mathcal{P}^+$, such that $|\mathcal{P}^+| \le |\mathcal{P}^-|$. It is assumed that there can be at most one CP disruption for any course c.

An RP disruption ($\langle r, p, d \rangle$) is generated by first selecting a room r randomly. Duration d is generated from $DU(1, 2)$,

Table 10 Example 3: Iterations

i	S	S'	Case	Archived	i	S	S'	Case	Archived
1	(1, 7)	(4, 3)	3	(4, 3)	6	(6, 5)	(7, 6)	1	(7, 6)
2	(4, 3)	(5, 9)	1	(5, 9)	7	(7, 6)	(9, 10)	2	–
3	(4, 3)	(8, 1)	3	(8, 1)	8	(9, 10)	(8, 8)	4	–
4	(8, 1)	(6, 5)	1	(6, 5)	9	(4, 3)	(3, 4)	3	(3, 4)
5	(6, 5)	(8, 8)	1	(8, 8)	10	(3, 4)	(5, 6)	1	(5, 6)

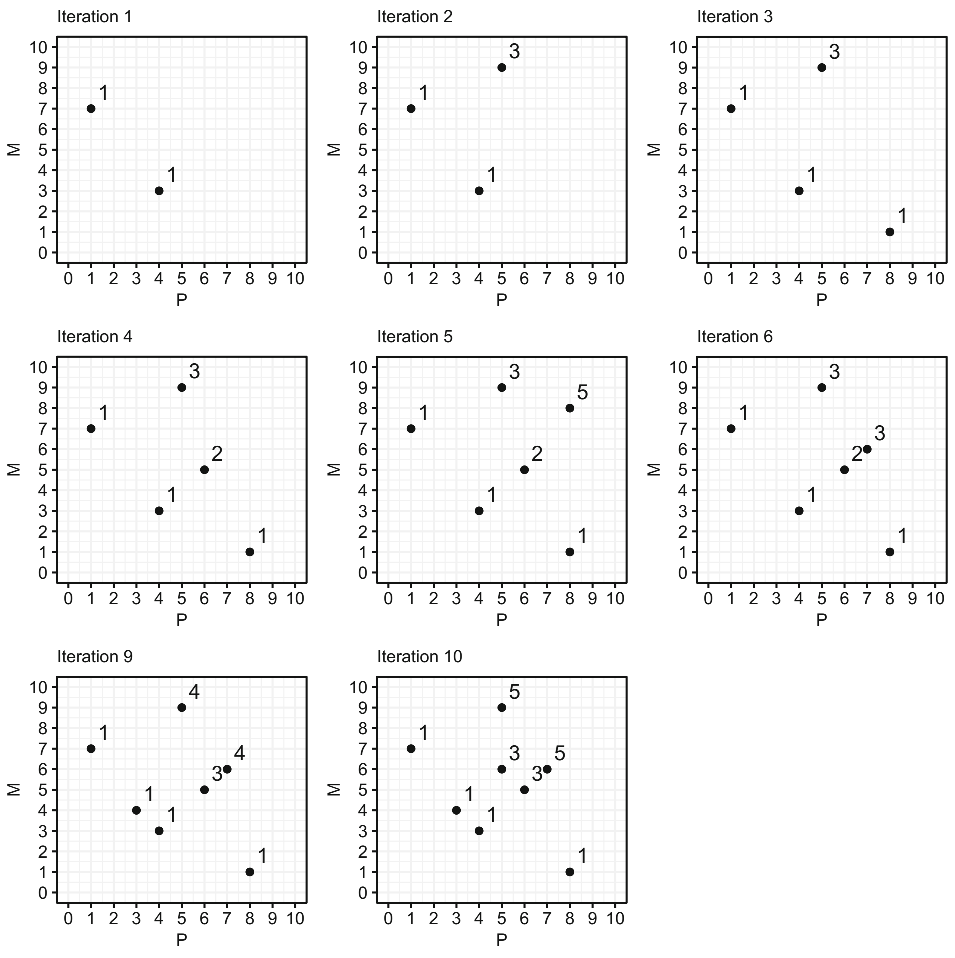

Fig. 2 Example 3: Maintaining the archive using $r^M(S)$ with $K = 5$ (numbers next to markers are $r^M(S)$)

where $DU(a, b)$ denotes the discrete uniform probability distribution with parameters a and b. Given d, period p is randomly chosen so that if $d = 2$, periods p and $p + 1$ are both on the same day. Since in the ITC-2007 instances all rooms are available for all periods, there are no prior room availability data to keep track of. It is assumed that there can be at most one RP disruption for any room r.

Finally, to generate a CS disruption ($\langle c, s\rangle$) first, a course c is randomly chosen, so that its first event, e_c, is not assigned to the room with the largest capacity, $K^{\max}$. (This could have been any event of the course, but for the sake of convenience we choose its first event.) Then, the new number of students, s, is randomly drawn from $DU(lowlim + 1, lowlim + gap)$, where $lowlim = \max(NS(c), K(R(e_c)))$ and $gap = \min(K^{\max} - lowlim, NS(c))$. It is assumed that there can be at most one CS disruption for any course c.

5.2 Test solutions for identifying a surrogate measure short list

For each instance ITCi, a set of 60 solutions, denoted by $\mathcal{S}_i$, were selected with the goal of having a diverse set of solutions. These solutions were selected from those that are accepted by a simulated annealing algorithm that minimizes the penalty associated with a solution. Selection was done from among nc collected solutions (nc = 50,000 or 100,000) that were accepted by the SA algorithm in its final iterations. Two criteria were used in making the selections: the penalty, $P(v)$, of the solution and the degree, $d(v)$, of the solution in a solution network created from the collected solutions. A solution network is designed so that an edge exists between two solutions i and j, if the Hamming distance between assigned period arrays for the events, A_v and A_w, is less than or equal to a cutoff value. The statistical analysis presented in Akkan et al. (2020) suggests that the degree of a node in these networks is weakly correlated with the robustness of that solution.

For each instance ITCi, four networks were generated (two with 50,000 and two with 100,000 collected solutions) and 15 solutions were selected from each network to be included in $\mathcal{S}_i$. Given a network, a joint frequency table was formed of all solutions of the network based on intervals of $d(v)$ and $P(v)$. Then, a solution was selected from those that fall into a selected joint interval. The selections were made from among the solutions with penalties that are close to the minimum penalty value, $P_{\min}$, found by the SA algorithm. For example, the $P_{\min}$ value for ITC7 was 39, so the $P(v)$ intervals used were [39, 40], [41, 42], [43, 44], [45, 46], [47, 48] and [49, 50]. Given a $P(v)$ interval, solutions were selected from the subsets that belonged to the smallest and largest $d(v)$ intervals, and in some cases from the intermediate ones. For example, given one of the networks for ITC7 with 50,000 solutions, among the solutions with $P(v) \in [39, 40]$ one solution was selected from each of the following intervals for $d(v)$: [4, 53], [54, 103], [104, 153] and [154, 203]. On the other hand, for those with $P(v) \in [43, 44]$ the degree intervals used were [4, 53] and [104, 153]. The details of how the networks were generated and how the solutions were selected from these networks are discussed in Akkan et al. (2020).

5.3 Correlation analysis

For each instance $ITCi$, Pearson correlation coefficients, ρ_i^m, were calculated between each surrogate measure, m, and the robustness measure $\overline{R}$, using the set of 60 solutions $\mathcal{S}_i$. Then, the absolute values of these correlations were ranked among those for each instance, in decreasing order so that the largest one is ranked first. Letting ϱ_i^m denote the rank of $|\rho_i^m|$, the average rank of m was calculated as, $\overline{\varrho}^m = 1/5 \sum_{i\in\{1,2,5,7,12\}} \varrho_i^m$. The nine best-ranking surrogate measures, their correlation coefficients and average ranks, $\overline{\varrho}^m$, are reported in Table 11. For each of these correlation coefficients, a test of hypothesis was done where the null hypothesis states the correlation is equal to zero.

The statistics presented in Table 11 reveal that three of the surrogate measures stand out, namely $\overline{UC}$, $\mathrm{CV_C}$ and $\mathrm{CV_{CL}}$. Correlation between $\overline{UC}$ and $\overline{R}$ is negative for all five instances, although for three of them we have sufficiently low p-values to reject the null hypothesis of 0 correlation. Both $\mathrm{CV_C}$ and $\mathrm{CV_{CL}}$, on the other hand, are positively correlated with $\overline{R}$. For two of the instances, the correlations associated with $\mathrm{CV_C}$ and, for three of the instances, the correlations associated with $\mathrm{CV_{CL}}$ are different from zero at a statistically significant level. Recall that $\mathbf{C}[c]$ array contains the number of conflict-free available periods for each course. Thus, the positive correlation associated with $\mathrm{CV_C}$ suggests the smaller $\mathrm{CV_C}$ is, the more robust the solution would be (with smaller $\overline{R}$). A similar interpretation applies to $\mathrm{CV_{CL}}$, as well. The scatter plots of these three surrogate measures with $\overline{R}$ are depicted in Figs. 3, 4, 5, 6 and 7.

5.4 Performance of the surrogate measures in identifying the Pareto frontier

The correlation analysis presented in the previous section has shown that $\overline{UC}$, $\mathrm{CV_C}$ and $\mathrm{CV_{CL}}$ are the three most promising surrogate measures based on their correlation with $\overline{R}$. However, considering the intended use of these measures within the MOSA algorithm, a second, complementary, analysis would shed more light into their potential performance.

Recall that the MOSA algorithm is designed to include solutions s with $r^M(s) \le K$ in the MOSA archive, where $K \in \mathbb{Z}^+$. With the right choice of M and K, one would like to obtain the final archive at the end of the MOSA algorithm's run, $\mathcal{A}^M(K)$, to be a short list of solutions that are highly likely to contain the solutions forming the Pareto frontier based on $(P, \overline{R})$, which we denote by $\mathcal{F}$. In other words, the task at hand is a binary classification of the solutions (on or not on the Pareto frontier). Evaluation of binary prediction performance is a very important issue in machine learning and data mining. Prati et al. (2011) provide a survey on graphical methods for predictive performance evaluation of classification rules/algorithms. They point out that when prior probabilities are very different, especially when negative cases heavily outnumber the positive ones and when positive class is of more interest than the negative one, it is better to use the so-called precision–recall curves. In the Pareto frontier classification problem addressed here, we have exactly the same situation: Only a couple or handful of solutions are on the Pareto frontier (i.e., positive cases) in a set of 60 solutions, and the solutions on the frontier are of more interest. Therefore, we have decided to follow their recommendation. Letting X be the true label of an instance (the "ground truth") and Y the prediction made by a predictive model, *precision* is defined as $P(X = True|Y = True)$ and *recall* is $P(Y = True|X = True)$ (which is also known as the true positive rate, tpr). Thus, for a surrogate measure M, and the associated archiving rule (M, K), *precision* is $P(s \in \mathcal{F}|s \in \mathcal{A}^M(K))$ and *recall* is $P(s \in \mathcal{A}^M(K)|s \in \mathcal{F})$.

To estimate these probabilities, we have done the following experimental analysis. Since we cannot determine the

Table 11 Pearson correlation coefficients with $\overline{R}$ and average ranks $\overline{\varrho}^m$

$\overline{\varrho}^m$	$\overline{UC}$ 8	CV_C 8.8	CV_{CL} 9.2	CV_{UC} 10	$\overline{RSU}$ 10.2	$\overline{RSA}$ 10.4	CV_{RSA} 11.8	SD_{UC} 12.4	SD_{DSU} 13.8
ITC1	−0.220*	0.136	−0.120	0.179	0.115	0.106	−0.123	−0.170	0.144
ITC2	−0.329♯	0.256♭	0.255♭	0.363♯	0.358♯	0.256♭	−0.363♯	0.369♯	0.047
ITC5	−0.141	0.222*	0.248*	0.114	−0.195	−0.169	0.139	0.092	−0.133
ITC7	−0.122	0.166	0.158	0.073	−0.169	−0.248♭	0.065	0.064	0.187
ITC12	−0.286♭	0.204	0.230*	0.236*	0.175	0.193	−0.224*	0.222*	−0.150

♯ $p < .01$, ♭ $p < .05$, * $p < .10$

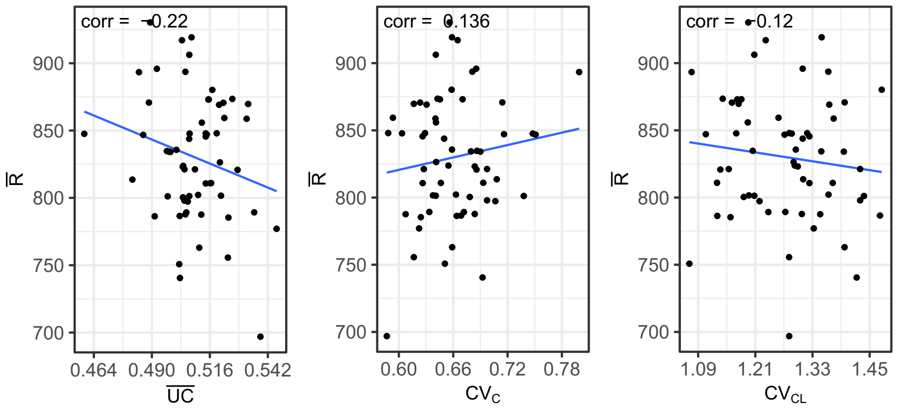
Fig. 3 ITC1: scatter plots of $\overline{R}$ versus selected surrogate measures

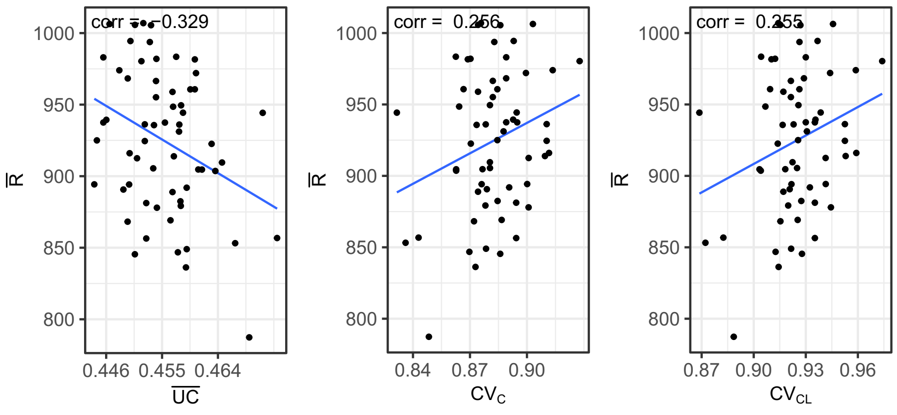
Fig. 4 ITC2: scatter plots of $\overline{R}$ versus selected surrogate measures

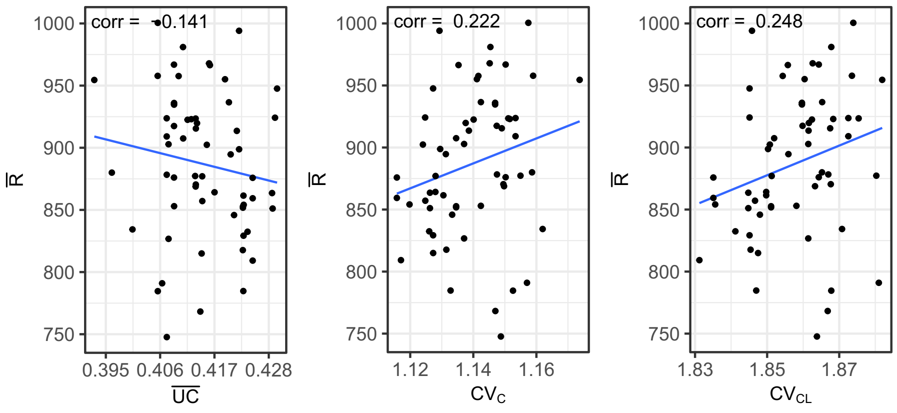
Fig. 5 ITC5: scatter plots of $\overline{R}$ versus selected surrogate measures

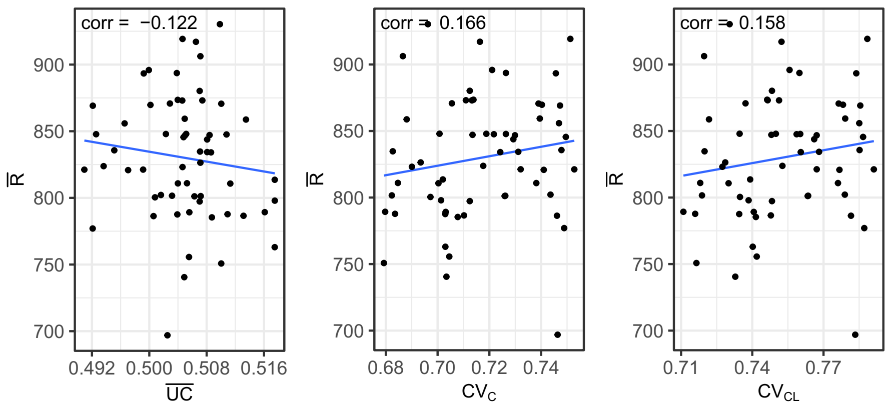

Fig. 6 ITC7: scatter plots of $\overline{R}$ versus selected surrogate measures

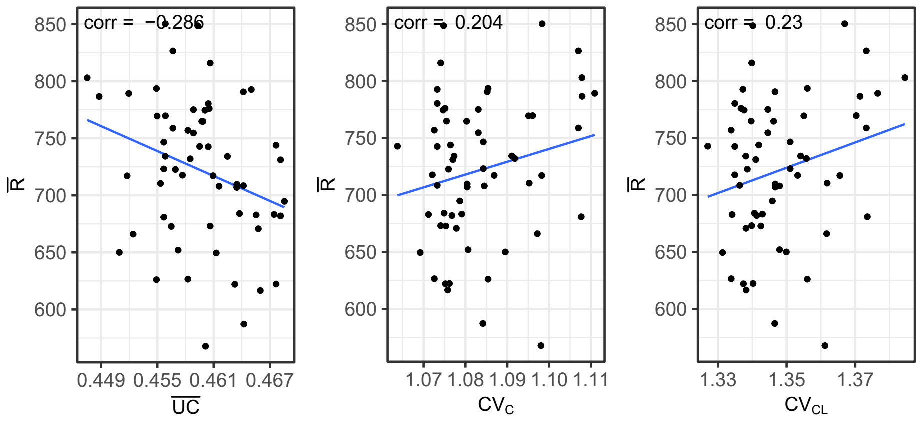

Fig. 7 ITC12: scatter plots of $\overline{R}$ versus selected surrogate measures

true (optimal) frontier, for instance, ITCi, $\mathcal{F}_i$, we obtained an approximation frontier, $\widetilde{\mathcal{F}}_i$. For each $s \in \mathcal{S}_i$, we calculated $\overline{R}(s, y)$ and determined the corresponding approximation frontier $\widetilde{\mathcal{F}}_i$. For each surrogate measure M, we calculated $r^M(s)$ of each solution based on objectives (P, M) and determined the solutions that fall into the archive $\mathcal{A}^M(K)_i$ for each (M, K). Then, the estimates of $P(s \in \mathcal{A}^M(K)|s \in \mathcal{F})$ and $P(s \in \mathcal{F}|s \in \mathcal{A}^M(K))$, denoted by $f^+(M, K)$ and $g^+(M, K)$, respectively, are calculated as:

$$f^+(M, K) = \frac{\sum_i |\{s : s \in \mathcal{A}^M(K)_i \ , s \in \widetilde{\mathcal{F}}_i\}|}{\sum_i |\widetilde{\mathcal{F}}_i|} \tag{5}$$

$$g^+(M, K) = \frac{\sum_i |\{s : s \in \mathcal{A}^M(K)_i \ , s \in \widetilde{\mathcal{F}}_i\}|}{\sum_i |\mathcal{A}^M(K)_i|} \tag{6}$$

Two additional surrogate measures were defined to serve as benchmarks. The first one, $MinR$, includes solution s in the archive $\mathcal{A}(K)$ if $r^B(s) \le K$, where $r^B(s) = \min_M r^M(s)$. The second one is $RAND$, which includes solution s in the archive if $u \le K$, where u is a random variate sampled from the $U(0, 1)$ probability distribution. Clearly, any reasonable surrogate measure should perform better than RAND.

Table 12 presents $f^+(M, K)$ for $\overline{UC}$, CV$_C$ and CV$_{CL}$, as well as $RAND$ and $MinR$, for $K = 1, \ldots, 12$. The results for $RAND$ are sample averages obtained with Monte Carlo simulation using a sample size of $n = 1000$. We can see that $f^+(\text{CV}_C, K)$ is greater than or equal to that of the other surrogate measures for all K. Table 13 presents $g^+(M, K)$ results for the same measures, and, again, CV$_C$ dominates other measures for all K. It is also worth noting that for $K > 5$ $MinR$ is worse than $RAND$, since it includes too many solutions in $\mathcal{A}(K)$ that are not in $\widetilde{\mathcal{F}}$.

Figure 8 depicts the precision–recall curve as a function of the K values, for comparing the best performing surrogate measure CV$_C$ with $RAND$. CV$_C$ is clearly better than $RAND$, especially for small K values. Precision values are relatively low due to the extremely small prior probability of a solution being on the Pareto frontier.

Table 14 presents some statistics on the sizes of the fronts for $K = 3$ and 5. One can see that the sizes of $\widetilde{\mathcal{F}}_i$ are quite small for all instances, so correctly identifying the solutions in $\widetilde{\mathcal{F}}_i$ is difficult. Comparing $|\widetilde{\mathcal{F}}_i|$ and $|\mathcal{A}^M(K)_i \cap \widetilde{\mathcal{F}}_i|$, we observe that $MinR$ is the only rule that managed to find at least one Pareto optimal solution for all ITC instances. On the other hand, this has led to roughly doubling the size of the solution archive, $|\mathcal{A}^{MinR}(K)_i|$, which is why we have seen a significant decline in precision as discussed above. Figure 9

Table 12 $f^{+}(K)$: *Recall* in identifying the solutions on the Pareto frontier

K	$\overline{UC}$	CV_C	CV_CL	$RAND$	$MinR$
1	0.182	0.273	0.182	0.073	0.273
2	0.273	0.364	0.273	0.135	0.364
3	0.364	0.364	0.364	0.209	0.545
4	0.455	0.455	0.455	0.275	0.545
5	0.455	0.545	0.455	0.348	0.545
6	0.545	0.545	0.455	0.418	0.545
7	0.545	0.545	0.455	0.466	0.545
8	0.545	0.636	0.455	0.515	0.636
9	0.636	0.636	0.455	0.542	0.727
10	0.636	0.636	0.545	0.581	0.727
11	0.727	0.727	0.636	0.628	0.909
12	0.727	0.818	0.636	0.668	0.909

Table 13 $g^{+}(K)$: *Precision* in identifying the solutions on the Pareto frontier

K	$\overline{UC}$	CV_C	CV_CL	$RAND$	$MinR$
1	0.125	0.25	0.154	0.068	0.143
2	0.1	0.148	0.125	0.065	0.087
3	0.083	0.108	0.108	0.065	0.086
4	0.083	0.102	0.1	0.065	0.07
5	0.07	0.109	0.089	0.066	0.063
6	0.076	0.091	0.074	0.066	0.054
7	0.064	0.081	0.064	0.064	0.048
8	0.058	0.081	0.056	0.062	0.051
9	0.064	0.071	0.048	0.059	0.054
10	0.059	0.065	0.054	0.058	0.051
11	0.061	0.066	0.058	0.058	0.058
12	0.057	0.066	0.052	0.058	0.053

depicts all solutions as well as those that are predicted to be on the frontier using $M = \text{CV}_\text{C}$ with $K = 5$, and those on the frontier (i.e., in $\widetilde{\mathcal{F}}_i$).

One might ask whether better performance can be achieved by a combination of the slack measures by the so-called meta-models or surrogate models. Jin (2005) gives a comprehensive review of fitness approximation methods in evolutionary computation. They list, among others, polynomial (response surface methodologies), kriging models, neural networks, support vector machines as some of the approaches used. They state that the literature provides no clear-cut conclusions on the advantages and disadvantages of different methods. Recently, Tong et al. (2021) provide a taxonomy of surrogate models with two major categories: absolute fitness models, which directly approximate the fitness function values of candidate solutions, and relative fitness models, which estimates the relative rank or preference of candidates

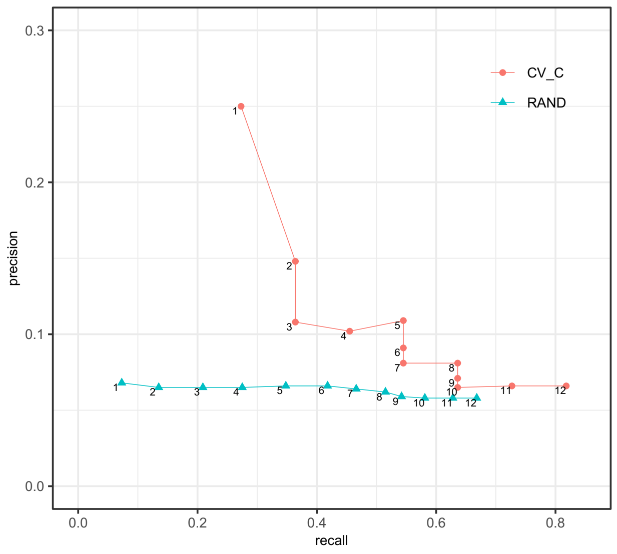

Fig. 8 Precision–recall curve for CV_C and RAND (with K next to each marker)

rather than their fitness values. The work in the latter category has developed after the review of Jin (2005), the first one being Runarsson (2006). To the best of our knowledge, most of this literature uses problem data and the values of the decision variables as input to the approximation models. Using slack measures as input would be a promising approach for the problem addressed here. Due to the large variety of approaches one should test, we believe investigating meta-models should be a follow-up study.

5.5 Performance of the surrogate measures within the MOSA algorithm

Given the results discussed in the previous section, the MOSA algorithm was run using surrogate measures CV_C and CV_CL, with $K = 5$. First, using CV_CL MOSA was run 20 times for each ITC instance. The final solution archives in these runs contained a total of 6083 solutions. Since for the calculation of $\overline{R}$ for a given solution we used a random sample of 100 disruption scenarios, calculating it for all solutions would be practically impossible. Hence, we decided to choose around 60 solutions for each ITC instance. With this computational budget for a given instance, it made sense to have these solutions with P values within an interval starting with $P_{\min}$, the best penalty obtained for that instance. It is sensible to focus on a segment of the Pareto frontier corresponding to a range of the lowest penalty values, where the performance of the algorithm matters the most. Furthermore, by making this a sufficiently narrow interval one can hope to obtain a good approximation of the Pareto frontier, $\widetilde{\mathcal{F}}$.

Say the chosen penalty interval is $[P_{\min}, P_{ul}]$. Using the same P_{ul} for runs using both CV_C and CV_CL would give

Table 14 Frontier size statistics (sample averages reported for $RAND, n = 1000$)

i	$\lvert\widetilde{\mathcal{F}}_i\rvert$	$\lvert\mathcal{A}^M(K)_i \cap \widetilde{\mathcal{F}}_i\rvert / \lvert\mathcal{A}^M(K)_i\rvert$									
		$K = 3$					$K = 5$				
		UC	$\mathrm{CV_C}$	$\mathrm{CV_{CL}}$	$RAND$	$MinR$	UC	$\mathrm{CV_C}$	$\mathrm{CV_{CL}}$	$RAND$	$MinR$
1	1	1/3	1/3	0/3	0.06/3.0	1/8	1/5	1/5	0/5	0.07/5.0	1/14
2	4	1/11	1/9	1/9	0.50/7.2	2/15	2/16	2/15	2/15	0.78/11.7	2/19
5	2	1/8	1/14	1/12	0.61/8.7	1/15	1/13	1/17	1/15	0.98/14.5	1/18
7	2	1/12	1/4	1/5	0.31/7.1	1/15	1/18	1/7	1/8	0.57/11.9	1/22
12	3	0/14	0/7	1/8	0.81/9.1	1/17	0/19	1/11	1/13	1.39/15.2	1/23
Ave. $\lvert\mathcal{A}^M(K)_i\rvert$		9.6	7.4	7.4	7.0	14	14.2	11	11.2	11.7	19.2

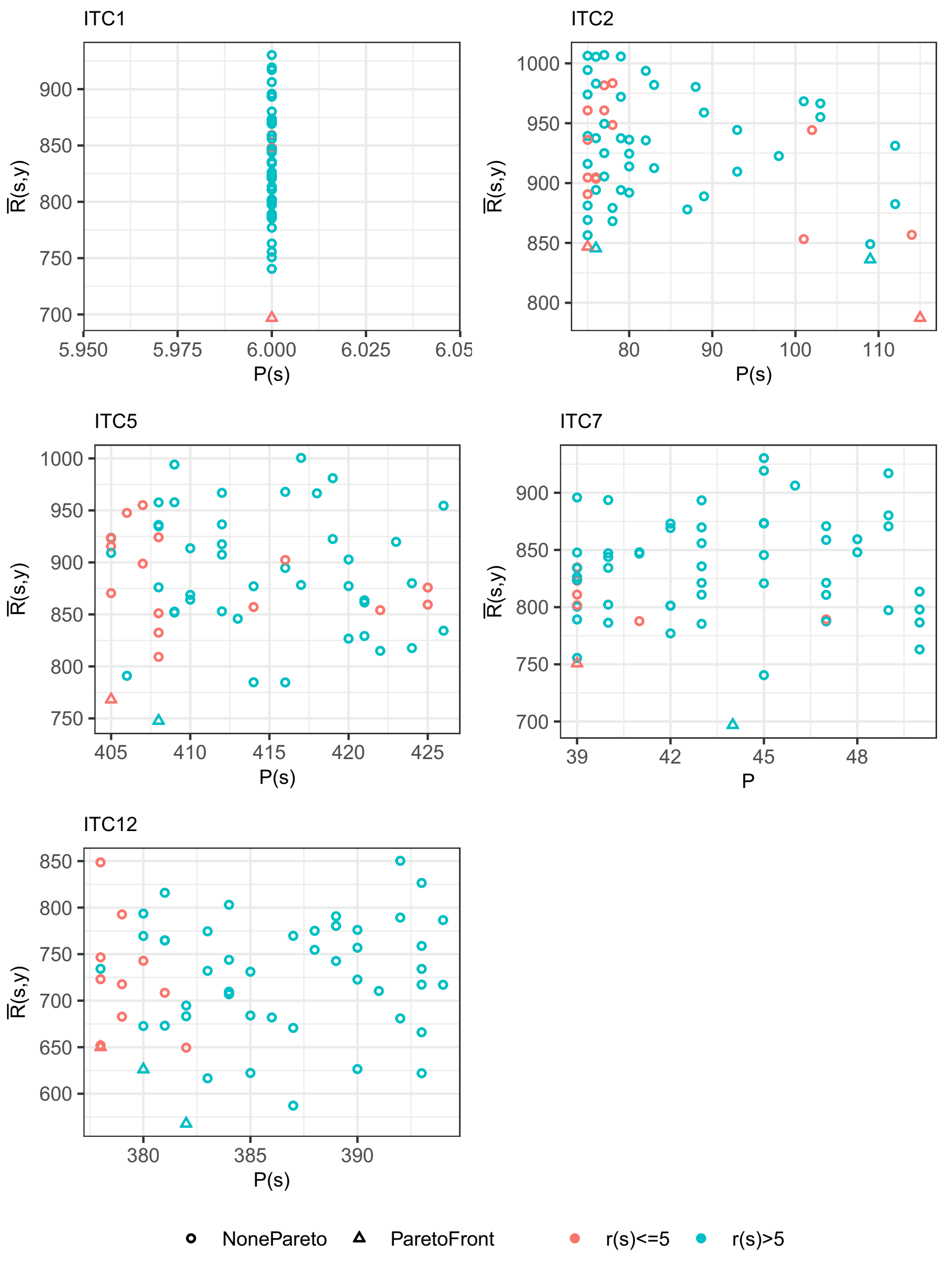

Fig. 9 Solutions with $M = \mathrm{CV_C}$, $K = 5$ and the $(P, \overline{R})$ Pareto frontier

Table 15 Statistics on solution archives and selected solutions—MOSA using CV_{CL}

		ITC1	ITC2	ITC5	ITC7	ITC12
No. runs used		20	20	20	20	18
No. runs with a selected soln.		20	10	15	16	18
Archive size per run	Min.–Max.	18–47	28–77	43–364	15–50	51–135
	Ave.	30.2	52.95	104.5	33.25	83.39
Selected solutions:						
No.		58	60	63	60	62
$P_{\min} - P_{ul}$		5–7	65–89	318–394	37–58	347–392
No. per run (excl. runs	Min.–Max.	1–6	3–11	2–8	2–8	2–9
with 0 selected soln.)	Ave.	2.9	6	4.2	3.75	4.13
With same P	Min.–Max.	8–36	1–6	1–6	1–6	2–5
	Ave.	19.33	2.73	2.33	3.16	2.19

Table 16 Statistics on solution archives and selected solutions—MOSA using CV_C

		ITC1	ITC2	ITC5	ITC7	ITC12
No. runs used		20	30	20	17	30
No. runs with a selected soln.		20	13	12	12	18
Archive size per run	Min.–Max.	22–48	19–76	56–350	18–78	35–251
	Ave.	30.45	44.2	95.35	34.06	83.9
Selected solutions:						
No.		64	59	59	64	60
$P_{\min} - P_{ul}$	Min.–Max.	5–7	75–89	345–394	40–58	357–392
No. per run (excl. run	Min.–Max.	1–6	2–9	2–8	2–10	2–6
with 0 selected soln.)	Ave.	3.2	4.54	4.92	5.33	3.53
With same P	Min.–Max.	15–28	1–8	1–5	1–9	1–6
	Ave.	21.33	3.93	2.15	3.76	2.61

Table 17 Summary statistics on the generated frontiers

	ITC1	ITC2	ITC5	ITC7	ITC12
Number of frontiers					
CV_C	20	13	12	12	18
CV_{CL}	20	10	15	16	17
Frontier size (ave., median)					
CV_C	3.20; 3	4.54; 5	4.69; 5	5.33; 5	3.33; 3
CV_{CL}	1.15; 1	2.7; 2.5	1.33; 1	1.25; 1	1.41; 1
Hypervolume (ave., median)—larger the better					
CV_C	0.081; 0.075	0.044; 0.04	0.029; 0.02	0.053; 0.060	0.023; 0.015
CV_{CL}	0.098; 0.080	0.096; 0.08	0.031; 0.02	0.069; 0.075	0.047; 0.050
Generational Distance (ave., median)—smaller the better					
CV_C	0.226; 0.216	0.150; 0.103	0.155; 0.144	0.149; 0.155	0.165; 0.144
CV_{CL}	0.473; 0.518	0.146; 0.134	0.300; 0.310	0.254; 0.203	0.125; 0.099

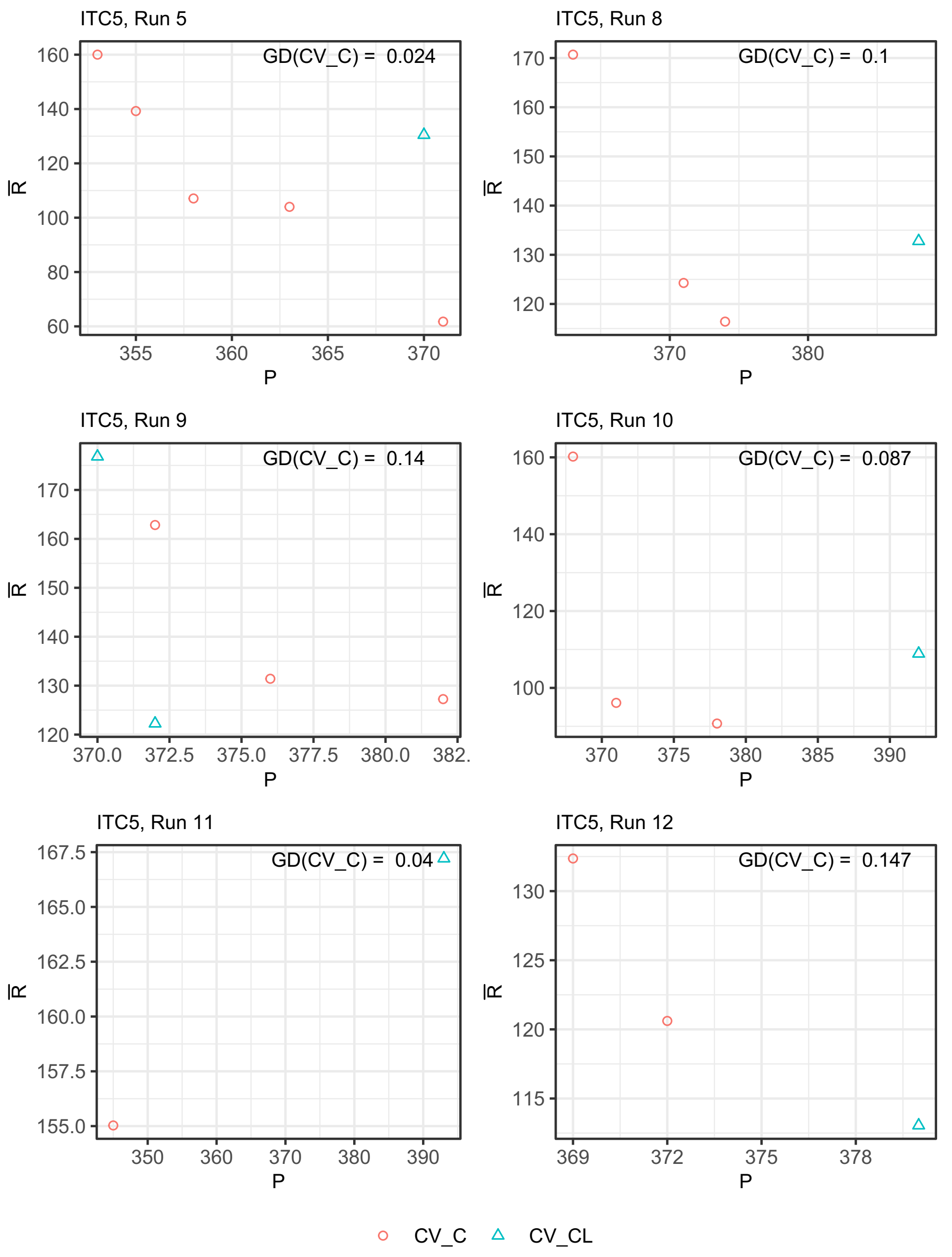

Fig. 10 $\mathcal{F}_r^{\mathrm{CV_{CL}}}(5)$ and $\mathcal{F}_r^{\mathrm{CV_C}}(5)$ for runs that gave the lowest GD for $\mathrm{CD_C}$

us around 120 solutions to obtain $\widetilde{\mathcal{F}}$. Since P values are by definition integer, we tried to ensure the average number of solutions for each P in $[P_{\min}, P_{ul}]$ was 2 or more, for each surrogate measure. In other words, by ensuring $[P_{\min}, P_{ul}]$ to be narrow enough, we can calculate $\overline{R}$ for multiple solutions with the same P and obtain a better estimate of Pareto frontier. Table 15 presents some summary statistics for the solutions selected from the runs that used $\mathrm{CV_{CL}}$. The number of solutions selected for each ITC instance ranges between 58 and 63. The average number of solutions with the same P was at least 2.19.

Then, using the same P_{ul} values for each instance, solutions were selected from the MOSA archives that were obtained when the surrogate measure $\mathrm{CV_C}$ was used. In this case, obtaining around 60 solutions with $P \leq P_{ul}$ required up to 30 MOSA runs. Table 16 summarizes the statistics for these solutions. The data files of these solutions and the corresponding disruption scenarios for each solution are available for download at figshare data repository (Akkan and Gülcü 2020).

To evaluate how good the approximation frontiers are, two widely used metrics, hypervolume (HV) and generational distance (GD), were used (Zitzler and Thiele 1999). GD was

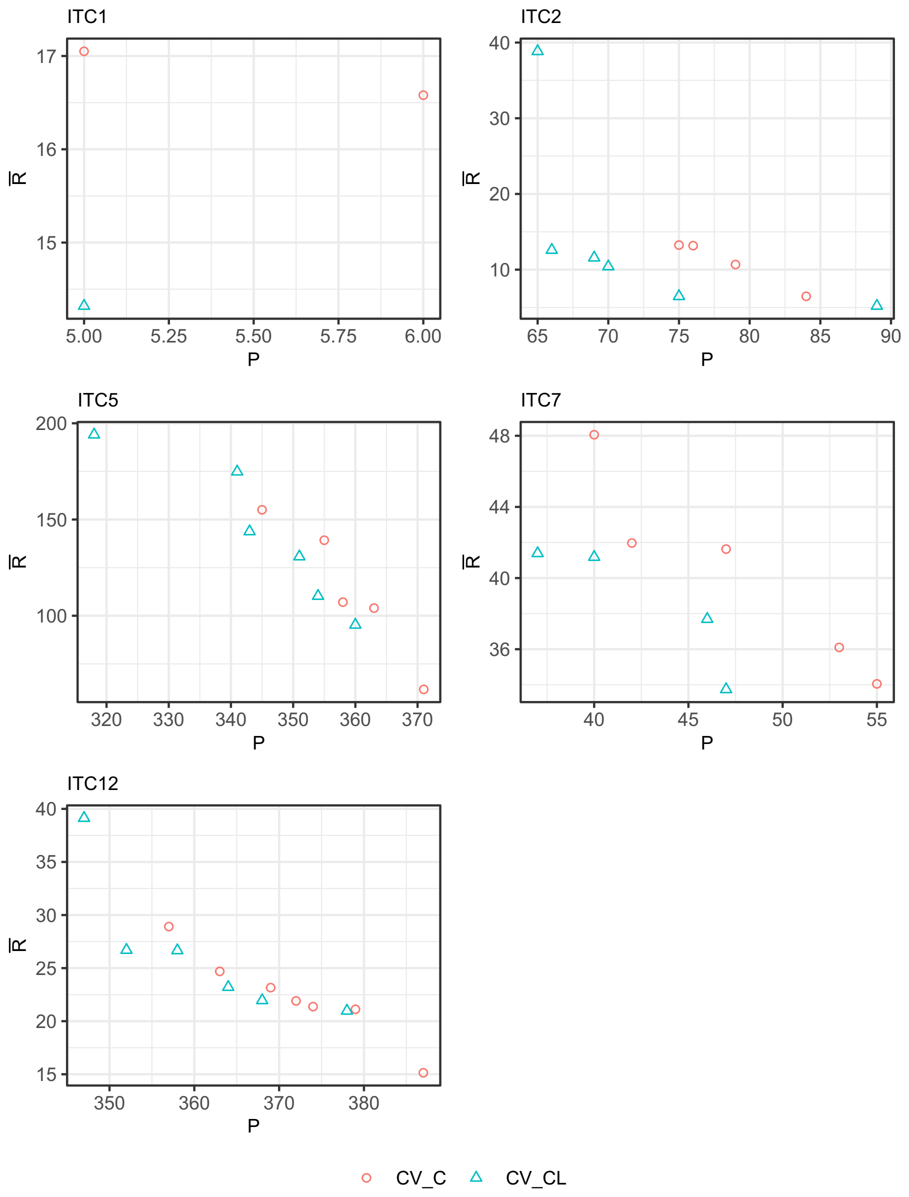

Fig. 11 Comparison of the best aggregate frontiers, $\mathcal{F}^{CV^*_{CL}}(i)$ and $\mathcal{F}^{CV^*_{C}}(i)$ for each ITCi

developed to measure the distance between two frontiers. Ideally, one of these would be the optimal Pareto frontier, however, when, as often the case, one does not know the optimal Pareto frontier, one would need to develop a benchmark or best frontier that can be constructed. We used the following approach to construct the best frontier: For a given instance ITCi, let $\mathcal{F}_r^M(i)$ denote frontier obtained in the r^{th} run of the MOSA algorithm using surrogate measure M. Combining the frontiers from all runs of CV$_{CL}$ and CV$_C$, an aggregate frontier is obtained, which is denoted by $\mathcal{F}^*(i)$. For each $\mathcal{F}_r^M(i)$, GD calculation is done with respect to $\mathcal{F}^*(i)$.

Table 17 provides some summary statistics on the frontiers and their HV and GD values. The number of frontiers reveal that the aggregate fronts are constructed by making use of between 23 frontiers (for ITC2) and 40 frontiers (for ITC1). Furthermore, we see that $\mathcal{F}_r^{CV_C}(i)$ have significantly more solutions than $\mathcal{F}_r^{CV_{CL}}(i)$ for all ITCi. Summary statistics on HV reveal that for all ITCi, on the average, CV$_{CL}$ yields better frontiers than CV$_C$. On the other hand, GD statistics provide mixed results that favor CV$_C$. For ITC1, ITC5, and ITC3, average and median GD for CV$_C$ is clearly better than that of CV$_{CL}$, whereas for ITC12 CV$_{CL}$ is better and for ITC2 the two surrogate measures perform equally well. As an example of how the individual frontiers could differ, Fig. 10 depicts the frontiers obtained in six runs where CV$_C$ gave the smallest GD for ITC5. Different Pareto frontier quality metrics yielding different results are unfortunately common in multi-criteria optimization (see Zitzler and Thiele 1999), and it is widely acknowledged that there is no single metric that can be used to evaluate the quality of an approximation frontier.

In order to gain better insight into the performance difference between CV_{CL} and CV_{C}, the aggregate fronts $\mathcal{F}^{M^*}(i)$ were constructed separately for each surrogate measure M. In other words, given M, the solutions in $\mathcal{F}_r^M(i)$ were combined for all r in order to obtain a good approximation front. This is how one would obtain the approximation frontier when one uses a multi-start MOSA algorithm, where MOSA would be run with a new random initial solution in each run. Figure 11 depicts, for each ITCi, these aggregate frontiers, $\mathcal{F}^{M^*}(i)$. For all ITCi, $\mathcal{F}^{\text{CV}^*_{\text{CL}}}(i)$ is clearly the better approximation frontier. Considering this finding along with results provided in Table 17 suggests that using CV_{CL} yields small frontiers (overall average frontier size was 1.63, compared to 4.3 for CV_{C}), but these are good solutions, and when the solutions from multiple runs are combined, one obtains quite good approximation frontiers. Relatively poor performance of CV_{CL} in terms of GD reported in Table 17 can partly be explained by having 1 or 2 solutions on $\mathcal{F}_r^{\text{CV}_{\text{CL}}}(i)$, which results in large average distance to the larger and more widely spread $\mathcal{F}^*(i)$. Taking into account these results, one can conclude that a multi-start MOSA using CV_{CL} would be an effective approach at constructing a good Pareto frontier for this timetabling problem.

6 Concluding remarks

In this article, we have addressed the problem of constructing an approximation to the Pareto frontier defined by two criteria, solution quality and solution robustness, for the curriculum-based university course timetabling problem of the International Timetabling Competition 2007 (ITC-2007). The need for the robustness objective comes from four types of input data changes/corrections, so-called disruptions, that can render a given solution infeasible and deteriorate its solution quality. Since disruptions are assumed to be random, the robustness objective is a stochastic one, but one without a closed-form formula.

To solve this problem, a multi-objective simulated annealing (MOSA) algorithm was developed. To find a good surrogate measure for robustness that can be used within MOSA, a large set of measures that attempted to quantify the availability of slack in the timetable (in rooms/periods) were developed. Analysis of the results of computational experiments helped reduce these to a short list of eight, and then to two. These two surrogate measures were then implemented in a MOSA algorithm. Computational experiments of the two versions of the MOSA algorithm revealed that a single run of CV_{C} is likelier to provide a better frontier with an average of around 4 solutions; however, if the MOSA is run multiple times with a different initial random solution using CV_{CL}, and the resultant frontiers are combined into one aggregate frontier, this should yield a significantly better frontier than one would get using the same multi-start approach with CV_{C}.

An interested avenue for future research could be developing a larger set of domain-specific features and testing the use of these with surrogate models (such as random forests), which, to the best of our knowledge, have not been tested on large-scale timetabling problems.

Appendix: The IP model for robustness calculation

As discussed in Sect. 2.1, the objective function of the CB-CTP of ITC-2007 is a weighted sum of soft constraint violations, which is referred to as the penalty. The robustness measure for a given initial solution and a disruption scenario was given in Equation 3 as

$$R(S_0, y_n) = \min_{S \in \mathcal{N}(S_0, y_n)} \Phi_n(S, S_0), \text{ where}$$
$$\Phi_n(S, S_0) = P_{\text{ave}} \cdot 1_{D(S,S_0) > \delta_n^p} + (P(S) - P(S_0))^+$$

Given the set of all events, $\mathcal{E}$, the feasible solution space $\mathcal{N}(S_0, y_n)$ is defined by the updated set of available periods $\mathcal{T}_e(y_n)$, for each event $e \in \mathcal{E}$ and the set of available rooms, $\mathcal{R}_p(y_n)$, available for each period $p \in \mathcal{T}_e(y_n)$ for all $e \in \mathcal{E}$. (Thus, rooms in $\mathcal{R}_p$ can be used by any event.) To simplify the notation, in what follows, we drop the notation for the disruption scenario, y_n, and simply denote the set of available periods and rooms as $\mathcal{T}_e$ and $\mathcal{R}_p$, respectively. The notation used for the input data of the IP model is listed in Table 18.

The formulation of the IP model is an event-based one using the following decision variables:

$$\begin{aligned}
X_{\text{epr}} &= 1 \text{ if event } e \text{ is scheduled at period} \\
&\quad\ p \text{ in room } r, 0 \text{ otherwise;} \\
&\quad \forall e \in E, p \in T_e, r \in R_p \\
Y_{cr} &= 1 \text{ if course } c \text{ is scheduled in room } r, 0 \text{ otherwise;} \\
&\quad\ \forall c \in C, r \in \mathcal{R} \\
Z_{pm} &= 1 \text{ if curriculum } m \text{ has an isolated course in period} \\
&\quad\ p, 0 \text{ otherwise;} \\
&\quad \forall p \in \mathcal{P}, m \in \mathcal{M}. \\
W_{cd} &= 1 \text{ if at least one event of course} \\
&\quad\ c \text{ is scheduled on day } d, \\
&\quad 0 \text{ otherwise; } \forall c \in \mathcal{C}, \forall d \in \Delta \\
V_{\text{C}} &= 1 \text{ if minimum work day constraint is violated} \\
&\quad \text{for course } c, \\
&\quad 0 \text{ otherwise; } \forall c \in \mathcal{C}
\end{aligned}$$

Table 18 Notation for the data

$K(r)$	: Capacity of room r
$\mathrm{MW}(c)$	: Minimum working days required for course c
$\mathrm{NS}(c)$	: The number of students for course c
$C(e)$	: The course to which event e belongs to
$M(e)$	: The curriculum of event e
$F^D(p)$	$= \begin{cases} -1, & \text{if period } p \text{ is the first period of a day} \\ 0, & \text{if period } p \text{ is neither last nor first period of a day} \\ 1, & \text{if period } p \text{ is the last period of a day.} \end{cases}$
$\pi(e)$	: the period to which event is assigned in the initial schedule S_0
$\mathcal{E}$	: Set of events
$\mathcal{C}$	: Set of courses
$\mathcal{I}$	: Set of instructors
$\mathcal{R}$	: Set of rooms
$\mathcal{T}$	: Set of periods
$\mathcal{T}_e$	: Available periods for event e
$T^N_{e,d}$	: Available periods on day d for event e
$\mathcal{R}_p$	: Available rooms for period p
$\mathcal{E}_c$	: Events of course c
$\mathcal{E}_{p,r,m,i}$	: Events taught by teacher i, of curriculum m, and can be assigned to period p, room r, $\{e : e \in \mathcal{E},\ p \in \mathcal{T}_e,\ r \in \mathcal{R}_p,\ M(e) = m,\ I(e) = i\}$

S_{pr} = number of students assigned to room r in period p, in excess of the room capacity; $\forall p \in \mathcal{P}, r \in \mathcal{R}$.

M_{pm} = 1 if there exists an event belonging to curriculum m scheduled in period p; $\forall p \in \mathcal{P}, m \in \mathcal{M}$.

Q = 1 if distance to the initial solution, S_0 exceeds δ_i^p, 0 otherwise;

The objective function to be minimized is then $R = P_{\text{ave}}Q + U$, where $U = \max\{P - P_0, 0\}$, P is the penalty function value of the solution that minimizes R, and P_0 is the penalty of the initial (disrupted) solution. The constraints of the model are as follows:

$$\sum_{e\in\mathcal{E}_{p,\cdot,\cdot,\cdot}} X_{\text{epr}} \le 1, \quad p \in \mathcal{T}, r \in \mathcal{R}_p \tag{7}$$

$$\sum_{e\in\mathcal{E}_{p,\cdot,m,\cdot}} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} \le 1, \quad p \in \mathcal{T}, m \in \mathcal{M} \tag{8}$$

$$\sum_{e\in\mathcal{E}_{p,\cdot,\cdot,i}} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} \le 1, \quad p \in \mathcal{T}, i \in \mathcal{I} \tag{9}$$

$$\sum_{p\in\mathcal{T}_e} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} = 1, \quad e \in \mathcal{E} \tag{10}$$

$$\sum_{e\in\mathcal{E}} \sum_{p\in\mathcal{T}_e\setminus\pi(e)} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} \le \delta_i^p + (f(\delta_i^p, \delta_i^r) - \delta_i^p)Q \tag{11}$$

$$\sum_{e\in\mathcal{E}_c} \sum_{p\in P_e} X_{\text{epr}} \le |E_c^C| Y_{cr}, \quad c \in \mathcal{C}, r \in \mathcal{R} \tag{12}$$

$$\sum_{e\in E_{p,\cdot,m,\cdot}} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} = M_{pm}, \quad p \in \mathcal{P}, m \in \mathcal{M} \tag{13}$$

$$M_{pm} - M_{(p+1)m} \le Z_{pm}, \quad p \in \mathcal{P}, F^D(p) = -1, m \in \mathcal{M} \tag{14}$$

$$-M_{(p-1)m} + M_{pm} - M_{(p+1)m} \le Z_{pm}, \quad p \in \mathcal{P}, F^D(p) = 0, m \in \mathcal{M} \tag{15}$$

$$-M_{(p-1)m} + M_{pm} \le Z_{pm}, \quad p \in \mathcal{P}, F^D(p) = 1, m \in \mathcal{M} \tag{16}$$

$$\sum_{e\in\mathcal{E}_c} \sum_{p\in\mathcal{T}_{e,d}} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} \le E^{\max} W_{cd}, \quad c \in \mathcal{C}, d \in \mathcal{D} \tag{17}$$

$$\sum_{e\in\mathcal{E}_c} \sum_{p\in\mathcal{T}_{e,d}} \sum_{r\in\mathcal{R}_p} X_{\text{epr}} \ge W_{cd}, \quad c \in \mathcal{C}, d \in \mathcal{D} \tag{18}$$

$$\sum_{d\in\mathcal{D}} W_{cd} + V_c \ge MW(c), \quad c \in \mathcal{C} \tag{19}$$

$$\sum_{d\in\mathcal{D}} W_{cd} \le |\mathcal{E}_c|, \quad c \in \mathcal{C} \tag{20}$$

$$\sum_{e\in\mathcal{E}} NS(C(e)) X_{\text{epr}} \le K(r) + S_{pr}, \quad p \in \mathcal{T}, r \in \mathcal{R} \tag{21}$$

$$\sum_{p\in\mathcal{T}} \sum_{r\in\mathcal{R}} S_{pr} = RC \tag{22}$$

$$\sum_{c\in\mathcal{C}} V_c \quad = MW \tag{23}$$

$$\sum_{m\in\mathcal{M}}\sum_{p\in\mathcal{P}} Z_{pm} \quad = CC \tag{24}$$

$$\sum_{c\in\mathcal{C}}\left(\sum_{r\in\mathcal{R}} Y_{cr} - 1\right) \quad = RS \tag{25}$$

$$(RC + 5MW + 2CC + RS) - P_0 \le U \tag{26}$$

$$X_{\text{epr}} \in \{0,1\} \quad e \in \mathcal{E}, p \in \mathcal{T}_e, r \in \mathcal{R}_{et} \tag{27}$$

$$W_{cd} \in \{0,1\} \quad c \in \mathcal{C}, d \in \mathcal{D} \tag{28}$$

$$M_{pm} \in \{0,1\} \quad p \in \mathcal{P}, m \in \mathcal{M} \tag{29}$$

$$V_c \in \{0,1\} \quad \forall c \in \mathcal{C} \tag{30}$$

$$Y_{cr} \in \{0,1\} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{31}$$

$$Z_{pm} \in \{0,1\} \quad \forall p \in \mathcal{P}, m \in \mathcal{M} \tag{32}$$

$$S_{pr} \ge 0 \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \tag{33}$$

$$U \ge 0 \tag{34}$$

$$Q \in \{0,1\} \tag{35}$$

Constraint (7) ensures at most one event can be assigned to each available room at each period. Constraint (8) ensures a period can have at most one event of a curriculum. Constraint (9) ensures a period can have at most one event of a teacher. Constraint (10) ensures every event is assigned to exactly one room in one period. Constraint (11) ensures the number of events that are not assigned to their pre-disruption period does not exceed $f(\delta_i^p, \delta_i^r)$ and sets the indicator decision variable Q to 1 if it exceeds δ_i^p. Constraint (12) is needed for the room stability soft constraints. They ensure that if at least one event of a given course $c \in C$ uses a room r, the corresponding decision variable Y_{cr} takes the value of 1. Constraint (13) ensures that if curriculum m has an event at period p, the decision variable M_{mp} takes a value of 1. Constraints (14), (15) and (16) ensure that if curriculum m has an *isolated* event at period p, the decision variable Z_{pm} takes a value of 1. Constraint (17) ensures that W_{cd} takes the value 1 if a rescheduled event of course c uses day d, where $E^{\max}$ is the maximum number of events per day per course. Constraint (18) forces the decision variable W_{cd} to 0 if no event of course c is scheduled on day d. Constraint (19) ensures that V_c gives the violation of the minimum working day requirement for each course. Constraint (20) ensures that for each course c the number of W_{cd} variables that are equal to 1 does not exceed the number of events for that course, $|\mathcal{E}_c^C|$. Constraint (21) ensures that S_{pr} provides the room capacity violation at each period and in each room. Finally, Constraint (26) calculates the penalty of the solution, where the calculations of individual soft constraint violations are depicted separately in Constraints (22), (23), (24) and (25).

References

Akkan, C., & Gülcü, A. (2018). A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. *Computers & Operations Research, 90*, 22–32. https://doi.org/10.1016/j.cor.2017.09.007

Akkan, C., & Gülcü, A. (2020). Solution and disruption scenario files for the robustness of curriculum-based course timetabling problem of ITC-2007. https://doi.org/10.6084/m9.figshare.13359500.

Akkan, C., Gülcü, A., & Kuş, Z. (2020). Search space sampling by simulated annealing for identifying robust solutions in course timetabling. In: *Proceedings of the 2020 IEEE congress on evolutionary computation*, pp 1–10, https://doi.org/10.1109/CEC48606.2020.9185823.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation, 12*(3), 269–283. https://doi.org/10.1109/TEVC.2007.900837

Bartz-Beielstein, T., & Zaefferer, M. (2017). Model-based methods for continuous and discrete global optimization. *Applied Soft Computing, 55*, 154–167. https://doi.org/10.1016/j.asoc.2017.01.039

Bonutti, A., de Cesco, F., Gaspero, L. D., & Schaerf, A. (2012). Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research, 194*(1), 59–70.

Brownlee, A. E., & Wright, J. A. (2015). Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation. *Applied Soft Computing Journal, 33*, 114–126. https://doi.org/10.1016/j.asoc.2015.04.010

Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R., & Kendall, G. (2021). A survey of university course timetabling problem: perspectives, trends and opportunities. *IEEE Access, 9*, 106515–106529. https://doi.org/10.1109/ACCESS.2021.3100613

Czyzk, P., & Jaszkiewicz, A. (1997). *Pareto Simulated Annealing* (pp. 297–307). Springer. https://doi.org/10.1007/978-3-642-59132-7_33.

Diaz, J. E., Handl, J., & Xu, D. L. (2017). Evolutionary robust optimization in production planning - interactions between number of objectives, sample size and choice of robustness measure. *Computers and Operations Research, 79*, 266–278. https://doi.org/10.1016/j.cor.2016.06.020

Fei, X., Branke, J., & Gulpinar, N. (2019). New sampling strategies when searching for robust solutions. *IEEE Transactions on Evolutionary Computation, 23*(2), 273–287. https://doi.org/10.1109/TEVC.2018.2849331

Gülcü, A., & Akkan, C. (2020). Robust university course timetabling problem subject to single and multiple disruptions. *European Journal of Operational Research, 283*(2), 630–646. https://doi.org/10.1016/j.ejor.2019.11.024

Hardy, B., Lewis, R., & Thompson, J. (2018). Tackling the edge dynamic graph colouring problem with and without future adjacency information. *Journal of Heuristics, 24*(3), 321–343. https://doi.org/10.1007/s10732-017-9327-z

Hazır, Ö., Haouari, M., & Erel, E. (2010). Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research, 207*(2), 633–643. https://doi.org/10.1016/j.ejor.2010.05.046

Hughes, M., Goerigk, M., & Dokka, T. (2021). Automatic generation of algorithms for robust optimisation problems using Grammar-Guided Genetic Programming. *Computers and Operations Research, 133*(March), 105364. https://doi.org/10.1016/j.cor.2021.105364

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing, 9*(1), 3–12. https://doi.org/10.1007/s00500-003-0328-5

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation, 1*(2), 61–70. https://doi.org/10.1016/j.swevo.2011.05.001

Jin, Y., & Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation, 9*(3), 303–317. https://doi.org/10.1109/TEVC.2005.846356

Kingston, J. H. (2013). Educational timetabling. In: Uyar, A. S., Ozcan, E., & Urquhart, N. (eds) *Automated scheduling and planning: from theory to practice*. Springer. https://doi.org/10.1007/978-3-642-39304-4

Kleywegt, A. J., Shapiro, A., & Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization, 12*(2), 479–502. https://doi.org/10.1137/S1052623499363220

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2008). Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of Scheduling, 11*(2), 121–136. https://doi.org/10.1007/s10951-007-0021-0

Lemos, A., Monteiro, P. T., & Lynce, I. (2020). Minimal perturbation in university timetabling with maximum satisfiability. In: *17th International conference on the integration of constraint programming, artificial intelligence, and operations research (CPAIOR20)*.

Leon, V. J., Wu, S. D., & Storer, R. H. (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions, 26*(5), 32–43.

Lindahl, M., Stidsen, T., & Sørensen, M. (2019). Quality recovering of university timetables. *European Journal of Operational Research, 276*(2), 422–435. https://doi.org/10.1016/j.ejor.2019.01.026

McCollum, B. (2007). A perspective on bridging the gap between theory and practice in university timetabling. *Practice and Theory of Automated Timetabling, LNCS, 3867*, 3–23.

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. https://doi.org/10.1287/ijoc.1090.0320

Müller, T., Rudová, H.,& Barták, R. (2005). Minimal perturbation problem in course timetabling. In: E. Burke & M. Trick (Eds.), *Practice and theory of automated timetabling V. Lecture notes in computer science* (Vol. 3616, pp. 126–146). https://doi.org/10.1007/11593577_8

Phillips, A. E., Walker, C. G., Ehrgott, M., & Ryan, D. M. (2017). Integer programming for minimal perturbation problems in university course timetabling. *Annals of Operations Research, 252*(2), 283–304. https://doi.org/10.1007/s10479-015-2094-z

Prati, R. C., Batista, G. E., & Monard, M. C. (2011). A survey on graphical methods for classification predictive performance evaluation. *IEEE Transactions on Knowledge and Data Engineering, 23*(11), 1601–1618. https://doi.org/10.1109/TKDE.2011.59

Runarsson, T. P. (2006). Ordinal regression in evolutionary computation. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) 4193 LNCS:1048–1057*. https://doi.org/10.1007/11844297_106.

Serafini, P. (1994). *Simulated annealing for multi objective optimization problems* (pp. 283–292). Springer. https://doi.org/10.1007/978-1-4612-2666-6_29.

Smith, K. I., Everson, R. M., Fieldsend, J. E., Murphy, C., & Misra, R. (2008). Dominance-based multiobjective simulated annealing. *IEEE Transactions on Evolutionary Computation, 12*(3), 323–342. https://doi.org/10.1109/TEVC.2007.904345

Suman, B. (2003). Simulated annealing-based multiobjective algorithms and their application for system reliability. *Engineering Optimization, 35*(4), 391–416. https://doi.org/10.1080/03052150310001597765

Suman, B. (2005). Study of self-stopping PDMOSA and performance measure in multiobjective optimization. *Computers & Chemical Engineering, 29*(5), 1131–1147. https://doi.org/10.1016/j.compchemeng.2004.12.002

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society, 57*(10), 1143–1160. https://doi.org/10.1057/palgrave.jors.2602068

Suppapitnarm, A., Seffen, K. A., Parks, G. T., & Clarkson, P. J. (2000). A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization, 33*(1), 59–85. https://doi.org/10.1080/03052150008940911

Tong, H., Huang, C., Minku, L. L., & Yao, X. (2021). Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. *Information Sciences, 562*, 414–437. https://doi.org/10.1016/j.ins.2021.03.002

Ulungu, E., Teghem, J., Fortemps, P., & Tuyttens, D. (1999). MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis, 8*(4), 221–236. https://doi.org/10.1002/(SICI)1099-1360(199907)8:4(221::AID-CDA247)3.0.CO;2-O.

Vansteenwegen, P., & Oudheusden, D. V. (2006). Developing railway timetables which guarantee a better service. *European Journal of Operational Research, 173*(1), 337–350. https://doi.org/10.1016/j.ejor.2004.12.013

Wang, H., & Jin, Y. (2020). A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE Transactions on Cybernetics, 50*(2), 536–549. https://doi.org/10.1109/TCYB.2018.2869674

Yasari, P., Ranjbar, M., Jamili, N., & Shaelaie, M. H. (2019). A two-stage stochastic programming approach for a multi-objective course timetabling problem with courses cancelation risk. *Computers & Industrial Engineering, 130*, 650–660. https://doi.org/10.1016/j.cie.2019.02.050

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271. https://doi.org/10.1109/4235.797969