Bassimir, Bernd; Wanka, Rolf

On the computation of robust examination timetables: methods and experimental results

Journal of Scheduling

Provided in Cooperation with:
Springer Nature

Suggested Citation: Bassimir, Bernd; Wanka, Rolf (2024) : On the computation of robust examination timetables: methods and experimental results, Journal of Scheduling, ISSN 1099-1425, Springer US, New York, NY, Vol. 28, Iss. 2, pp. 159-181, https://doi.org/10.1007/s10951-024-00815-y

This Version is available at:
https://hdl.handle.net/10419/323371

Journal of Scheduling (2025) 28:159–181
https://doi.org/10.1007/s10951-024-00815-y

On the computation of robust examination timetables: methods and experimental results

Bernd Bassimir[1] · **Rolf Wanka**[1]

Accepted: 11 July 2024 / Published online: 7 November 2024

Abstract
With ever-rising student numbers and an increasing shift towards more interdisciplinary study programs, the requirements for finding schedules for courses and exams become ever more complex. In real-world scenarios, the models used for calculating solutions to the course and the examination timetabling problem often must be provided to the students at the time of registration. In the field of curriculum-based course timetabling, timetables are calculated based on the structure of the study programs. For the examination timetabling problem, only a few papers focus on scheduling exams without registration data, as the requirements for exams are often more strict, or partial information is known from course registrations. In this paper we show that with the use of robustness techniques, we can also define the examination timetabling problem based on curricula. We introduce three robustness measures that address the inherent uncertainty when using the curriculum-based model. These robustness measures, along with other quality measures, are analyzed using a multi-objective simulated annealing algorithm. The results are compared on the Pareto front approximations found. We present a case study showing that, without a significant loss in solution quality, the chance is significantly reduced that rescheduling will be required after the exact numbers for the model are known.

Keywords Examination timetabling · Curriculum-based · Simulated annealing · Multi-objective optimization · Robustness

1 Introduction

At universities there are many scheduling tasks that need to be handled in each term. These can range from simple meetings that are held only once or are repeated a few times, to very complex scheduling tasks such as scheduling all exams. Each scheduling task can be summarized as the process of assigning a number of entities, e.g., exams, lectures, or meetings, to a set of resources, e.g., rooms, time slots, or specialized hardware. As the number of entities that have to be scheduled can be extensive and the additional requirements for a reasonable schedule quite complex, this scheduling often cannot be satisfactorily solved manually. This has given rise to the field of educational timetabling, including the two similar but distinct specializations called *course timetabling problem (CTTP)* and *exam timetabling problem (ETTP)*, which are sometimes grouped as university timetabling. In very simplified terms, both problems involve finding a timetable that is feasible if each lecture or exam is assigned to a time and a room, while a specified set of constraints has to be satisfied. For a recent and more detailed overview and discussion of different variants of educational timetabling, see (Ceschia et al., 2022). The difference between the CTTP and the ETTP is in the constraints that need to be satisfied and the requirement that the timetable needs to be repeatable for the CTTP, which is not the case for ETTP. However, in practice, we are not really satisfied with simply finding a feasible timetable, as this timetable might not be acceptable when evaluated under quality measures. As a consequence, these problems are usually modeled as optimization problems. In terms of complexity, both problems are NP(O)-complete, which was shown by (Even et al., 1975), as both problems are extensions of the timetabling problem proven NP-complete by the authors, or by a simple reduction from the graph coloring problem which is NP-complete as shown in (Garey & Johnson, 1979). For large problem sizes, it becomes challenging

✉ Bernd Bassimir
bernd.bassimir@fau.de

Rolf Wanka
rolf.wanka@fau.de

[1] Department of Computer Science, University of Erlangen-Nuremberg, Cauerstraße 11, 91058 Erlangen, Germany

to find good solutions in reasonable time, which has led to an extensive field of research.

A solution to the ETTP is a timetable that assigns each exam one or more rooms and a time slot such that all specified hard constraints are satisfied and which is optimal in regard to a set of specified measures. As each exam is held only once every term, this assignment is not required to be repeatable. A complete model can be found in (McCollum et al., 2012), from which our model as introduced in Sect. 2 is derived. As the ETTP is NP(O)-complete, we cannot expect to find the best solution in polynomial time. In the literature, many different approaches are used to actually solve this timetabling problem. Prominent optimization approaches include genetic algorithms (Leite et al., 2018), local search heuristics (Amaral & Pais, 2016), and swarm intelligence-based algorithms (Eley, 2007), which are often inspired by nature, that try to improve a solution or a set of solutions iteratively. On the other hand, there are also solvers that work more directly on a model of the problem, such as mixed-integer linear programming solvers (Cataldo et al., 2017) or constraint programming (Battistutta et al., 2020), which reduce the search space in each step by eliminating areas where no improvements can be found. For a comprehensive overview on examination timetabling, we refer to the surveys of (Qu et al., 2009) and (Ceschia et al., 2022). Other research has been done in bringing this abstract model closer to the requirements found in real-world timetabling scenarios, where additional requirements might be present. This often leads to new hard or soft constraints as well as other measures to be optimized that need to be addressed by the solver. Additional requirements that are commonly found for educational—or here, more specifically, university—timetabling include fairness (Muklason et al., 2017; Mühlenthaler & Wanka, 2016) or robustness objectives Akkan and Gülcü (2018) that should be fulfilled for a timetable to be acceptable. These additional requirements are more prominently found for the CTTP, as there is a significant trend towards bringing the abstract models closer to real-world requirements, while for ETTP they are addressed by only a few papers. (Yanez & Ramirez, 2003) address examination timetabling as an example of the robust graph coloring problem and describe a model that uses known probabilities for the robustness calculation. They show that their model is NP-complete and provide examples for a simple small examination timetabling problem. (Gładysz & Kuchta, 2010) model the uncertain information in regard to nonstandard student conflicts using fuzzy numbers and provide an integer linear programming model for finding robust timetables for small problem sizes.

In general, robustness approaches can be classified into several distinct categories based on how one chooses to handle the different types of uncertainty in a given model and the information that is available regarding the uncertainty. A more detailed overview of the different categories can be found in (Ben-Tal et al., 2009), with many different forms of robustness being introduced in the field of railway scheduling. If the complete scenario space, i.e., all possible uncertainty realizations, is known, or at least can be bounded, strict robustness can be used to find solutions that are feasible regardless of uncertainty. However these solutions often come at a large cost of quality in other aspects. Furthermore, if the probability distributions of the uncertain variables are known at the time of optimization, stochastic optimization approaches can be used; see (Birge & Louveaux, 2011). These approaches use the probability distribution of each uncertain variable to minimize the overall probability of a found solution being infeasible at the time the actual values of the variables become known. Another robustness category can be used if we only expect minor disruptions that cannot be fixed trivially. For such cases, (Liebchen et al., 2009) introduced recovery robustness in the area of railway optimization and (Cicerone et al., 2009) for shunting and timetabling problems. In recovery robustness, solutions are calculated that have good values in solution quality while providing simple strategies for fixing infeasible solutions after the scenario is known. For the CTTP, both (Phillips et al., 2017) and (Lindahl et al., 2019) formulate a minimal perturbation problem to find a feasible solution once the actual numbers are known. This can be used even if robustness was not addressed during the initial optimization process. (Phillips et al., 2017) show that using different neighborhoods of allowed perturbations can lead to solutions with only a few perturbations, which the scheduler can customize to adapt the process to different situations. (Lindahl et al., 2019) show that if the perturbation problem is solved directly, the solution quality is negatively impacted in a significant way. They show that increasing the number of perturbations can actually lead to overall better quality of the recovered solutions. More closely related to our approach is light robustness, which was introduced by (Fischetti & Monaci, 2009). In this approach, a slack is introduced for each constraint containing uncertain variables, and this slack is used in finding robust solutions with only a bounded deviation from a previously calculated nominal solution.

In this paper, we want to focus on such robustness aspects. In the literature, the ETTP is often described as a post-enrollment problem, where prior to the optimization process the students register for their exams, so all information is present when the actual optimization is performed. We want to focus on a fundamental deviation from this model, which in CTTP is often called curriculum-based. In this model, students do not register for their exams, or they only register after the optimization is already performed.

This situation can occur for different reasons that lead to an absence of registration data, which can all be modeled by our approach. To give an example, at the Friedrich-Alexander-

University of Erlangen-Nuremberg (FAU), students do not have to register for their courses. Restrictions are only enforced for exams, and not courses, where students can only get credits for exams that are specified by their curriculum. Therefore, there are no registration data available for the courses, and the numbers of students in a course do not correspond to the number of students that actually take the exam. This is especially prevalent in the masters programs, where students only have to accumulate a fixed number of credit points from a large selection of courses and their respective exams. Furthermore, while students have to register for their exams eventually, the university wants to provide the students with a schedule at the time of registration.

Therefore, not all information that is required in the optimization process is known. When handling such situations, many papers in the literature use the information that was acquired in the previous year and treat the optimization process as a post-enrollment problem. (Cataldo et al., 2017) directly addressed the issue of uncertainty resulting from this curriculum-based model for examination timetabling with multistage integer linear programming. In contrast to the recovery robustness approach that is mainly used when the chance for an infeasible timetable is low and only a few disturbances are expected, the uncertainty resulting from missing registration information is always present, which leads to more prevalent feasibility issues. Therefore, our goal is to find solutions that only have a very small chance of becoming infeasible even if the data used are subject to uncertainty. This paper is an extension of three conference papers (Bassimir & Wanka, 2018, 2019, 2021) where we first introduced these approaches for finding robust solutions. Here, we provide a more detailed study of these approaches where we use a multi-objective simulated annealing algorithm (MOSA) to study the trade-off between the robustness aspect and the quality of the solution based on the Pareto front, similar to the idea described by (Schöbel & Kratz, 2009), which we analyze using the hypervolume measure first introduced by (Zitzler & Thiele, 1998). We further extend the case studies presented in the conference papers by using a random instance generation approach introduced in this paper, consequently extending the number of instances that can be used when testing different approaches. For comparing different optimization strategies, there exist several sets of instances that can be used for a fair comparison between different solvers; these include the Toronto data sets (Carter et al., 1996) and the data sets used in the ITC 2007 competition (McCollum et al., 2007). There also exist some randomly generated instances and simple random generators for generating more instances to compare different solvers. Most of the time the generation process is quite simple. A large set of instances are sampled uniformly, and afterwards instances are selected by some kind of similarity measure to existing instances (Battistutta et al., 2017). In this paper, we introduce a more systematic framework for generating instances that are structurally similar to a given instance and which can be easily adapted to different requirements for the model. We use a simulation approach, where we simulate student plans, i.e., for each student all exams that the student actually takes and the semester in which the student takes the exams, based on a curriculum structure where choices are made according to random distributions that can be chosen by the researcher. These student plans are then used to derive an instance of our ETTP. We can therefore keep our structure, and only actually randomize the parts of our model that are subject to uncertainty.

The remainder of the paper is structured as follows. In Sect. 2, we define the model we used and the first part of handling the uncertainty by finding good estimations. In Sect. 2.3, we introduce two real-world instances and discuss problems that can be encountered when calculating estimations, and in Sect. 2.4 we introduce our instance generation approach. In Sect. 3, we discuss the robustness approaches we developed to find solutions that are robust in regard to the uncertainty in the model. Sections 4 and 5 introduce the solver we used and discuss experimental results we obtained with the solver when using our robustness approaches.

2 Model

Most papers about ETTP focus on the so-called post-enrollment variant of the ETTP, or at least treat it as such. Therefore, the instances that are used do not contain uncertainty. As we want to focus on a pre-enrollment variant of the ETTP, often called curriculum-based, we have to deal with uncertainty in the input data. To still attain results that are comparable to the existing approaches and to be able to use existing algorithms for solving the actual scheduling problem, we want our model to be similar to existing models. (McCollum et al., 2012) introduce an extensive model for the ETTP. Our model is in large part similar to that model, with the key difference in the associated soft and hard constraints; see Sect. 2.2.

Definition 1 An instance $I = (\mathscr{E}, \mathscr{R}, \kappa, \nu, C, \mathrm{H}, \mathrm{S}, \mathrm{TS}) \in \mathscr{D}$ of the examination timetabling problem is presented as follows.

- $\mathscr{E}$: A set of exams
- $\mathscr{R}$: A set of rooms
- $\kappa : \mathscr{R} \to \mathbb{N}$: the capacity available for each room and $\kappa : \mathscr{P}(\mathscr{R}) \to \mathbb{N}$ as the canonical extension such that for $R \subseteq \mathscr{R} : \kappa(R) := \sum_{r \in R} \kappa(r)$
- $\nu \in \mathbb{N}^{|\mathscr{E}|}$: A vector specifying for each exam how many students are attending

- Conflict matrix $C \in \mathbb{N}^{|\mathscr{E}|\times|\mathscr{E}|}$: specifying the number of overlapping students of two exams
- A set of hard constraints H and a set of soft constraints S with associated weights
- A set of available time slots TS, possibly divided into weeks, days, and periods

Note that the vector ν and the conflict matrix C are subject to uncertainty in our case.

For an instance $I \in \mathscr{D}$, we can define a valid timetable as an assignment of an exam to a time slot and a set of rooms.

Definition 2 Let $I \in \mathscr{D}$ be an instance of our model. We can define the set of valid timetables $\mathscr{T}_I$ of instance I:

$$\mathscr{T}_I := \{T \mid T : \mathscr{E} \to \mathrm{TS} \times \mathscr{P}(\mathscr{R})\}. \tag{1}$$

As we perform the scheduling before students register for their respective exams, we do not have all of the actual data available when generating the respective instances. It therefore becomes necessary to estimate parts of the instance. In this work we focus on the uncertainty regarding student data, i.e., uncertainty in ν and C. Consequently, we assume the basic structure of the instances to be fixed. We assume the rooms to be a fixed set, with no uncertainty in κ. Furthermore, we assume the set of exams to be fixed, with the set of all possible conflicts to be known. The uncertainty for conflicts in regard to feasibility is whether the conflict is active, i.e., there is at least one student attending both exams, or inactive, i.e., no student takes both exams in this term, even if taking both exams is a valid choice. We disregard the cases where a student takes two exams in which neither is specified as a valid choice by the curriculum. This structure is often provided by curriculum data which are available before the scheduling process. As the different terms, i.e., summer and winter terms, are expected to differ extensively in their exams and possible conflicts, we restrict our calculations to the same term that is repeated each year. The following calculations can be performed for each term independently.

2.1 Estimation approach

Even if scheduling is performed without the actual student data, students still often have to register for their respective exams. This provides us with two possibilities. First, we can verify whether the calculated timetable is actually feasible and use recovery algorithms to restore feasibility, if necessary, before the timetable is actually applied. Second, combined with the structure data that were used for the generation of the instance, we can further improve our estimations for the next year by using both sets of data points. In the rest of the paper we call the curriculum data the *structure instance*, which consists of a set of exams, each specifying a set of curricula, i.e., major, degree, and semester combinations, with their respective priority, i.e., mandatory or elective, that include the exam. Additionally, for each major, degree, and semester, the enrolled students ν_{curr} are given. We refer to a semester of a curriculum as the denomination of time relative to the enrollment, which is measured in terms. For example, at FAU, a bachelor's degree is regularly six consecutive terms starting at a winter term. Therefore, an exam with a curriculum specifying semester 3 should be taken by a student of this major and degree in the second winter term after enrollment. Lastly, a set of possible conflicts is given, each specifying a set of curricula that induce the conflict, i.e., that contain both exams of the conflict. We call the actual data that are attained by the students registering for the exams the *data instance*, which consists of a set of exams with the registered students split into their respective curricula and the actual conflicts with the actual overlapping student numbers split analogously. In the following paragraphs, we describe how these sets of data points of the structure instance and the data instance are used to calculate the estimations ν and C, which are an extension of the definitions first introduced in (Bassimir & Wanka, 2018).

2.1.1 Available data for estimations

Even when using robustness approaches, it is still necessary to have numbers for ν and C that can be used by the solver to find solutions for the instance. As argued before, we do not have the exact scenario and therefore do not have student and actual conflict numbers. As a result, estimation techniques are necessary to calculate a close approximation to the exact unknown scenario. The literature offers many different approaches for calculating good estimations. However, in the case of examination timetabling, the information is limited. The numbers of students do not follow a fixed distribution which we could use in the estimation, and tight bounds are often impossible to find. A common problem with many estimation techniques is that they require a large set of data points to be available for each variable to be estimated, and therefore they often cannot be successfully applied if there are only limited data available for the calculations. In Sect. 2.3 we argue why we used only 3–4 data points in the estimations. However, even in a general sense, it is often neither possible nor reasonable to use a larger number of data points for the estimation. The study programs a university offers can change, at least in part, with new study programs offered and existing ones changed to address new developments in research and industry. Furthermore, even if we assume the study programs to be fixed, there are sociological factors that influence students' choices regarding which study programs they choose and which courses are more popular. For example, in computer science there has recently been a dramatic increase in the popularity of arti-

ficial intelligence (AI) courses. Given these restrictions, we opted for a simple approach using arithmetic mean and standard deviation for the estimation, and focus on robustness to compensate for the errors made by the estimation approach.

2.1.2 Attending student numbers

As we assume the set of exams of the data and structure instances to be fixed over the different years, we can find an isomorphism between the exams of the structure instance and the data instance for each of the previous years of the given term. More specifically, we can build an isomorphic relation between the curricula for which students can attend the exam with the respective maximal number of students and the actual number of students of this curriculum that attended/canceled the exam, i.e., actually attended the exam or canceled their registration to write the exam in a different year and therefore were not in attendance. The value of 0 is used if no student of a given curriculum did attend the exam. In our case, a *curriculum* is specified by a tuple of a major, a degree, and a semester.

Definition 3 Let $\nu_{\max}(cr, e)$ be the maximal number of students of a curriculum cr that can attend an exam e, and $\nu_{\text{attend}}(cr, e)$ and $\nu_{\text{cancel}}(cr, e)$ the students of curriculum cr attending exam e and that have canceled exam e, respectively. Furthermore, Curr(e) denotes the set of curricula associated with exam e. This gives us, for each exam e, the available data points $D(e)$.

$$\forall e \in \mathscr{E} : D(e) := \{(\nu_{\max}(cr, e), \nu_{\text{attend}}(cr, e), \nu_{\text{cancel}}(cr, e)) \mid cr \in \text{Curr}(e)\}. \tag{2}$$

Note: For generality we use $\nu_{\max}(cr, e)$ for the data points, which allows us to use knowledge of upper bounds on the number of students that can attend exam e. In our case, we use $\nu_{\max}(cr, e) = \nu_{\text{curr}}(cr)$, the number of students enrolled in curriculum cr.

With the assumptions mentioned in the previous section, we can further build isomorphic relations between the data points of Def. 3 for each different year, as the basic structure is considered fixed.

Definition 4 Let $[e]$ be the ordered set of exam e in regard to the isomorphic relation of data sets $D(e)$ available for different years. $D_{cr}(e)$ denotes the element of $D(e)$ corresponding to a curriculum cr:

$$D([e]) := \left\{\{D_{cr}(e') \mid e' \in [e]\} \mid cr \in \text{Curr}(e)\right\}. \tag{3}$$

Note that the inner sets are ordered in regard to the years.

When simply using the number of students that have attended an exam in the previous years, the changes in the number of students that are enrolled in a major are disregarded. In our estimations, we instead use the arithmetic mean and standard deviation of the enrollment/cancel factors based on the data described in Def. 4, which represent a simple popularity approximation.

$$\forall e \in \mathscr{E} \forall cr \in Curr(e) : \bar{\nu}_{\text{enroll}}(cr, e) := \frac{1}{|[e]|} \cdot \sum_{e' \in [e]} \frac{\nu_{\text{cancel}}(cr, e') + \nu_{\text{attend}}(cr, e')}{\nu_{\max}(cr, e')} \tag{4}$$

$$\forall e \in \mathscr{E} \forall cr \in Curr(e) : \bar{\nu}_{\text{cancel}}(cr, e) := \frac{1}{|[e]|} \cdot \sum_{e' \in [e]} \frac{\nu_{\text{cancel}}(cr, e')}{\nu_{\text{cancel}}(cr, e') + \nu_{\text{attend}}(cr, e')} \tag{5}$$

The standard deviation $\sigma(\nu_{\text{enroll}}(cr, e))$ is calculated analogously.

With Eqs. (4) and (5) and the standard deviation, we can calculate an estimation for the number of students that will attend an exam e in curriculum cr.

$$\nu(e, cr) := \lceil (\nu_{\max}(cr, e) \cdot \bar{\nu}_{\text{enroll}}(cr, e)) \cdot (1 - \bar{\nu}_{\text{cancel}}(cr, e)) + \nu_{\max}(cr, e) \cdot \sigma(\nu_{\text{enroll}}(cr, e)) \cdot \mu \rceil. \tag{6}$$

Equations (4) and (5) provide a simple estimation for the popularity, i.e., the fraction of students we expect to enroll, and additionally the fraction of students that will afterwards cancel the registration for the exam e and curriculum cr. We multiply the number of students that are enrolled in the curriculum in the current year by the enrollment factor and afterwards subtract the students we estimated will cancel their registration. Using the parameter μ when calculating the estimations for ν, we increase the number of students in the exam by the specified factor of the standard deviation. This represents a trade-off between reducing the chance that the assigned rooms will be too small and reducing the number of feasible solutions and therefore the quality of the solutions. In our experiments, a small value was sufficient for a good estimation, as we will use robustness measures in the optimization. Using a small value for the factor μ, we mostly skew the values for exams with high deviations, while exerting very little influence on the solution space.

$$\nu(e) := \sum_{cr \in \text{Curr}(e)} \nu(e, cr). \tag{7}$$

To obtain the overall estimation for the number of students, we simply accumulate the numbers of students estimated for each curriculum in Eq. (7).

2.1.3 Conflict numbers

As with the uncertainty in the student numbers, we use estimations based on collected data for the uncertainty in the conflict matrix C. With the previous assumptions of a fixed structure, the uncertainty in the conflicts is in the number of students attending both exams. Therefore, we know what conflicts are actually possible based on the curriculum data.

In contrast to the number of students attending an exam, the estimations for the conflicts have only a limited influence on the feasibility of a solution. For feasibility, it is only relevant if a conflict has at least one student attending both exams or if no student wants to attend both. The actual number of students inducing this conflict is not relevant for feasibility; however, as we will discuss in a later section, it heavily influences the soft constraints and therefore the perceived quality of the solution.

As we have a fixed structure, we can again build an isomorphic relation for the possible conflicts of the structure instance with the maximal number of students that can induce the conflict of a curriculum and the actual number of students that attended both exams.

Definition 5 Let $C_{\max}(e_1, e_2, cr)$ be the maximal number of students of a curriculum cr that can attend exam e_1 and exam e_2, and $C_{\text{attend}}(e_1, e_2, cr)$ the actual number of students of curriculum cr attending exam e_1 and e_2. Furthermore, $\text{Curr}(e)$ denotes the set of curricula associated with an exam e and $\text{Curr}(e_1, e_2) = \text{Curr}(e_1) \cap \text{Curr}(e_2)$. This gives us the available data points $D(e_1, e_2)$ if the two exams e_1 and e_2 are in conflict, i.e., they have at least one curriculum in common:

$$\begin{aligned}&\forall e_1, e_2 \in \mathscr{E}, \text{Curr}(e_1, e_2) \neq \emptyset : \\ &\quad D(e_1, e_2) := \{(C_{\max}(e_1, e_2, cr), \\ &\quad C_{\text{attend}}(e_1, e_2, cr)) \mid cr \in \text{Curr}(e_1, e_2)\}. \end{aligned} \tag{8}$$

Note: As with Def. 3, we use $C_{\max}(e_1, e_2, cr)$ for generality. In our case this is again simply $C_{\max}(e_1, e_2, cr) = \nu_{\text{curr}}(cr)$.

This can be analogously extended with the isomorphic relation between the data points of Def. 5 for each different year.

Definition 6 Let $[\{e_1, e_2\}]$ be the ordered set of conflicts in regard to the isomorphic relation of data sets available for different years. $D_{cr}(e_1, e_2)$ denotes the entry of $D(e_1, e_2)$ corresponding to a curriculum cr and $\text{Curr}(e_1, e_2)$ as defined in Def. 5:

$$\begin{aligned}D([\{e_1, e_2\}]) := \big\{\{D_{cr}(e_1', e_2') \mid \{e_1', e_2'\} \in [\{e_1, e_2\}]\} \mid \\ cr \in \text{Curr}(e_1, e_2)\big\}. \end{aligned} \tag{9}$$

Note that the inner sets are again ordered in regard to the years.

Curriculum information often provides us with not only the maximal number of students that can attend an exam in a curriculum, but additionally the priority for the exam and, consequently, for the possible conflicts. The common distinction is between *mandatory* and *elective* exams for a curriculum. If an exam is mandatory, a student must take the exam to achieve the degree, while an elective exam is optional. Therefore, for a conflict in a curriculum between two mandatory exams, we have to guarantee that the exams are scheduled conflict-free, while a conflict between two elective exams might not be present in the actual enrollment data.

Let $[\{e_1, e_2\}]$ be the ordered set of possible conflicts in the isomorphic relation between the different years. We use the arithmetic mean and the standard deviation of the quotient of the values in $D([\{e_1, e_2\}])$ as defined in Def. 6 of each year:

$$\bar{c}(e_1, e_2, cr) := \frac{1}{|[\{e_1, e_2\}]|} \cdot \sum_{\{e_1', e_2'\} \in [\{e_1, e_2\}]} \frac{C_{\text{attend}}(e_1', e_2', cr)}{C_{\max}(e_1', e_2', cr)}. \tag{10}$$

With Eq. (10) and the analogously calculated value for the standard deviation $\sigma(c(e_1, e_2, cr))$, we can calculate the estimated value for a given possible conflict $\{e_1, e_2\}$ of a curriculum cr.

$$\begin{aligned}&c(e_1, e_2, cr) \\ &:= \begin{cases} min(\nu(e_1, cr), \nu(e_2, cr)) & \text{if } e_1 \text{ and } e_2 \text{ are mandatory for } cr \\ C_{\max}(e_1, e_2, cr) \cdot \bar{c}(e_1, e_2, cr) & \\ \quad + C_{\max}(e_1, e_2, cr) \cdot \sigma(c(e_1, e_2, cr)) & else \end{cases}. \end{aligned} \tag{11}$$

In this equation, we add the standard deviation instead of only a factor, unlike Eq. (6). In regard to feasibility, there is actually only a difference between a value of 0, i.e. the conflict is not present, and a value greater than 0, i.e. students are estimated to induce the conflict. Therefore, increasing the value does not influence feasibility but will affect solution quality directly by increasing the influence of exams with a high standard deviation. We also use the minimum of the estimated student numbers if both exams are mandatory in curriculum cr to further enforce these conflicts, as all estimated students should induce this conflict without any choice involved.

$$C(e_1, e_2) := \sum_{cr \in \text{Curr}(e_1, e_2)} c(e_1, e_2, cr). \tag{12}$$

The resulting value for the conflict between exam e_1 and exam e_2 is the sum over all curricula in Eq. (12). Note that all other values in C are 0, as there can be no conflict between these exams.

2.2 Hard and soft constraints

After the generation of the appropriate instance based on the model introduced in Def. 1, we can use an optimization algorithm to solve the actual scheduling problem. The feasibility and the quality of the generated schedule are based on the hard and soft constraints introduced in the following paragraphs.

2.2.1 Hard constraints

For a timetable to be feasible, the set of hard constraints 1–5 must be fulfilled. These are similar to the constraints specified in (McCollum et al., 2012), with the exception that we allow multiple rooms per exam, which is especially relevant for universities with very large exams or many large exams and not enough large rooms.

1. Each exam is assigned to exactly one time slot.
2. Each exam is assigned to one or more rooms.
3. Two exams that are in conflict are not scheduled at the same time.
4. No room is used by two exams at the same time.
5. The sum of capacities of the assigned rooms is larger or equal to the number of students taking the exam.

We can see that the only hard constraints that are influenced by the uncertainty are 3 and 5.

2.2.2 Soft constraints

In our reduced model we use the soft constraints *two-in-a-row*, *two-in-a-day*, and *period-spread* as defined in (McCollum et al., 2012). Other constraints are possible, but we do not have data for these available for our test instances.

1. *two-in-a-row*: If two exams are in conflict according to C they should not be assigned to two adjacent time slots on the same day.
2. *two-in-a-day*: If two exams are in conflict according to C they should not be assigned to two time slots on the same day. Note that we exclude the directly adjacent time slot to avoid double counting.
3. *period-spread*: If two exams are in conflict according to C, they should not be assigned to time slots less than λ apart.

Each violation of a soft constraint induces a penalty corresponding to the number of students that are in conflict, given by C. Given a feasible timetable, i.e., a timetable that satisfies hard constraints 1–5, we can formulate a quality measure as the weighted sum of the penalties induced by the soft constraints 1–3. As a violation of soft constraint 1 implies a violation of soft constraint 2, we exclude the adjacent time slot in the calculation of the violations of soft constraint 2. For the soft constraint 3, we ignore overlaps and compensate this in the weight. The weights of the penalties can vary for each university and therefore are not fixed in the model.

These soft constraints are all heavily influenced by the estimations for the conflict matrix C, as the actual penalty for each soft constraint is dependent on the numbers in C.

2.3 Real-world instances

Based on this model, we have generated two real-world instances taken from the Friedrich-Alexander-University of Erlangen-Nuremberg (FAU). We used the estimations (7) and (12) for the number of students and the conflicts, respectively. In practice, the fixed structure assumptions do not hold completely. Exams that are not present in the structure instance for the target year can simply be ignored, but there might be new exams. For these exams, we used the values of the data instance, as in practice the estimations for these exams would be obtained through lecturer feedback or other sources. In our instances, the number of these exams was very limited, and therefore the influence is minimal.

In practice, there are a few problems that have to be addressed. The first problem is the availability of data. There are several factors that can influence the usability of a data set. If the curriculum data are undergoing a larger restructuring, the old data might not be relevant and therefore cannot be used. For example, at the FAU, the system was restructured to a bachelor/master system in the years 2009/2010 and two cohorts starting at the same time due to a change in the Bavarian school system in 2011/2012. Other influences might also prevent the use of some data sets because they are biased by isolated circumstances, e.g., the pandemic in the years 2020/2021 which led to a temporary restructuring for online lectures.

Even without the external influence on the curriculum structure, it might be reasonable to limit the number of data points used in the estimations. The estimations are designed to at least incorporate changes in the student numbers; however, larger-scale popularity shifts might lead to overly conservative estimations, as the old data are no longer relevant.

In our experiments, for the winter term 2017 instances we used the data sets of the winter terms 2014–2016, and for the summer term 2018 instance the data sets of the summer terms 2014–2017. In the following sections we call the generated winter term instance *FAU-Winter* and the summer term instance *FAU-Summer*. Table 1 shows the numbers for the two generated real-world instances.

Table 1 Numbers for the real-world instances

	FAU-Winter	*FAU-Summer*
Exams $\|\mathscr{E}\|$	305	274
Rooms $\|\mathscr{R}\|$	13	13
Nonzero conflicts $\|C\|$	9822	8658
Possible conflicts $\|C_{all}\|$	18,836	15,856
Students	7586	6540
Time slots TS	5 weeks, 5 days, 3 periods	

2.4 Random instance generation

The approach we used for generating our real-world instances drastically limits the number of instances we can use in our experiments. In this section we will introduce a random instance generation algorithm we used in our case study to increase our test size.

As we use estimation techniques and want to focus on the robustness of the calculated solutions, we cannot simply generate a random instance which is then solved. We have two goals we want to preserve for our randomly generated instance. The first is an instance structure that has similar properties to real-world instances. The second is generating realistic scenarios for the uncertainty we might expect in our real-world instances.

In the literature, when dealing with randomly generated instances, the dominant approach is to generate completely random instances and afterwards select from a set of such generated instances the ones that are similar to the real-world instances via stochastic similarity tests. This would satisfy our first goal of randomly generated instances that have structure similar to real-world instances; however, we neither see the influence of the estimation techniques nor have a realistic scenario for the uncertainty of our model.

To fulfill both of our goals, we used a simulation-based approach for generating random instances. The primary input to our randomly generated instance generation is a structure instance and the corresponding data instance.

The structure instance provides us, for each major and degree combination $m \in M$, the set of exams that are mandatory $\mathscr{E}_{\text{mand}}(m)$ and a selection of exams that are elective $\mathscr{E}_{\text{elect}}(m)$ from which students can choose $N_e(m)$. Furthermore, we know for a major and a degree m and a semester s how many students are enrolled $\nu_{\text{curr}}(m, s)$. Let S be the ordered set of semesters in which students can be enrolled. Algorithm 1 shows the main procedure for generating the artificial history, i.e., a set of instances of the same term, one for each year, based on a structure and data instance.

Based on the data of the structure instance, i.e., the numbers for the different major and degree combinations, we generate $\nu_{\text{curr}}(m, s)$ artificial student plans for all students in each major and degree combination $m \in M$ and semester $s \in S$; see Algorithm 2.

First, the mandatory exams are scheduled according to the possible semesters in which they can or should be taken, line 6. These are simply scheduled in the first semester that is available without further checks. Afterwards, $N_e(m)$ elective exams are chosen according to the empirical distribution provided by the data instance, line 9. These are then scheduled based on their possible or recommended semester, line 11. For elective exams, the scheduling is a bit more complex. We try to balance the plans to avoid clustering in only a few semesters. Therefore, elective exams that can only be selected in a subset of semesters are scheduled in the first semester with less than 5 exams, which is university-specific. Elective exams that are possible to take in any semester are scheduled in the semester with the smallest number of exams already scheduled. Finally, we perturb the generated schedule by applying a randomization to simulate cancellation, i.e., the shift of the exam to the next semester and failure of exams, i.e., the duplication of the exam into the next semester, line 12. For this chance we use a random variable with a decreasing probability for each repeat r: for cancellation, $0.2 \cdot \frac{1}{5^r}$, and $0.2 \cdot \frac{1}{10^r}$ for failure. However, different distributions can be applied.

To generate the data instance, lines 5 and 15 in Algorithm 1, we simply attribute each student to the exams that are specified by the corresponding student plan for semester s. Conflicts are generated for each exam combination of these exams and are aggregated over all student plans.

For the rest of the instances, we iterate the generated structure and data instances. The new data for the student numbers are generated from the old data using a beta-distribution $B(5.0, 5.0)$, and the corresponding number of student plans are generated accordingly using Algorithm 2. All other numbers are shifted to the next semester and reduced by a stochastic process, with a probability according to the original differences between the semester which simulates the dropout rates for each major and degree according to the structure instance, line 11. We used $B(5.0, 5.0)$ for the calculation of the new numbers of students in the first semester, as with the given parameter it has a symmetrical probability density function in the interval [0, 1], with a mean of 0.5 and standard deviation of 0.15. As such, the new number of students in the first semester is the old number with a random symmetrical deviation of about 15%. It is possible to use other distributions for the calculation of the new number, e.g., not limited to double the number or with a skew to increasing numbers. As we use the same distribution for all majors and degrees, we did not use a skew to increasing numbers due to FAU having some majors and degrees with decreasing student numbers and others with increasing numbers. The new student plans for all other semesters than 1

are taken from the same term in the previous year and the previous semester. These are then cleaned in line 13 of plans that would be empty, i.e., student has finished all exams, and from the remaining plans $\nu^h_{\text{curr}}(m, s+1)$ are selected at random, line 14. From these plans we can generate the new data instance for the same term in the previous year, similar to line 15, and the structure instance from the previous structure instance, with only the new student numbers exchanged.

Algorithm 1: Generate random structure and data instances

input : Structure St, Data D, history length H
output: history instances $St^{1,\dots,H}$ and $D^{1,\dots,H}$

1 $St^1 := St$;
2 **for** $m \in M$ **do**
3 **for** $s \in S$ **do**
4 $SP^1(m, s) :=$ GeneratePlans$(St, m, \nu_{\text{curr}}(m, s))$;
5 $D^1 :=$ generate data instance from the student plans SP^1;
6 **for** h **from** 2 **to** H **do**
7 **for** $m \in M$ **do**
8 $\nu^h_{\text{curr}}(m, 1) := \lceil \nu^{h-1}_{\text{curr}}(m, 1) - ((2 \cdot B(5.0, 5.0) - 1) \cdot \nu^{h-1}_{\text{curr}}(m, 1)) \rceil$;
9 $SP^h(m, 1) :=$ GeneratePlans$(St, m, \nu^h_{\text{curr}}(m, 1))$;
10 **for** $s \in S : (s+1) \in S$ **do**
11 $\nu^h_{\text{curr}}(m, s+1) := \sum_{n=1}^{\nu^{h-1}_{\text{curr}}(m,s)} \mathbb{1}_{\left[r \sim U(0,1) < \frac{\nu^{h-1}_{\text{curr}}(m,s+1)}{\nu^{h-1}_{\text{curr}}(m,s)}\right]}$;
12 $SP^h(m, s+1) := SP^{h-1}(m, s)$;
13 Remove all plans from $SP^h(m, s+1)$ that have no exams in a semester $\geq s+1$ and decrease $\nu^h_{\text{curr}}(m, s+1)$ if necessary;
14 $SP^h(m, s+1) :=$ Select $\nu^h_{\text{curr}}(m, s+1)$ plans from $SP^h(m, s+1)$ at random;
15 $D^h :=$ generate data instance from the student plans SP^h;
16 $St^h :=$ generate structure from the numbers ν^h_{curr};
17 **return** $St^{1,\dots,H}$, $D^{1,\dots,H}$

Algorithm 2: Generation of student plans for the major and degree combination m

1 **function** *GeneratePlans*(St, m, n)**:**
  input : Structure Instance St=(mandatory exams $\mathscr{E}_{\text{mand}}(m)$, elective exams $\mathscr{E}_{\text{elect}}(m)$, number of elective exams $N_e(m)$), major and degree m, number of plans to generate n,
  output: Student plans $SP(m)$ for major and degree m
2 $SP(m) := \emptyset$;
3 **for** u **from** 1 **to** n **do**
4 $SP_u := \emptyset$;
5 **for** $e \in \mathscr{E}_{mand}(m)$ **do**
6 Schedule(SP_u, e, m);
7 $ES := \emptyset$;
8 **for** i **from** 1 **to** $N_e(m)$ **do**
9 $e :=$ SelectExam$(\mathscr{E}_{\text{elect}}(m) \setminus ES)$;
10 $ES := ES \cup \{e\}$;
11 Schedule(SP_u, e, m);
12 perturb(SP_u);
13 $SP(m) := SP(m) \cup \{SP_u\}$;
14 **return** $SP(m)$

Table 2 Numbers for two randomly generated instances from the history generated using Algorithm 1

	Random-winter	*Random-summer*
Exams $\lvert\mathscr{E}\rvert$	309	277
Rooms $\lvert\mathscr{R}\rvert$	13	13
Nonzero Conflicts $\lvert C\rvert$	9654	8636
Possible Conflicts $\lvert C_{all}\rvert$	18,351	15,696
Students	6784	5026
Time slots TS	5 weeks, 5 days, 3 periods	

These generated structure and data instances can then be used to generate an instance $I \in \mathscr{D}$ of our model according to the estimations introduced in Sect. 2.1.

Table 2 shows the values for two randomly generated instances we used in our experiments, using our random instance generation approach introduced in this section.

3 Robustness

As mentioned in Sect. 2.1, we use estimations for the values of the number of students attending an exam ν and for the number of students inducing a conflict C. Using estimations, we inherently must deal with uncertainty in the input data to our scheduling algorithm.

There are a few ways to deal with this uncertainty. The first possibility is to improve the estimations to a point that the deviation between the estimations and the actual numbers in the data instance is negligible. However, as our uncertainty comes from students choosing exams, the uncertainty does not follow a fixed distribution, and therefore we cannot bound the deviations realistically. Another approach is to deal with the inherent uncertainty of the model and the deviations from the calculated estimations. In the following sections we discuss how the optimization algorithm can try to improve the robustness in regard to the uncertainty of the returned solution.

3.1 Robustness of student numbers per exam

In contrast to the CTTP, the hard constraint for the capacity assigned to an exam in the ETTP cannot be relaxed, as each student must have a seat to write the exam. This is enforced via the hard constraint 5. However, as we have uncertainty in the number of students attending an exam, there might be more students registered than estimated, and therefore the room capacity assigned to the exam can be too small. Therefore, even if the solution based on the estimations is feasible, the solution when using the data attained by the students registering for their exams might become infeasible.

If we calculate a timetable based on our estimations, the assigned room capacity has to be larger than the estimated number of students. This is enforced by the hard constraint 5. When using this timetable, we can observe three possible scenarios for an exam:

1. The actual number of students in the exam is smaller than the estimated number and therefore smaller than the assigned room capacity.
2. The actual number of students in the exam is larger than the estimated number of students, but still smaller than the assigned room capacity.
3. The actual number of students in the exam is larger than the estimated number of students and also larger than the assigned room capacity.

The first scenario is the least problematic, as hard constraint 5 enforces only an upper bound on the number of students of the exam; the timetable is still feasible when evaluated with the actual number of students. The drawback is that the assigned rooms are not completely filled and the timetable might be inefficient in regard to room utilization. In the second scenario, we underestimated the number of students, but the actual number of registered students is still smaller than the assigned room capacity. In this scenario there are still enough seats in the assigned rooms for all registered students, i.e., hard constraint 5 is not violated and the timetable is still feasible. In the last scenario, the actual number of students exceeds the assigned room capacity and therefore violates hard constraint 5, which leads to an infeasible timetable.

As we only have estimations at the time of scheduling, we cannot prevent the last scenario from occurring without a drastic increase in estimated numbers, which would lead to infeasible instances at scheduling time. To deal with this scenario, we use robustness techniques to reduce the chance of scenario 3 occurring.

To achieve this, we use a slack-based approach in the form of a new robustness measure we first introduced in (Bassimir & Wanka, 2018) and improved in (Bassimir & Wanka, 2019). As we have discussed before, if we overestimate the number of students, the only real downside is that we decrease the utilization of our rooms. However, as this can be improved after registration without any real changes to the schedule by removing excess rooms, it is only of secondary concern. The basic idea of the objective based on this robustness measure is to increase the slack, i.e., the difference between the estimated number of students and the assigned room capacity. It is sufficient for feasibility to have more room capacity than registered students assigned to an exam. Therefore, we do not try to maximize the slack for each exam, but rather try to improve a lower bound on the slack. For this, we define a measure that is shown in Eq. (RM1).

Definition 7 Let $I \in \mathscr{D}$ be an instance of our model and $T \in \mathscr{T}_I$ be a valid timetable for instance I. Let $RP(e, T)$ denote the room pattern assigned to an exam e in timetable T, and $\nu(e, I)$ be the number of students attending exam e in instance I. We define the student robustness measure $SR : \mathscr{D} \times \mathscr{T}_I \to \mathbb{Q}^+$ as

$$SR(I, T) = \min_{e \in \mathscr{E}} \left\{ \frac{\kappa(RP(e, T))}{\nu(e, I)} \right\}. \tag{RM1}$$

This differs from the robustness measure introduced in (Bassimir & Wanka, 2018) insofar as the quotient is inverted and minimized instead of maximized. The reason for this change is that in this new form, the measure is linear in T, and therefore an objective based on this form of the robustness measure can easily be linearized for an integer linear programming solver.

3.2 Robustness of conflict numbers

When using estimations for the conflict numbers, we can identify two major influences of this number on the model. The first is a binary decision as to whether a conflict should be considered in the model and therefore enforced via hard constraint 3. The second influence pertains to the soft constraints, which are dependent on the actual estimated number. In this section we look at both influences and discuss robustness approaches for dealing with them.

As discussed in Sect. 2.1, the curriculum information provides us with the priority of a possible conflict in regard to a curriculum. If two exams that have overlapping curriculum lists have at least one curriculum in which both exams are mandatory, we can always assume that there are students that induce this conflict in the data instance. Therefore, these conflicts are always present and have to be considered for hard constraint 3. In the rest of the paper we call this type of conflict a *mandatory conflict*. As for conflicts between exams that have overlapping curriculum lists with no curriculum in which both exams are mandatory, this is not the case. These conflicts depend entirely on the choices of the individual students. We call these conflicts *elective conflicts*, as they always depend on students electing to take both exams or the one that is elective if one exam is mandatory for the curriculum. In the rest of the paper, when considering robustness in regard to conflicts, we restrict the discussion to these elective conflicts.

The simplest solution to this uncertainty is the strict robustness approach. For this we consider every possible conflict to be active and enforce all elective conflicts through hard constraint 3. This solves the problem of feasibility in regard to all possible uncertainty scenarios. However, depending on the size of the elective portfolio a university provides its students and the possible restrictions on the choices a student has to respect, this approach can lead to a completely infea-

sible instance. In contrast, the completely opposite approach that is often used when dealing with elective courses/exams is to accept that conflicts might occur. Therefore, the problem of infeasible instances is solved; however, this approach leads to complaints and consequently nonacceptance of the solutions by the student body.

An obvious compromise is to use the estimations for the conflicts. These provide us with the knowledge as to whether a conflict was active in previous years. Therefore, if a student has chosen both exams, we might expect to have other students making similar choices. The main benefit is that we can reduce the set of conflicts we have to consider to a manageable size. Furthermore, we can expect to inherently address some correlation between exams that are possible to take in the same year but which, due to similar topics with one exam covering the more advanced subjects, might be beneficial to take in a specific order. This can lead to the conflict never being an actual issue in the data instance.

We are presented with several issues we have to address when using estimations. First and foremost, we use several years of data for the calculations of the estimations. As we only check if at least one student has chosen two possible conflicting exams in the past, it is quite possible that for larger majors, each choice will be made at least once. Therefore, most conflicts might be active, and consequently the instance might become infeasible. The second issue we have to address is the issue of bias. If the estimation for a conflict of our instance is that no student takes both exams even if the conflict is possible, we might schedule both exams at the same time without incurring any penalty. In this case, no student will register for both exams, even if the student wants to take both. In the next year the estimation will again be that no student wants to take both exams. Thus, a conflict that would actually be present in the data would not be addressed, and consequently the acceptance of the student body will decrease without the person managing the scheduling process actually noticing the issue.

Instead of enforcing all possible elective conflicts that have estimated students via hard constraint 3, we introduce a new robustness measure (RM2) that, when minimized as an objective, reduces the estimated number of students inducing conflicts. Note that a student can induce more than one conflict and is counted for each induced conflict.

Definition 8 Let $I \in \mathscr{D}$ be an instance of our model, with C_I being the conflict matrix of I and $T \in \mathscr{T}_I$ a valid timetable for instance I; we define the conflict robustness measure $CR_1 : \mathscr{D} \times \mathscr{T}_I \to \mathbb{N}_0$ as

$$CR_1(I,T) = \sum_{t \in T} \sum_{\substack{e_1,e_2 \in t \\ e_1 \neq e_2}} C_I(e_1,e_2). \tag{RM2}$$

Using this robustness measure to try to enforce the elective conflicts instead of hard constraint 3, we solve the first issue of possible infeasible timetables. However, the second problem of bias is still present, as only conflicts with at least one estimated student are considered.

To solve the issue of bias, we extend the robustness measure (RM2) with a lower bound of 1.

Definition 9 Let $I \in \mathscr{D}$ be an instance of our model, with C_I being the conflict matrix of I and $T \in \mathscr{T}_I$ a valid timetable for instance I; we define the conflict robustness measure $CR_2 : \mathscr{D} \times \mathscr{T}_I \to \mathbb{N}_0$ as

$$CR_2(I,T) = \sum_{t \in T} \sum_{\substack{e_1,e_2 \in t \\ e_1 \neq e_2}} \max\left\{ C_I(e_1,e_2), \mathbb{1}_{\left[\substack{Curr(e_1,e_2) \neq \emptyset \ \wedge \forall cr \in Curr(e_1,e_2): \\ e_1 \text{ or } e_2 \text{ elective exam in } cr}\right]} \right\}. \tag{RM3}$$

Using the maximum in robustness measure (RM3), each possible conflict as specified in the structure instance has a penalty of at least 1. Even if a conflict is inactive, i.e., the estimated number of students is 0, the conflict will be considered with a low priority. Minimizing the robustness measure (RM3) as an objective not only provides an increase in the robustness of the found solution by reducing the number of estimated elective conflicts, but also increases the chance of an inactive conflict still being considered and the corresponding exams not being scheduled at the same time. This provides us with an exploratory component which enables us to find previously unrecognized conflicts and reduces the chance for bias in the solutions.

4 Scheduling algorithm

In this section we will introduce the scheduling algorithm we used for evaluating the influence of the introduced robustness measures on the quality of the solutions and the objective function used.

4.1 Optimization function

As the goal of this paper is to investigate the influence of the robustness measures on the quality of the solution, we use a multi-objective algorithm. We first must define the objective function used by our multi-objective algorithm.

Definition 10 Let $I \in \mathscr{D}$ be an instance of our model, and $T \in \mathscr{T}_I$ be a valid timetable for instance I. Furthermore, let $P_{tr}(I,T)$, $P_{td}(I,T)$ and $P_{ps}(I,T)$ be the penalties incurred by violations measured in the number of students given by instance I of the soft constraints 1, 2, and 3 in timetable T, respectively. ω_1, ω_2, and ω_3 are weights for the linearization

of the soft constraint measures. We define the quality measure $Q : \mathscr{D} \times \mathscr{T}_I \to \mathbb{N}_0$ as follows.

$$Q(I,T) = \omega_1 \cdot P_{tr}(I,T) + \omega_2 \cdot P_{td}(I,T) + \omega_3 \cdot P_{ps}(I,T). \tag{13}$$

With Q and the robustness measures SR, CR_1, and CR_2 as defined in (RM1)–(RM3), we can define the actual fitness function $F : \mathscr{D} \times \mathscr{T}_I \to \mathbb{N}_0 \times \mathbb{Q}^- \times \mathbb{N}_0$ as a 3-valued function that is to be minimized.

$$F(I,T) = (Q(I,T), -SR(I,T), \{CR_1(I,T), CR_2(I,T)\}). \tag{14}$$

Note that the second objective value is negative, as we have a minimization problem. An objective using the robustness measure defined in Eq. (RM1), however, should be a function to be maximized. Furthermore, the third value is a choice between the two introduced conflict robustness measures, which is fixed for a specified run.

To reduce the number of dimensions, we use the weighted sum of the soft constraint violations, which gives us the quality of a solution in regard to student acceptance. We used $\omega_1 = 200$, $\omega_2 = 100$, and $\omega_3 = 1$ with a period spread of 4 in our experiments. The other dimensions have no direct impact on the perceived quality of the solution; instead they represent the robustness of the solution in regard to the uncertainty of the number of registered students and the induced conflicts.

4.2 Multi-objective simulated annealing algorithm

For the optimization algorithm, we used a MOSA algorithm (see Algorithm 3), first introduced in (Suppapitnarm et al., 2000) with an archive of non-dominated solutions, which gives us an approximation of the Pareto front.

During the optimization process, an archive of non-dominated solutions A is maintained. Every Δ_r iterations, a decreasing interval of iterations (line 13 in Algorithm 3)—we used 100 and 0.95 for r_{cut} and r_{red}, respectively, and 20,000 for the initial value—a solution of this archive is selected (see Algorithm 5), and the normal SA algorithm is executed. To select a new solution from the archive, the archive is first sorted according to an isolation metric (line 7 in Algorithm 5) and a solution is then drawn uniformly at random (u.a.r.) from the first s entries of the sorted archive, additionally guaranteeing that the best solutions can always be selected. In our experiments we used $\alpha = 1.0002$ as a factor for the length of the selection from the archive.

In every other iteration, a solution is selected from the neighborhood of the current solution; see Sect. 4.2.1. If the new solution is not dominated by any of the solutions in the archive, it is immediately accepted, all dominated solutions in the archive are removed, and the algorithm continues from this solution (line 18 in Algorithm 3). Otherwise, it is accepted with a probability corresponding to the current temperature or it is discarded (line 22 in Algorithm 3).

Algorithm 4 shows the cooling scheme which is used by the MOSA algorithm. In the first ρ_{start} iterations, we used 8000 for this parameter. The temperature is at ∞, so all solutions are simply accepted. Afterwards, the temperature

Algorithm 3: MOSA-Algorithm for the ETTP

```
   input     : Instance I, maximal number of iterations n, solution
               dimensions D, fitness function F
   parameter: cooling start ρ_start, cooling interval Δ_ρ, restart
               interval Δ_r, restart cutoff r_cut, restart reduction r_red
   output    : Pareto front of solutions A
1  T := initial_solution(I);
2  A:={T};
3  ∀d ∈ {1,...,D} : ρ_d:=∞ ;
4  accepted :=∅;
5  r:=0;
6  for i from 0 to n do
7      if i mod Δ_ρ = 0 ∧ i > ρ_start then
8          ρ:= reduce_temperature(ρ, accepted, D);
9          accepted :=∅;
10     if i mod Δ_r = 0 then
11         T := select_from_archive(A, r, F, D);
12         r:=r + 1;
13         Δ_r:= max{r_cut, Δ_r · r_red};
14     T' := select_from_neighborhood(I, T);
15     if non_dominated(T', A) then
16         A:=A ∪ {T'};
17         accepted := accepted ∪ {T'};
18         T := T';
19     else
20         if probability_accept(T', ρ) then
21             accepted := accepted ∪ {T'};
22             T := T';
23 return A
```

Algorithm 4: Function for reducing the temperature used by the MOSA algorithm

```
1  function reduce_temperature(ρ, accepted, D):
       input     : temperature vector ρ, list of accepted solutions
                   accepted, solution dimensions D
       parameter: temperature reduction cutoff ρ_cut, temperature
                   reduction factor ρ_red
       output    : temperature vector ρ
2      if ρ = ∞ then
3          for d from 1 to D do
4              ρ_d:=5 · σ(accepted_i);
5      else
6          for d from 1 to D do
7              ρ_d:=ρ_d · max{ρ_cut, e^(−ρ_d·ρ_red / (2·σ(accepted_i))) − 0.2};
8      return ρ
```

Algorithm 5: Function that selects a solution from the archive to be used after a restart

1 **function** *select_from_archive(A, r, F, D)*:
  input : archive A, restart counter r, fitness function F, solution dimensions D
  parameter: selection factor α
  output : selected solution T'
2 **for** d **from** 1 **to** D **do**
3 $worst_d := \text{argmax}_{a \in A}\{F(a)_d\}$;
4 $best_d := \text{argmin}_{a \in A}\{F(a)_d\}$;
5 $s := \max\{(|A| \cdot (\alpha^r - 1)) - |\bigcup_{d \in \{1,\dots,D\}} best_d|,\ 0\}$;
6 **for** $T \in A$ **do**
7 $\xi(T) := \sum_{a \in A \setminus \{T\}} \sum_{\substack{d \in \{1,\dots,D\} \\ F(worst_d)_d - F(best_d)_d > 0}} \left(\frac{F(a)_d - F(T)_d}{F(worst_d)_d - F(best_d)_d}\right)^2$;
8 *sorted_archive* := sort $\left(A \setminus \left(\bigcup_{d \in \{1,\dots,D\}} best_d\right)\right)$ according to ξ decreasing;
9 *archive_selection* := $\bigcup_{i<s}$ sorted_archive$_i$ $\cup \bigcup_{d \in \{1,\dots,D\}} best_d$;
10 T':= select solution u.a.r. from *archive_selection*;
11 **return** T'

is set to five times the standard deviation of all solutions accepted since the last update (line 4). After ρ_{start} iterations, a reduction of the temperature is done for each dimension individually every Δ_ρ iterations. We used 4000 for this interval. The reduction is limited with a cutoff value ρ_{cut}, which we set to 0.5 in our experiments. An exponential function is used in the reduction, where the old value is reduced by the factor ρ_{red}, which we set to 0.95, and divided by two times the standard deviation of all solutions accepted since the last update (line 7).

The parameters specified in this section were selected through an experimental survey and showed good results across all robustness and instance variants, which we discuss in Sect 5.

4.2.1 Neighborhood

In our experiments, we use the Kempe-exchange neighborhood to select a new solution, unless a restart is done. Using Kempe-exchanges instead of the commonly used swap operation, i.e., switching time slots between two exams, or move operation, i.e., moving an exam to a different time slot, ensures that each neighbor of a feasible solution is again feasible in regard to the conflicts. Kempe-exchanges, or Kempe-changes, were first defined by (Kempe, 1879) in his proof of the four-color theorem, which even though the proof turned out to be incomplete are helpful in recoloring. A new solution is calculated from the old solution by selecting an exam e (line 3 in Algorithm 6) with its corresponding time slot t and a second different time slot t' u.a.r. from the remaining time slots. The conflict graph induced by the two time slots is a bipartite graph. On this sub-graph we calculate the connected components. The connected component of the

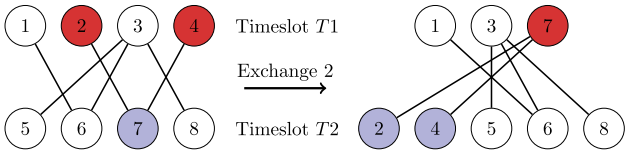


Fig. 1 Kempe-exchange of the connected component of selected exam 2. The upper row shows the exams in time slot $T1$ and the lower row the exams in time slot $T2$ before and after the exchange. The bipartite subgraph induced by the two time slots is 2-colored. Only the connected component that is selected by exam 2 is colored in the two colors in this example for clarity (Color figure online)

chosen exam e is selected, and all exams in the same connected component are swapped between the two time slots, i.e., all exams of the connected component in t are swapped to t' and all exams of the connected component in t' are swapped to t (line 7). Figure 1 shows an example of this operation with 2 as the selected exam e, $t = T1$, and $t' = T2$. This exchange cannot induce new conflicts, if the solution was feasible in regard to the conflicts. As the solution was feasible, the graph induced by t and t' is guaranteed to be bipartite. If we want to swap exam e from t to t', we have to swap all exams of t' that are in conflict with e. Therefore, exam e does not induce a conflict in t', but this does not hold true for the exams we swapped from t' to t. However, if we swap the complete connected component, this cannot occur, as all exams from t that would induce conflicts with exams swapped from t' are swapped to time slot t', and the reverse is also true.

However, this only holds true for the conflicts, and in our model we also consider the room assignment sub-problem. If we swap all exams of the connected component, it is possible that no room assignment exists for one of the changed time slots. In such a case we discard the calculated neighbor and continue with solution T in the next iteration.

4.2.2 Room assignment

After exams are swapped as discussed in 4.2.1, we have to calculate new rooms for each exam in the two time slots. For feasibility it is sufficient to calculate an assignment of exams to rooms such that the estimated number of students for the exam does not exceed the assigned room capacity. However, as discussed in Sect. 3.1, the number of students is subject to uncertainty. We use Eq. (RM1) to measure the robustness of a timetable in regard to this uncertainty. This measure is directly influenced by the room assignment. Therefore, we use an optimization algorithm that returns a room assignment that maximizes this property for a given time slot.

In our experiments, we use a heuristic for the assignment of rooms for the given exams of a time slot developed by (Häckler, 2019) in his bachelor thesis (see Algorithm 7 and 8), as the room assignment problem with multiple rooms per

Algorithm 6: Function for selecting a solution from the neighborhood of a given solution

1 **function** *select_from_neighborhood*(I, T)**:**
  input : Instance I, Solution T
  output : New solution T'
2 T':=T;
3 e:= $\begin{cases} \operatorname{argmin}_{e \in \mathscr{E}} \left\{ \frac{\kappa(RP(e,T))}{\nu(e,I)} \right\} & \text{with probability } 0.1 \\ e \in \mathscr{E} \quad \text{u.a.r.} & \text{otherwise} \end{cases}$;
4 t:= time slot of e;
5 t':= select time slot u.a.r. from the remaining time slots;
6 calculate connected components of bipartite Graph induced by t and t' according to I;
7 swap all exams of connected component that includes e between t and t' in T';
8 calculate_room_assignment(I, t) in T';
9 calculate_room_assignment(I, t') in T';
10 **if** *room assignments exist* **then**
11 **return** T'
12 **else**
13 **return** T

exam is NP(O)-complete. This can be proven by a reduction from 3-partition.

Algorithm 7: Heuristic for calculating a new room assignment for a given time slot maximizing the slack

1 **function** *calculate_room_assignment*(I, t)**:**
  input : Instance I, Time slot t
  output : Room assignment $RP'_t : E_t \to \mathscr{P}(\mathscr{R})$
2 E_t:= Exams of Time slot t;
3 **for** $e \in E_t$ **do**
4 $\nu'(e)$:=$\nu(e, I)$
5 RP'_t:= greedy_assign(I, E_t, ν');
6 r:=0;
7 **while** *True* **do**
8 *end_round* := True;
9 **for** $e \in E_t$ **do**
10 **if** $\nu'(e) < \nu(e, I) \cdot 1.3^r$ **then**
11 $\nu'(e)$:=$\nu'(e) + 5$;
12 *end_round* := False;
13 **if** *round_end* = *True* **then**
14 r:=$r + 1$;
15 continue;
16 $\hat{RP}_t$:= greedy_assign(I, E_t, ν');
17 **if** $\hat{RP}_t$ *is room assignment* **then**
18 RP'_t:=$\hat{RP}_t$;
19 **else**
20 **return** RP'_t

The basis of the algorithm is a greedy room assignment strategy that returns a solution to the assignment problem without considering the quality of the assignment; see Algorithm 8. The greedy algorithm handles each exam iteratively

Algorithm 8: Greedy algorithm for the room assignment problem

1 **function** *greedy_assign*(I, E_t, $\nu' \in \mathbb{N}_0^{|E_t|}$)**:**
  input : Instance I, Exams E_t, Student numbers $\nu' \in \mathbb{N}_0^{|E_t|}$
  output : Room assignment $RP'_t : E_t \to \mathscr{P}(\mathscr{R})$
2 sort E_t non-increasing in ν';
3 sort $\mathscr{R}$ non-increasing in κ;
4 **for** *unassigned rooms remaining* $\wedge$ *unassigned exams remaining* **do**
5 e:= largest unassigned exam in E_t;
6 $RP'_t(e)$:=$\{R\}$ with R largest unassigned room in $\mathscr{R}$;
7 **if** $\kappa(RP'_t(e)) \geq \nu'(e)$ **then**
8 continue;
9 **if** $\exists R$ *unassigned* : $\kappa(R) > (\nu'(e) - \kappa(RP'_t(e)))$ **then**
10 $RP'_t(e)$:=$RP'_t(e) \cup \{R'\}$ with R' smallest unassigned room with $\kappa(R) > (\nu'(e) - \kappa(RP'_t(e)))$;
11 **while** $\kappa(RP'_t(e)) < \nu'(e)$ *and unassigned rooms remaining* **do**
12 $RP'_t(e)$:=$RP'_t(e) \cup \{R'\}$ with R' smallest unassigned room;
13 **if** *All exams have sufficient capacity assigned* **then**
14 **return** RP'_t
15 **else**
16 **return** *Failure*

in non-increasing order. It starts with assigning to the exam the largest room that is not yet assigned to a different exam (line 6). If this room is not sufficient, we try to find a room that is large enough to hold the remaining students. If such a room exists, we use the smallest such room to keep as much room capacity as possible for other exams (line 10). If no such room exists, we assign rooms in non-decreasing order of capacity until the exam has sufficient rooms (line 12). The heuristic increases the number of students in the exams considered until the greedy room assignment strategy can no longer find a feasible solution (line 16ff in Algorithm 7). The increase in the number of students is proportional to the number of students in the unperturbed exam (line 10 in Algorithm 7). However the increase is not done in one step; instead, we increase the number of students of each exam in rounds. One round is finished if all exams have more students than the corresponding relative increase in the round (line 10 in Algorithm 7). During a round, the number of students is increased by a fixed amount (line 11 in Algorithm 7) to find solutions closer to the unknown theoretical bound. This algorithm therefore returns an approximate solution in regard to the robustness measure in Eq. (RM1), as the slack simulated by the increase in students per exam is maximized.

5 Results

5.1 Experimental setup

In this section we present the results obtained in regard to the robustness and quality of the solutions found by the MOSA algorithm for different instances I. The MOSA algorithm returns a set of feasible solutions that are all mutually non-dominated. This set of feasible solutions is an approximation of the unknown Pareto front defined by F.

5.1.1 Evaluation layers

To evaluate the robustness of the solutions and the influence of said robustness on the quality of the solutions returned by the MOSA algorithm, we define two different evaluation layers of such a solution. The first layer, which is used when calculating the solution set, is defined on the instance $I_{\text{uncertain}}$ of our model which uses the estimated values for the number of students and the conflicts as introduced in Sect. 2.1. We call this layer *objective space*, as it is used by the MOSA algorithm in the calculation of the fitness of a possible solution and is defined by the objective function $F(I_{\text{uncertain}}, T)$. Thus, the result of the MOSA algorithm is an approximation of the fitness function for instance $I_{\text{uncertain}}$.

The second layer is not used in the optimization process, as it is not available at the time of optimization. However, it allows us to evaluate a given solution in regard to robustness measures. This layer is called *scenario space*. Instead of using the estimations for the number of students and the conflicts, we use the values of the corresponding data instance I_{scenario}. The values of $F(I_{\text{scenario}}, T)$ provide us with a representation of the scenario when implementing the given solution as a timetable for the exams. Evaluating a solution in the scenario space, we verify whether the solution would have been feasible and can measure the actual quality of the solution when used in the given year. In our experiments, for each instance we generated and use as $I_{uncertain}$, i.e., *FAU-Summer*, *FAU-Winter*, *Random-Summer*, and *Random-Winter*, we use the corresponding data instance as I_{scenario}, i.e., for the real-world instances the data instances for the years 2018 and 2017, respectively, and the data instances D^H which were generated by our random instance generation algorithm for the randomly generated instances.

However, when evaluating the solution set in the scenario space, the resulting vectors do not form an approximation of the Pareto front in the scenario space. As we want to use Pareto front measures to compare the performance of the robustness measures, we need an approximation to the Pareto front in the scenario space. To achieve this we calculate two subsets of the given solution set. The first set includes all solutions that are non-dominated in the scenario space, and the second set includes all solutions that do not dominate other solutions. As both solution sets are mutually non-dominating, they are approximations of the Pareto front in the scenario space. The solution set including all non-dominated solutions is therefore a *lower bound*, and the solution set containing all non-dominating solutions an *upper bound* on the solution set, as we have a minimization problem.

5.1.2 Instance constraint variants

We furthermore differentiate between two different instance constraint variants for our model in regard to conflicts between elective exams. In the first instance constraint variant we call *NonZero*, all elective conflicts with estimated students inducing the conflict are enforced via hard constraint 3. In the second instance constraint variant called *NoElective*, no elective conflict is enforced with hard constraint 3.

5.1.3 Pareto measure

As the optimization process is subject to randomization, we have to deal with variance in our results. For multi-objective optimization algorithms, this is not as simple as for single-objective algorithms. Several approaches can be found in the literature to compare different Pareto fronts. The two major categories are unary and binary measures. Binary measures directly compare two Pareto fronts in regard to a specified relation. This can be successfully used to compare the results of two runs of the algorithm. However, if we want to use several runs for each variant we introduced and provide aggregated comparisons between these variants to account for the variance in the results, this method is ill-suited. The unary Pareto measures assign each Pareto front a real value, and comparisons are done on these values. These can be more easily aggregated, and are therefore more suited to the analysis we provide. We therefore use the unary Pareto measure called *Hypervolume indicator*, which is often used for comparing multi-objective evolutionary algorithms in the literature, on the introduced Pareto front approximations. Using this measure, we calculate the mean and standard deviation of the hypervolumes of the three Pareto fronts discussed in this section, obtained by several runs on the same instance $I_{uncertain}$ using the algorithm proposed by (While et al., 2006). As the range of values of the different objectives are quite distinct, we normalize the values of each objective in respect to a reference point and use (1.1, 1.1, 1.1) as our new reference point.

5.1.4 Optimization time

All our experiments were performed on a 64-bit Linux machine running Ubuntu with an 8-core 3.8 GHz Intel Core i7-4770 processor and 31.3 GB memory. One run of the introduced MOSA algorithm, implemented in Common Lisp,

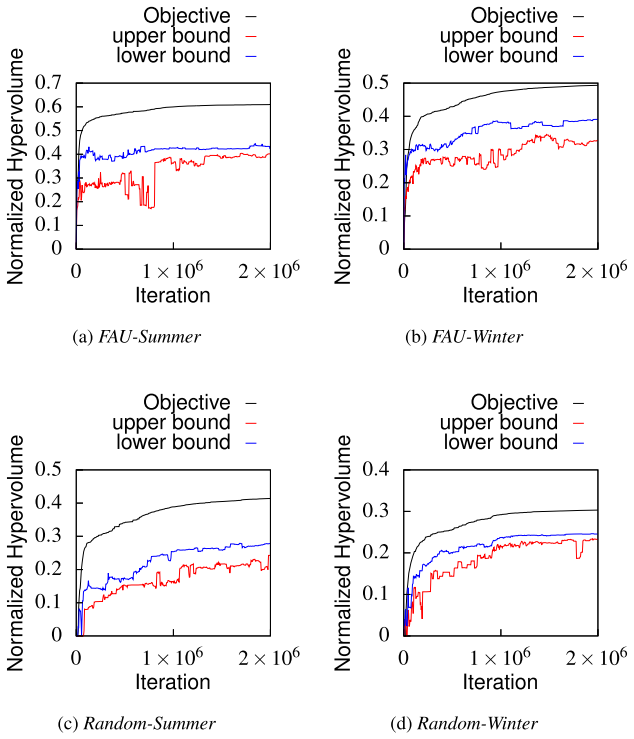


Fig. 2 Hypervolume in the optimization process for the *FAU-Summer* and *FAU-Winter* instances and two random instances

performing $2 \cdot 10^6$ iterations for one variant and instance, took about 1 hour in regular normal operations on this computer system.

5.2 Impact of robustness scenario instance

In the first set of experiments, we study the influence of the robustness constraints on the progress of the optimization process. To show that the robustness constraints actually influence the solutions in the scenario space, we calculated the hypervolume normalized to the reference point on the archive and the lower and upper bounds in the scenario space. We used the same reference point for the lower and upper bounds in the scenario space, but a different one for the archive. Therefore, the actual values of the hypervolume in the objective space and the scenario space are not directly comparable; however, the changes in the values can be compared.

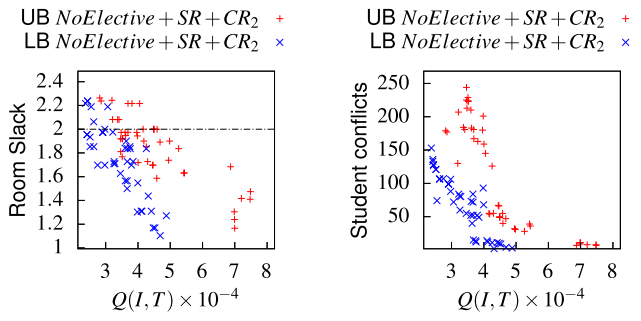
In Fig. 2 we show the changes in the normalized hypervolume in the objective and scenario space in the course of a run of our optimization algorithm on the *FAU-Summer* and *FAU-Winter* instances, as well as on two randomly generated instances *Random-Summer* and *Random-Winter* using our random generation approach. One can see that the hypervolume of both the lower and upper bounds, while subject to fluctuation, follows the graph of the hypervolume in the objective space. The same can be observed for the randomly generated instances using the approach introduced in Sect. 2.4 and shown for two such instances in Fig. 2. A key observation on the random instances is that the fluctuations in the hypervolume of the lower and upper bounds are smaller. This can be attributed to the outliers that are found in the real-world instances and are not as present in the randomly generated instances. We can conclude that the robustness constraints actually influence the solution found, such that improvements in the robustness constraints reduce the influence of the uncertainty and lead to fewer actual conflicts in the scenario. However, as shown by the fluctuations, large errors in the estimations can still detrimentally influence the quality of the solutions.

5.3 Influence on the solution quality

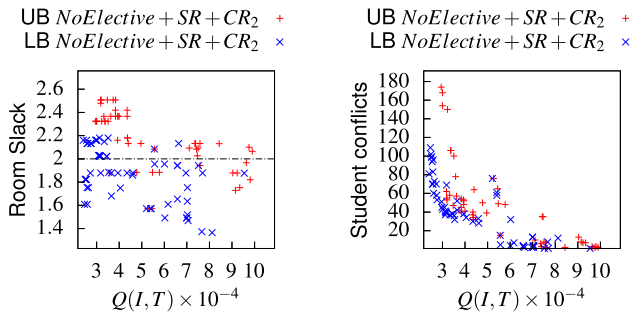
By considering the robustness of solutions in the optimization process, it is often the case that the robustness of a solution is a trade-off with respect to the quality of the solution. In robustness optimization, this is called the cost of robustness.

As we use robustness measures SR and CR_1 or CR_2 in the objective function of our optimization algorithm, we can directly observe the influence of these measures by studying the approximations of the Pareto front returned by our multi-objective algorithm. Similar to the first experiment, we look at the results in the scenario space which do not directly translate to an approximation of a Pareto front. Therefore, we again observe the bounds of the solutions, i.e., the lower and upper bounds, as previously defined.

In Fig. 3 we plotted the 2D projections of the 3D Pareto front returned by the MOSA algorithm of a single run on the *FAU-Summer* and *FAU-Winter* instances, and in Fig. 4 for two randomly generated instances using instance constraint variant *NoElective* and robustness measures SR and CR_2 for robustness. In the first two pictures of Fig. 3 we can observe the robustness cost for our real-world instance *FAU-Summer*. In the case of SR, we can see that the relative slack can be reduced to 1.6 for all solutions, with a drop in the solution quality only from about 30,000 to 60,000. Note that we calculate the slack as a factor of the assigned room capacity and the students in the exam. As the SR value is maximized, we show the negative value shifted by 3 to have all objective values in $\mathbb{R}^+$. Almost all data points are below 2, which is the line of feasibility, as a value higher than 2 translates to a slack of below 1. In the case of actual student conflicts that would have been present in the scenario, the improvement is even more significant. To reduce the number of students with elective conflicts from around 150 to close to 0, the quality of the solutions $Q(T)$ drops again from 30,000 to 60,000. To further improve the robustness for either robustness measure, we can see a greater impact on the solution quality. However, decreasing the quality from 30,000 to 60,000, we can nearly completely eliminate the conflicts, which is a sig-

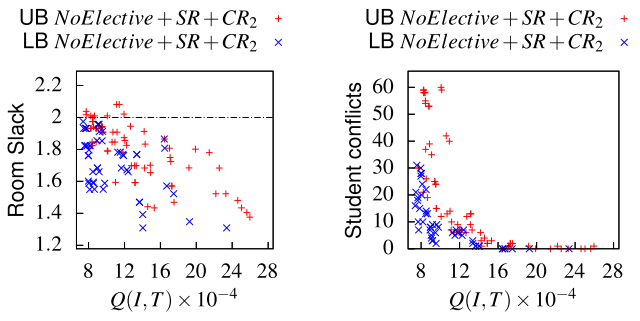


(a) *FAU-Summer*

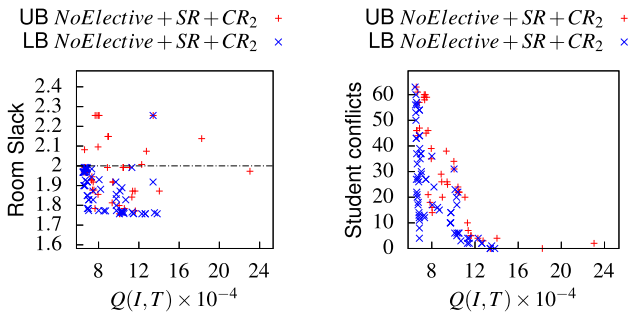


(b) *FAU-Winter*

Fig. 3 2D projections of the Pareto front using the soft constraints SR and CR_2 for the *FAU-Summer* and *FAU-Winter* instance. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to have only positive values



(a) *Random-Summer*



(b) *Random-Winter*

Fig. 4 2D projections of the Pareto front using the soft constraints SR and CR_2 for the *Random-Summer* and *Random-Winter* instances. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to have only positive values

nificant improvement. We can further observe that beyond a quality value of 60,000, we do not have improvements in the robustness.

For the *FAU-Winter* instance shown in Fig. 3b, the improvement is even more significant. In this instance, the solutions for the best quality values are not feasible. However, with a similar drop in solution quality as we discussed for the *FAU-Summer* instance, the solutions are reliably below the feasibility line of 2, and with a slightly greater drop in solution quality, the student conflicts are again close to 0.

As the randomly generated instances do not contain as many outliers, the drop in solution quality is significantly less for reducing the student robustness to near 0 and all solutions being feasible.

5.4 Comparison of constraint variants

In this section we compare the different constraint variants we introduced in this paper. In Tables 3 and 4 we show the arithmetic mean and standard deviation of the normalized hypervolume indicator over 10 runs on the *FAU-Summer* and *FAU-Winter* instances, respectively. The first two lines of Tables 3 and 4 show the results for the *FAU-Summer* and *FAU-Winter* instances with all nonzero conflicts as hard constraints, only excluding the elective conflicts that have no estimated students associated, while the last three lines are the results of the *FAU-Summer* and *FAU-Winter* instances with all elective conflicts excluded from the hard constraints. For the objective Q, all nonzero conflicts are considered in the robustness approaches. All entries use the robustness constraint SR.

Table 3 shows that the best results are obtained by using the conflict robustness measure CR_2. If we use the robustness measure CR_1, the hypervolume is increased to about the same value as the hypervolume for the $NonZero + SR$ constraint variant, i.e., the runs without using conflict robustness and only conflicts with a value of 0 excluded. However, when using this measure, the standard deviation is greater than with all other constraint variants, especially in the upper bound. When using the measure CR_2, we can observe a further increase in the hypervolume indicator. This holds true for both instance versions. Similar differences can be observed in Table 4. Tables 5 and 6 show the same information for the two randomly generated instances introduced before. Here, the differences are smaller; however, the previously discussed observations still hold.

The hypervolume indicator, while providing a value which helps in evaluating the different constraint variants, particularly when aggregated over multiple runs, is subject to discrimination. The first and most obvious discrimination

Table 3 Constraint variants for the *FAU-Summer* instance. The arithmetic mean ± the standard deviation is shown for the lower and upper bounds

Variant	Upper bound	Lower bound
NonZero+SR	0.34765977 ± 0.077687204	0.38247722 ± 0.047305077
NonZero+$SR + CR_2$	0.40336004 ± 0.052247487	0.516373 ± 0.048616074
NoElective+SR	0.12412788 ± 0.059266403	0.2173398 ± 0.05206661
NoElective+$SR + CR_1$	0.33652505 ± 0.07653496	0.49600616 ± 0.034985162
NoElective+$SR + CR_2$	0.43798947 ± 0.060741153	0.59320325 ± 0.03665869

Table 4 Constraint variants for the *FAU-Winter* instance. The arithmetic mean ± the standard deviation is shown for the lower and upper bounds

Variant	Upper bound	Lower bound
NonZero+SR	0.2848891 ± 0.051172428	0.3453682 ± 0.050359294
NonZero+$SR + CR_2$	0.3030739 ± 0.030513562	0.42566514 ± 0.026174081
NoElective+SR	0.15639836 ± 0.0427168	0.20648953 ± 0.037506036
NoElective+$SR + CR_1$	0.28924003 ± 0.11056109	0.3956934 ± 0.047037054
NoElective+$SR + CR_2$	0.34289578 ± 0.041596193	0.46169314 ± 0.030278686

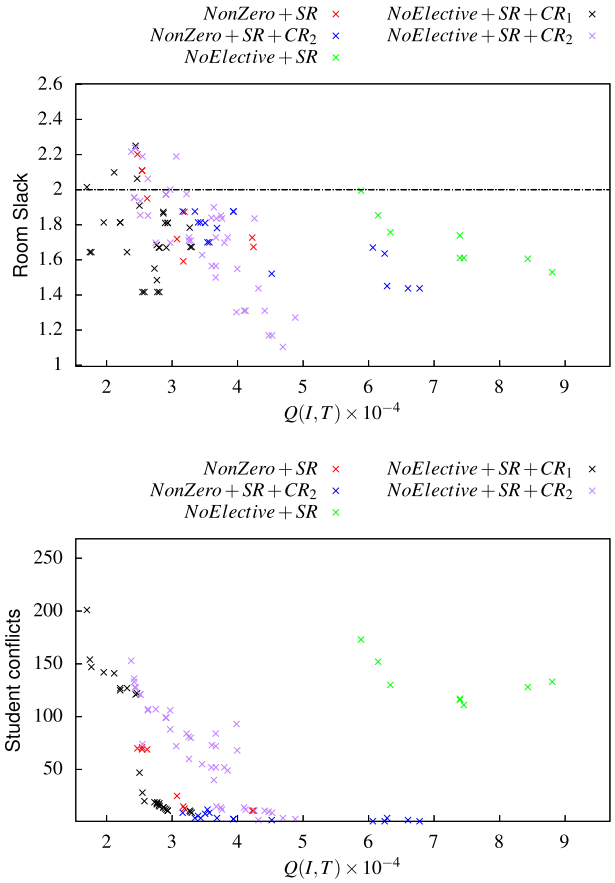


Fig. 5 2D projections of the Pareto fronts for the different constraint variants for the *FAU-Summer* instance. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to have only positive values

is that the indicator rates a large and diverse Pareto front higher than one with only a few but very good points. Furthermore, it also depends on the reference point and the scale of the different objective functions. The second problem can be reduced by the normalization of the data points, but it does not account for different weights of the objective functions.

Therefore, we must take a closer look at the Pareto fronts themselves. In Figs. 5 and 6 we show the 2D-projections of the results for the different constraint variants for a single run of the MOSA algorithm on the *FAU-Summer* and *FAU-Winter* instances, and in Figs. 7 and 6 for the same randomly generated instances, we used in previous parts of this section, respectively.

If we compare the results for the different constraint variants, we can see that, as with the hypervolume, the baseline constraint variant *NoElective* + SR has the worst Pareto fronts. This is especially prevalent in the run for the *FAU-Summer* instance, but can also be observed to varying degrees for the other instances. For instance *FAU-Summer*, a closer look at the number of elective conflicts shows that for the constraint variants *NonZero* + SR and *NonZero* + $SR + CR_2$, the values are quite small, with objective values between 20,000 and 40,000. We can see that when using CR_2 with this instance version, the number of elective student conflicts can be further decreased, with a slight drop in the quality Q. The results are similar for *FAU-Winter*, although on this run of this instance, the improvement is minor. Furthermore, for the student robustness on the *FAU-Summer* instance in Fig. 5, the results of the two constraint variants show that the room slack for *NonZero* + SR for most timetables is below 2, i.e., most timetables are feasible. For *NonZero* + $SR + CR_2$, the resulting timetable increased not only the conflict robustness, but also the student robustness, with all timetables being feasible. For the *FAU-Winter* instance shown in Fig. 6, the results are worse in regard to student robustness. On this instance

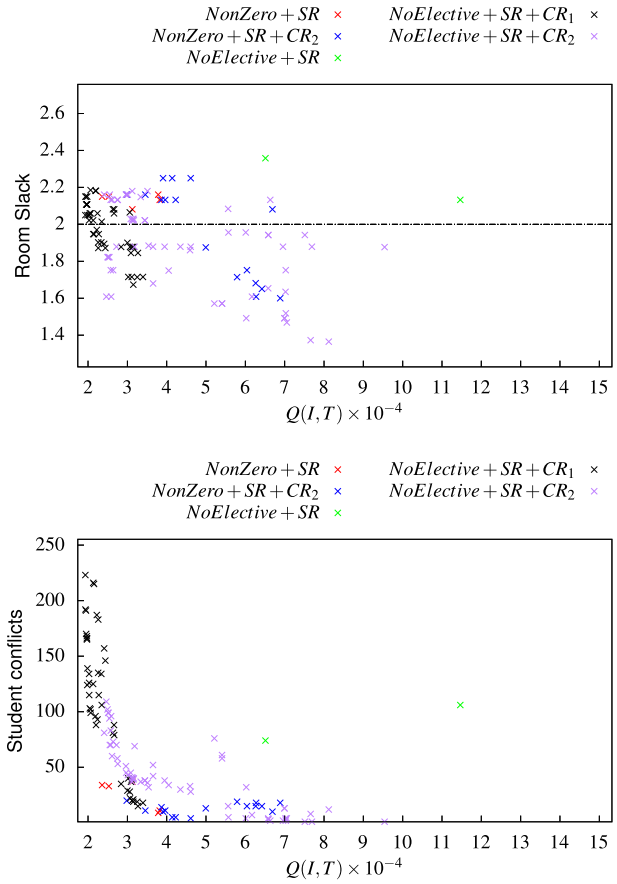


Fig. 6 2D projections of the Pareto fronts for the different constraint variants for the *FAU-Winter* instance. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to have only positive values

version, the solutions found are not feasible using constraint variant $NonZero + SR$. Using $NonZero + SR + CR_2$, the solutions with the best quality are also not feasible; however, with a drop in solution quality from 30,000 to 60,000, the solutions are feasible, with similar results for the student conflicts.

When we compare the values for the *FAU-Summer* instance version shown in Fig. 5 with all elective conflicts not enforced by hard constraint 3, we can see that the constraint variant $NoElective + SR + CR_1$ can reduce the elective student conflicts to about the same value as $NoElective + SR + CR_2$. The gradient of the Pareto front is even a bit better for CR_1 when compared with CR_2, and more solutions are found with good values in both measures. As shown in Fig. 6, the difference between CR_1 and CR_2 is a bit smaller, but still in the same order of magnitude for the *NoElective* version of the *FAU-Winter* instance. If we look at the student robustness in Fig. 6, we can see that while CR_1 has solutions that are better in the quality Q, these solutions are not feasible. If we restrict the comparison to the feasible solutions, both conflict robustness measures show the same performance, with CR_1 being slightly better. For the *FAU-Summer* instance in Fig. 5, both CR_1 and CR_2 found solutions with the best quality values being infeasible. Using CR_1 and CR_2, we find solutions that are feasible, with only a small drop in solution quality. And with a drop in solution quality that reduces the student conflicts to near 0, we obtain solutions with slack up to 1.6 for CR_1 and 1.8 for CR_2. However, as we have shown in Table 3, CR_1 is more susceptible to variance.

We can conclude that with the use of our robustness measure SR, most solutions are feasible, and with only a slight drop in solution quality we can increase the slack significantly. This also holds true when using this robustness measure with the two introduced conflict robustness measures, as they do not negatively influence each other, and both favor timetables with a more even spread in exams per time slot. For the number of elective student conflicts, we can conclude that when using robustness measure CR_1, we obtain slightly better values in solution quality while reducing the number of elective student conflicts to near 0. However, this is more susceptible to variance and does not account for bias. Using CR_2, with only a slightly worse solution quality, we can also reduce the number of elective student conflicts to near 0, and have the added benefit of a reduction in variance. In addition, as all possible conflicts have a value of at least 1, we can reduce bias in our solutions.

We further repeated the experiment for two randomly generated instances. As Tables 5 and 6 and the respective Figs. 7 and 8 show, the differences between the variants are smaller; however, over all experiments the previously discussed results can be observed. The smaller differences can be attributed to the real-world instances having higher and more frequent outliers than the randomly generated instances with the parameters used.

We can conclude that using CR_2 is therefore encouraged when a significant number of deviations in the conflicts between different years of the same term are expected; otherwise, the measure CR_1 is sufficient to reduce the number of elective student conflicts to close to 0.

6 Conclusion

We introduced a model for the examination timetabling problem that is defined on the curriculum information provided by the structure defined for the study programs available at a university. The resulting models are subject to uncertainty, given that the exact numbers of students for exams, and consequently the conflicts induced by the students between exams, are not known at the time of scheduling. We discussed the exact influence this uncertainty has on the model and discussed the benefits of using robustness measures in conjunction with estimation approaches, as data are limited for

improving estimations using standard techniques. We introduced three different robustness measures for the number of students of exams and the conflicts between exams, and discussed their use. We showed that the robustness measures can significantly improve the robustness of the solutions found by the optimization algorithm, with only a moderate impact on the solution quality. We can conclude that the use of CR_2 is therefore encouraged when a significant number of deviations in the conflicts are expected between different years of the same term; otherwise the robustness measure CR_1 is sufficient to reduce the number of elective student conflicts to near 0.

A Appendix

Tables 5 and 6 show the aggregated results that we discussed in Sect. 5.4 for the randomly generated instances *Random-Summer* and *Random-Winter* introduced in Sect. 2.4. Furthermore, the Pareto fronts for the different variants of a single run of the MOSA algorithm for each of the randomly generated instances can be found in Figs. 7 and 8.

Table 5 Constraint variants for the *Random-Summer* instance. The arithmetic mean ± the standard deviation is shown for the lower and upper bounds

Variant	Upper bound	Lower bound
NonZero+SR	0.35925484 ± 0.05955675	0.40935653 ± 0.03784311
NonZero+$SR + CR_2$	0.36059603 ± 0.026806103	0.4284907 ± 0.015303807
NoElective+SR	0.25598246 ± 0.07676259	0.33621103 ± 0.051204514
NoElective+$SR + CR_1$	0.37343073 ± 0.08591962	0.47737598 ± 0.02296595
NoElective+$SR + CR_2$	0.3752732 ± 0.044009466	0.444837 ± 0.01957122

Table 6 Constraint variants for the *Random-Winter* instance. The arithmetic mean ± the standard deviation is shown for the lower and upper bounds

Variant	Upper bound	Lower bound
NonZero+SR	0.22616526 ± 0.04205869	0.2570113 ± 0.044756006
NonZero+$SR + CR_2$	0.24433252 ± 0.027770452	0.29589984 ± 0.007044443
NoElective+SR	0.15014997 ± 0.04854938	0.18565114 ± 0.04065477
NoElective+$SR + CR_1$	0.24279578 ± 0.068723746	0.31989923 ± 0.0065063466
NoElective+$SR + CR_2$	0.2812462 ± 0.013288419	0.30888095 ± 0.0060866736

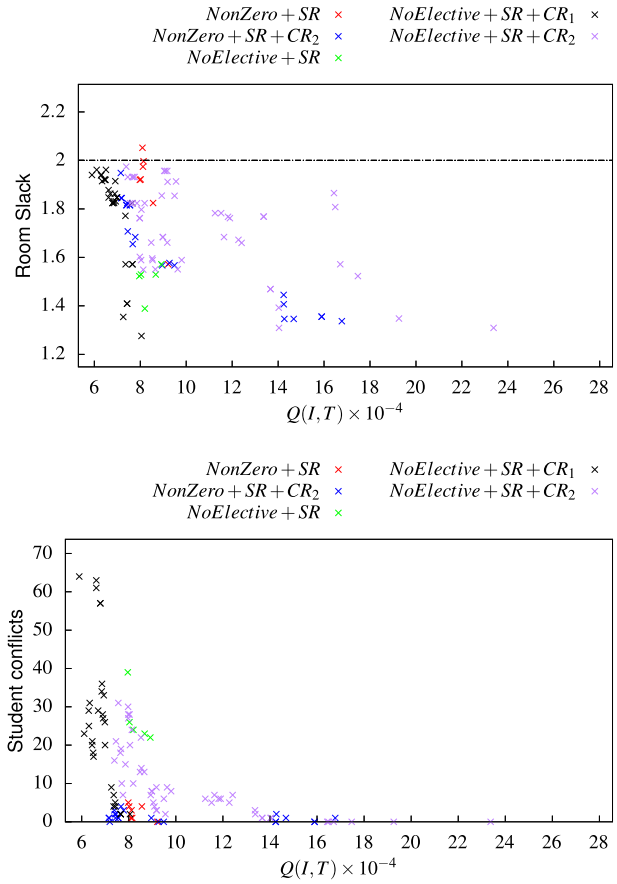


Fig. 7 2D projections of the Pareto fronts for the different constraint variants for the *Random-Summer* instance. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to only have positive values

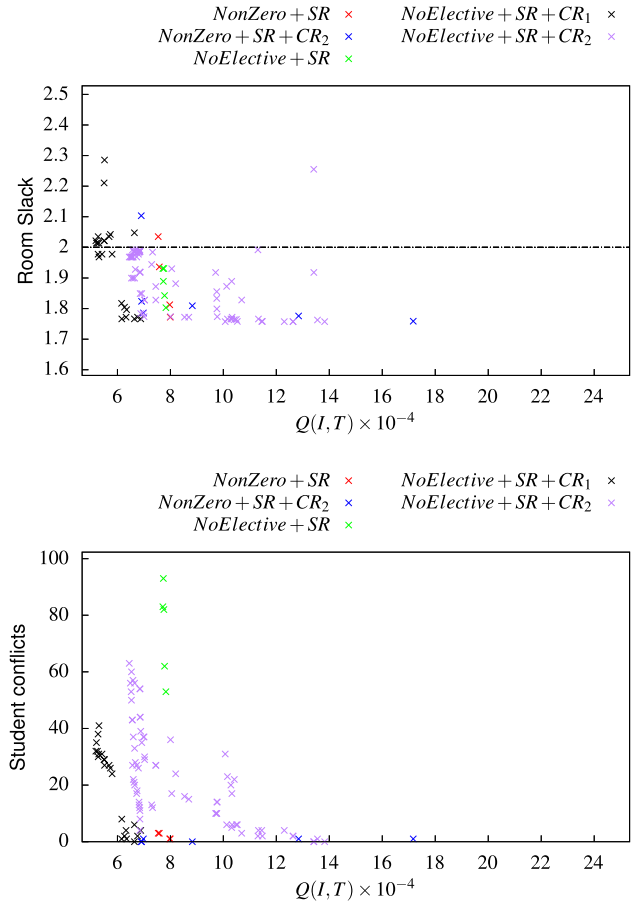


Fig. 8 2D projections of the Pareto fronts for the different constraint variants for the *Random-Winter* instance. Room slack is calculated analogously to the soft constraint SR multiplied by -1 and the negative values shifted by $+3$ to only have positive values

Acknowledgements We would like to thank Matthias Kergaßner for fruitful discussions and the anonymous referees for valuable remarks.

Funding Open Access funding enabled and organized by Projekt DEAL. Research was funded in part by the School of Engineering of the University of Erlangen-Nuremberg.

Data availability The data that support the findings of this study are available from the corresponding author upon request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

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

Amaral, P., & Pais, T. C. (2016). Compromise ratio with weighting functions in a tabu search multi-criteria approach to examination timetabling. *Computers & Operations Research, 72*, 160–174. https://doi.org/10.1016/j.cor.2016.02.012

Bassimir, B., & Wanka, R. (2018). Probabilistic curriculum-based examination timetabling. In: Proc 12th International conference on the practice and theory of automated timetabling (PATAT), 273–285.

Bassimir, B., & Wanka, R. (2019). Robustness approaches for the examination timetabling problem under data uncertainty. In: Proc. 9th multidisciplinary international conference on scheduling: theory and applications (MISTA), pp 381–395.

Bassimir, B., & Wanka, R. (2021). Conflicts in examination timetabling under uncertainty. In: Proceedings of the 13th international conference on the practice and theory of automated timetabling-PATAT.

Battistutta, M., Schaerf, A., & Urli, T. (2017). Feature-based tuning of single-stage simulated annealing for examination timetabling. *Annals of Operations Research, 252*, 239–254. https://doi.org/10.1007/s10479-015-2061-8

Battistutta, M., Ceschia, S., & De Cesco, F., et al. (2020). Local search and constraint programming for a real-world examination timetabling problem. In: Integration of constraint programming, artificial intelligence, and operations research (CPAIOR). Springer international publishing, pp 69–81, https://doi.org/10.1007/978-3-030-58942-4_5.

Ben-Tal, A., Ghaoui, L., & Nemirovski, A. (2009). Robust optimization. Princeton series in applied mathematics, princeton University Press, https://books.google.de/books?id=DttjR7IpjUEC.

Birge, J. R., & Louveaux, F. (2011). *Introduction to Stochastic Programming* (2nd ed.). Incorporated: Springer Publishing Company.

Carter, M. W., Laporte, G., & Lee, S. Y. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society, 47*, 373–383. https://doi.org/10.2307/3010580

Cataldo, A., Ferrer, J. C., Miranda, J., et al. (2017). An integer programming approach to curriculum-based examination timetabling. *Annals of Operations Research, 258*(2), 369–393. https://doi.org/10.1007/s10479-016-2321-2

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2022). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2022.07.011

Cicerone, S., D'Angelo, G., & Di Stefano, G., et al. (2009). Recoverable robustness in shunting and timetabling, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 28–60. https://doi.org/10.1007/978-3-642-05465-5_2.

Eley, M. (2007). Ant algorithms for the exam timetabling problem. In: Proceedings of the 6th International conference on practice and theory of automated timetabling VI. Springer-Verlag, Berlin, Heidelberg, PATAT'06, pp 364–382. https://doi.org/10.1007/978-3-540-77345-0_23.

Even, S., Itai, A., & Shamir, A. (1975). On the complexity of time table and multi-commodity flow problems. In: 16th annual symposium on foundations of computer science (sfcs 1975), pp 184–193,https://doi.org/10.1109/SFCS.1975.21

Fischetti, M., & Monaci, M. (2009). Light Robustness, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 61–84. https://doi.org/10.1007/978-3-642-05465-5_3.

Garey, M.R., & Johnson, D.S. (1979). Computers and intractability. A Guide to the Theory of NP-Completeness

Gładysz, B., & Kuchta, D. (2010). Multicriterial examination timetabling with uncertain information. *Multiple Criteria Decision Making, 09*(5), 97–112.

Häckler, A. (2019). Robuste Greedy-Approximationsverfahren für das Raumzuweisungsproblem. Bachelor thesis, University of Erlangen-Nuremberg.

Kempe, A. B. (1879). On the geographical problem of the four colours. *American Journal of Mathematics, 2*(3), 193–200.

Leite, N., Fernandes, C. M., Melício, F., et al. (2018). A cellular memetic algorithm for the examination timetabling problem. *Computers & Operations Research, 94*, 118–138. https://doi.org/10.1016/j.cor.2018.02.009

Liebchen, C., Lübbecke, M., & Möhring, R., et al. (2009). The Concept of recoverable robustness, linear programming recovery, and railway applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1–27. https://doi.org/10.1007/978-3-642-05465-5_1,

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., McMullan, P., & Burke, E.K., et al. (2007). The second international timetabling competition: Examination timetabling track. Tech. rep., Technical Report QUB/IEEE/Tech/ITC2007/-Exam/v4. 0/17, Queen's University.

McCollum, B., McMullan, P., Parkes, A. J., et al. (2012). A new model for automated examination timetabling. *Annals of Operations Research, 194*(1), 291–31. https://doi.org/10.1007/s10479-011-0997-x

Mühlenthaler, M., & Wanka, R. (2016). Fairness in academic course timetabling. *Annals of Operations Research, 239*, 171–188. https://doi.org/10.1007/s10479-014-1553-2

Muklason, A., Parkes, A. J., Özcan, E., et al. (2017). Fairness in examination timetabling: Student preferences and extended formulations. *Applied Soft Computing, 55*, 302–318. https://doi.org/10.1016/j.asoc.2017.01.026

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

Qu, R., Burke, E. K., McCollum, B., et al. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling, 12*(1), 55–8. https://doi.org/10.1007/s10951-008-0077-5

Schöbel, A., & Kratz, A. (2009). A Bicriteria approach for robust timetabling, Springer Berlin Heidelberg, Berlin, Heidelberg, 119–144. https://doi.org/10.1007/978-3-642-05465-5_5.

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

While, L., Hingston, P., Barone, L., et al. (2006). A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation, 10*(1), 29–38. https://doi.org/10.1109/TEVC.2005.851275

Yanez, J., & Ramirez, J. (2003). The robust coloring problem. *European Journal of Operational Research, 148*(3), 546–558. https://doi.org/10.1016/S0377-2217(02)00362-4

Zitzler, E., & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Parallel problem solving from nature—PPSN V: 5th international conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings 5, Springer, 292–301. https://doi.org/10.1007/BFb0056872.