Journal of Scheduling
https://doi.org/10.1007/s10951-024-00806-z

Solving the medical student scheduling problem using simulated annealing

Eugenia Zanazzo[1] · **Sara Ceschia**[1] · **Agostino Dovier**[2] · **Andrea Schaerf**[1]

Accepted: 18 January 2024

Abstract
We consider the medical student scheduling (MSS) problem, which consists of assigning medical students to internships of different disciplines in various hospitals during the academic year to fulfill their educational and clinical training. The MSS problem takes into account, among other constraints and objectives, precedences between disciplines, student preferences, waiting periods, and hospital changes. We developed a local search technique, based on a combination of two different neighborhood relations and guided by a simulated annealing procedure. Our search method has been able to find the optimal solution for all instances of the dataset proposed by Akbarzadeh and Maenhout (Comput Oper Res 129: 105209, 2021b), in a much shorter runtime than their technique. In addition, we propose a novel dataset in order to test our technique on a more challenging ground. For this new dataset, which is publicly available along with our source code for inspection and future comparisons, we report the experimental results and a sensitivity analysis.

Keywords Medical student scheduling · Local search · Simulated annealing

1 Introduction

The medical student scheduling (MSS) problem regards the assignment of medical students in subsequent periods to wards in designated hospitals for internships on the various disciplines, as the final task to get their degree. On the one hand, the students perfect their preparation learning from senior doctors; on the other hand, they are useful to the hospitals to provide actual assistance to patients.

The constraints of the problem regard the capacities of the wards, service levels, abilities, availabilities, precedences among disciplines, student preferences, waiting periods, and student rotations between hospitals. The typical horizon considered is one year, split into either 12 periods of one month or 24 periods of about two weeks.

The objective of the problem is to design a roster that simultaneously maximizes students' preferences and fairness between students, satisfying rules, regulations, and requirements of the medical school and the hosting hospitals.

Our research aim is to design and evaluate an efficient, effective, and robust solution technique for the MSS problem, able to address large instances. In addition, we aim to validate and revise formulations of the MSS problem already available in scientific literature, with the longer-term objective of defining a general and flexible version of the problem.

As a starting point, we consider the MSS formulation proposed by Akbarzadeh and Maenhout (2021b), which is a simplified version of the more general problem previously proposed by the same authors Akbarzadeh and Maenhout (2021a).

For this problem, we used a metaheuristic approach, and in particular we developed a local search technique based on a combination of two different neighborhood relations and guided by a simulated annealing procedure. The procedure has been tuned using rigorous statistical tests on a set of training instances, distinct from the validation ones.

✉ Andrea Schaerf
andrea.schaerf@uniud.it

Eugenia Zanazzo
eugenia.zanazzo@uniud.it

Sara Ceschia
sara.ceschia@uniud.it

Agostino Dovier
agostino.dovier@uniud.it

[1] Polytechnic Department of Engineering and Architecture, University of Udine, via delle Scienze 206, 33100 Udine, Italy

[2] Department of Mathematics, Computer Science and Physics, University of Udine, via delle Scienze 206, 33100 Udine, Italy

Published online: 06 April 2024

The outcome is that our solution method has been able to consistently find the optimal solution for all instances of the dataset proposed by Akbarzadeh and Maenhout (2021b), in a much shorter runtime on average than their exact technique.

In order to test our search method on harder benchmarks, we implemented a generator that we used to create more challenging instances including up to 320 students. Such number of students represents a realistic size and is much larger than those in the original dataset (max 80 students). In addition, as explained in Sect. 4, the generated instances also activate various constraints not used in the original dataset. In fact, some constraints that are present in the mathematical model are never actually binding in the original dataset, given that they are always satisfied. In particular, the constraint on the minimum number of students in a ward, which makes the instances particularly difficult to solve, is redundant as the corresponding value is always equal to zero.

In addition, we revised the MIP model proposed by Akbarzadeh and Maenhout (2021b) and we implemented it using CPLEX. The CPLEX solver, suitably warm-started with feasible solutions coming from our local search method, has been run for comparison and for obtaining lower bounds.

Finally, we propose a sensitivity analysis on the weight of the fairness component, which highlights the need for more advanced ways to define and compute fairness.

All instances and solutions are publicly available at https://bitbucket.org/satt/mss_data. The source code is also available in the same repository for future comparisons and possible extensions by other researchers.

The remainder of the paper is organized as follows: In Sect. 2, we review the related literature; in Sect. 3, we describe the problem through a mathematical formulation; in Sect. 4, we introduce the available instances; in Sect. 5, we describe the solution techniques that we implemented; Sect. 6 reports the computational results, and lastly, Sect. 7 provides some concluding remarks and possible future directions.

2 Related work

The problem of scheduling the training program for medical residents was firstly introduced by Franz and Miller (1993) as a special case of the multiperiod staff assignment problem. In particular, they studied the situation of family practice physicians whose training program is particularly complex because it must ensure sufficient experience in a variety of disciplines, given that they are non-specialists. The objective of the schedule was to maximize the student preferences related to disciplines and period unavailabilities, while meeting the hospital coverage requirements and some specific rules about the design of the program. Franz and Miller affirm that the main issue in practice is to obtain feasible solutions and propose a rounding heuristic applied to the solution of the corresponding linear programming problem, which runs quickly so that it can be used iteratively and in an interactive way by the decision maker.

Beliën and Demeulemeester (2007) addressed the problem of constructing the trainee scheduling for medical students who want to specialize in a specific discipline using decomposition and column generation techniques. In their formulation, the training program is composed of activities that have to be assigned to trainees to guarantee the staff coverage, ensuring that each trainee performs each activity between a minimum and a maximum number of periods, and respecting students' unavailability constraints. As a consequence, in their model there is no possibility to customize the individual training schedule.

A similar problem, where each student must follow the same set of disciplines (but in different periods), was proposed by Goodman et al. (2012). In their case, the objective function penalizes multiple assignments to the same ward/hospital in order to get experience in a variety of clinical environments. As a solution technique, they implemented Greedy Randomized Adaptive Search Procedures (GRASP) (Resende & Ribeiro, 2016) using several construction heuristics and a random descent local search for the improvement phase.

Guo et al. (2014) defined the resident scheduling problem for a one-year training program, which includes some general constraints on staff coverage, teaching demands and educational requirements. Comparative results are reported for a greedy assignment heuristic and for an IP model implementation on a set of artificial instances. In addition, the authors extended the mathematical model to simultaneously determine the training schedule and the daily working schedule for residents.

The notion of rotation, i.e. a period of training to spend in a ward, was employed in different problem formulations. Zheng et al. (2016) studied the trainee scheduling problem in the context of nursing schools; it differs from the medical student problem since all trainees must follow the same training program composed of different rotations that must be performed consecutively. The problem is solved by a two-phase heuristic algorithm that first solves a simplified version of the problem by a random search procedure and then applies specific neighborhood operators to further explore the solution space.

Proano and Agarwal (2018) studied the problem of scheduling resident rotations for a three-year internal medici ne residency program: students rotate in different units and have to complete a minimum number of rotations in each unit during the training program. The duration of a rotation can vary across different units. For this problem, the authors formulated a multi-objective mathematical model that is subsequently solved with a goal programming framework.

Similarly, Cire et al. (2019) tackled the rotation assignment and scheduling problem arising typically in North American medical schools. They formulate a network-flow-based model where the objective function minimizes the sum of fixed and variable costs associated with the assignment of students to hospitals and rotations.

Several other works tackled the problem at the operational level and studied the rostering problem for medical residents, i.e. assigning working shifts for a planning horizon taking into account staff requirements, on-call duties, working rules and preferences (White & White, 2003; Topaloglu, 2006; Cohn et al., 2009; Topaloglu & Ozkarahan, 2011; Güler et al., 2013; Bard et al., 2016; Smalley & Keskinocak, 2016; Lemay et al., 2017; Brech et al., 2019).

On the contrary, Kraul et al. (2019) studied the strategic problem of defining how many residents of a given specialty a hospital can timely train taking into account uncertainty about the number of medical procedures. They developed a column generation heuristic which is applied to a real-world case study of a German university hospital. Subsequently, the same problem is tackled at the tactical level by generating the annual schedule for residents using a genetic algorithm (Kraul, 2020).

Recently, Seizinger and Brunner (2023) investigated the problem of planning the apprenticeship program of professional nurses in dual vocational systems, that arises in German-speaking countries. The peculiarity of this system is that it combines theoretical education and practical training: the first one can take place in professional schools, while the practical subjects at any employer (e.g. a hospital). The authors propose a two-stage approach: in the first stage blocks of practical and theoretical education are defined and scheduled for each class (school schedule); in the second stage, individual students are assigned to hospital units during their work block (unit assignment). Two corresponding IP models have been developed that are used in a hierarchical approach. In addition, a *matheuristic* that decomposes the unit assignment model by students is presented and tested on real-world data.

The MSS problem was introduced by Akbarzadeh and Maenhout (2021a) as the tactical problem of assigning medical students to disciplines over one year considering medical schools' requirements, hospitals' availabilities and students' preferences. The authors propose a rich formulation inspired by the real-life practices at Ghent University, taking into account many complex constraints, such as different training programs consisting of mandatory and elective disciplines, precedence relationship between disciplines, students' past studies, reserve and emergency capacities of wards, possibility of attending a discipline abroad, and utilization of accommodations. The objective function is the weighted sum of six components: maximize total student preferences and worst student desire score, and minimize the use of reserve and emergency capacity, the number of students below the minimum demand threshold, and the under-utilization of the accommodations. For this problem, Akbarzadeh and Maenhout (2021a) developed a solution method that uses a constructive heuristic to obtain an initial solution and then applies local search operators. It was tested on two datasets of generated instances: one of small size instances (20/40 students and 12/24 time periods) and the other of real-life dimension instances (320 students and 24 time periods). The problem formulation proposed in Akbarzadeh and Maenhout (2021b) and presented in Sect. 3 simplifies the one previously presented by the same authors, preserving the essential and more common features of the medical student scheduling problem. They proposed an exact branch and price algorithm that is able to obtain optimal solutions for instances up to 80 students and 12 periods within less than 500 s of running time (see Table 7 for detailed results). Lastly, the MSS was further investigated in Akbarzadeh et al. (2022) where the heuristic approach developed in Akbarzadeh and Maenhout (2021a) was improved, adapted to the specific real-life situation of the Faculty of Medicine and Health Sciences at Ghent University, and integrated in the information system of the faculty as the resident scheduling module.

3 Problem definition

In this section, we introduce the mathematical model of the problem, which is the one presented in the work by Akbarzadeh and Maenhout (2021b, Section 3.2), with some minor corrections (the detailed list is reported in the Appendix).

In the MSS problem, the curriculum manager creates the individual training schedule of each medical student by assigning disciplines, wards and hospitals over the scheduling horizon, considering the student's characteristics and preferences, the hospitals' availabilities and the training program. Table 1 shows the notation employed in the multi-objective mixed integer programming model (MIP) proposed for the problem.

The main decision variables for the MIP model are: the assignment variable v_{sdth}, which is equal to 1 if student s starts discipline d at period t in hospital h, 0 otherwise; the network-flow variable $y_{s\bar{d}dh}$, which is equal to 1 if student s attends discipline d directly after discipline $\bar{d}$ in hospital h, 0 otherwise. The latter is used as in a multi-commodity flow model where nodes represent disciplines, that are connected if they are attended one directly after the other. As a consequence, it is necessary to define a dummy source node where all flows originate. The dummy start node corresponds to the dummy discipline 0 that cannot be preceded by any other discipline. Obviously, the duration of the dummy discipline

Table 1 Notation of the input data

Symbol	Description
$\mathcal{S}$	set of students
$\mathcal{D}$	set of disciplines
$\mathcal{D}^0$	set of disciplines without the dummy discipline ($\mathcal{D}^0 = \mathcal{D} \setminus \{0\}$)
$\mathcal{G}$	set of groups
$\mathcal{T}$	set of time periods
$\mathcal{W}$	set of wards
$\mathcal{H}$	set of hospitals
k_{sg}	total number of disciplines student s must attend from group g
φ_{sdg}	1 if student s has discipline d in group g; 0 otherwise
Du_d	duration of discipline d
$Sk_{\bar{d}d}$	if discipline $\bar{d}$ must be completed before discipline d
λ	maximum number of disciplines for a student in a hospital
ξ_{dwh}	1, if discipline d can be assigned to ward w in hospital h
$\mathrm{Dem}^{\mathrm{Min}}_{wht}$	minimum student requirements for ward w in hospital h in period t
$\mathrm{Dem}^{\mathrm{Max}}_{wht}$	maximum student requirements for ward w in hospital h in period t
Av_{st}	1, if student s is available in period t; 0 otherwise
Ab_{sdh}	1, if student s is qualified for discipline d in hospital h; 0 otherwise
$\mathrm{Pr}^{\mathrm{disc}}_{sd}$	preference of student s about discipline d
$\mathrm{Pr}^{\mathrm{hosp}}_{sh}$	preference of student s about hospital h
$\mathrm{Pr}^{\mathrm{trPr}}_{d}$	preference of the curriculum manager about discipline d
α	weight of student desires objective
β	weight of fairness between students objective
M	big number

is zero. All decision variables are listed in Table 2, including the auxiliary ones used to express the objectives.

The objective function (1) is a weighted sum of two components to maximize: the total student desire *score* and the *fairness* between students. In the first objective, the students' desires relate to discipline preferences, hospital preferences, discipline preferences of the curriculum manager (if students have not expressed an explicit preference), hospital changes, and possible waiting time between two consecutive disciplines. The fairness objective is pursued by a max–min approach, i.e., by maximizing the worst student schedule, that is the one with the minimum desire score $\mathrm{Des}^{\min}$.

$$\max z = \alpha \sum_{s \in \mathcal{S}} \mathrm{Des}_s + \beta \mathrm{Des}^{\min} \tag{1}$$

The problem involves three main stakeholders that have different and sometimes conflicting requirements, i.e. the medical school, local hospitals, and medical students. The medical school is responsible for the training program, which concerns different disciplines divided into groups. For each of these groups, each student must carry out a minimum number of disciplines k_{sg}.

$$\sum_{d \in \mathcal{D}} \varphi_{sdg} \sum_{t \in \mathcal{T}} \sum_{h \in \mathcal{H}} v_{sdth} \geq k_{sg} \qquad \forall s \in \mathcal{S}, \forall g \in \mathcal{G} \tag{2}$$

For each student, the total number of assigned disciplines is equal to the sum of the number of disciplines to be selected from each group.

$$\sum_{d \in \mathcal{D}^0} \sum_{t \in \mathcal{T}} \sum_{h \in \mathcal{H}} v_{sdth} = \sum_{g \in \mathcal{G}} k_{sg} \qquad \forall s \in \mathcal{S} \tag{3}$$

Constraints (4–7) regulate the flow in the network. In particular, each discipline can be directly preceded by another discipline at most.

$$\sum_{\bar{d} \in \mathcal{D}} \sum_{h \in \mathcal{H}} y_{s\bar{d}dh} \leq 1 \qquad \forall s \in \mathcal{S}, \forall d \in \mathcal{D}^0 \tag{4}$$

For each student, the flow starts from a dummy source node which corresponds to the dummy discipline 0.

$$\sum_{d \in \mathcal{D}^0} \sum_{h \in \mathcal{H}} y_{s0dh} = 1 \qquad \forall s \in \mathcal{S} \tag{5}$$

A discipline cannot be followed by the dummy one or by itself.

$$\sum_{s \in \mathcal{S}} \sum_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}} (y_{sddh} + y_{sd0h}) = 0 \tag{6}$$

The flow conservation constraints ensure that for each selected discipline d, a preceding path $\bar{\bar{d}} \rightarrow \bar{d} \rightarrow d$ must exist.

$$\sum_{d \in \mathcal{D}^0} \sum_{h \in \mathcal{H}} y_{s\bar{d}dh} \leq \sum_{\bar{\bar{d}} \in \mathcal{D}} \sum_{h \in \mathcal{H}} y_{s\bar{\bar{d}}\bar{d}h} \qquad \forall s \in \mathcal{S}, \forall \bar{d} \in \mathcal{D}^0 \tag{7}$$

The flow variable is active when the corresponding assignment variable is selected.

$$\sum_{t \in \mathcal{T}} v_{sdth} = \sum_{\bar{d} \in \mathcal{D}} y_{s\bar{d}dh} \qquad \forall s \in \mathcal{S}, \forall d \in \mathcal{D}^0, \forall h \in \mathcal{H} \tag{8}$$

Precedence constraints between pairs of disciplines $(\bar{d}, d)$ are imposed using the Big M method, where M can be fixed as the total number of time periods $|\mathcal{T}|$. In detail, if the

Table 2 Decision variables

Symbol	Domain	Description
v_{sdth}	$\in \{0, 1\}$	1, if student s starts discipline d at period t in hospital h; 0 otherwise
$y_{s\bar{d}dh}$	$\in \{0, 1\}$	1, if student s attends discipline d directly after discipline $\bar{d}$ in hospital h; 0 otherwise
Ch_{sd}	$\in \{0, 1\}$	1, if student s changes hospital for discipline d; 0 otherwise
W_{sd}	≥ 0	number of periods student s has to wait idle before starting discipline d
Des_s	free	aggregated desire score for student s
$\text{Des}^{\min}$	free	worst aggregated desire score across all students

parameter $\text{Sk}_{\bar{d}d}$ is equal to one, discipline $\bar{d}$ is a mandatory propaedeutic discipline for d.

$$\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} t\cdot v_{\text{sdth}} + M\left(1-\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} v_{\text{sdth}}\right) \geq Sk_{\bar{d}d}\left(\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} v_{s\bar{d}th}\right) + Sk_{\bar{d}d}\cdot M\left(1-\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} v_{s\bar{d}th}\right) \quad \forall s\in\mathcal{S}, \forall d\in\mathcal{D}^0, \forall \bar{d}\in\mathcal{D}^0 \tag{9}$$

There is a maximum limit Λ on the total number of disciplines that can be carried out in the same hospital.

$$\sum_{d\in\mathcal{D}\setminus 0}\sum_{\bar{d}\in\mathcal{D}} y_{s\bar{d}dh} \leq \Lambda \qquad \forall s\in\mathcal{S}, \forall h\in\mathcal{H} \tag{10}$$

Local hospitals define requirements in terms of minimum (Constraint 11) and maximum (Constraint 12) student staff necessary in each ward for each period, and the characteristics and competencies that students must fulfill (Constraint 13).

$$\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}^0}\sum_{\bar{t}=\max(0,t-\text{Du}_d+1)}^{t} \xi_{\text{dwh}}\cdot v_{sd\bar{t}h} \geq \text{Dem}^{\text{Max}}_{\text{wht}} \quad \forall t\in\mathcal{T}, \forall w\in\mathcal{W}, \forall h\in\mathcal{H} \tag{11}$$

$$\sum_{s\in\mathcal{S}}\sum_{d\in\mathcal{D}^0}\sum_{\bar{t}=\max(0,t-\text{Du}_d+1)}^{t} \xi_{dwh}\cdot v_{sd\bar{t}h} \leq \text{Dem}^{\text{Max}}_{\text{wht}} \quad \forall t\in\mathcal{T}, \forall w\in\mathcal{W}, \forall h\in\mathcal{H} \tag{12}$$

$$\sum_{t\in\mathcal{T}} v_{\text{sdth}} \leq \text{Ab}_{\text{sdh}} \qquad \forall s\in\mathcal{S}, \forall d\in\mathcal{D}^0, \forall h\in\mathcal{H} \tag{13}$$

The matrix Av stores the availability of students during the scheduling horizon, such that a student can be assigned to any discipline in period t only if $\text{Av}_{\text{st}} = 1$.

$$\sum_{h\in\mathcal{H}} \text{Du}_d\cdot v_{\text{sdth}} \leq \sum_{\bar{t}=t}^{t+\text{Du}_d-1} \text{Av}_{s\bar{t}} \quad \forall s\in\mathcal{S}, \forall t\in\mathcal{T}, \forall d\in\mathcal{D}^0 \tag{14}$$

Constraints (15–17) are used to evaluate the different components of the objective function. In Constraint (15), the variable Ch_{sd} representing the number of hospital changes is constrained to be larger than or equal to hospital changes induced by the flow variables for consecutive disciplines.

$$Ch_{sd} \geq y_{s\bar{d}dh} - \sum_{\bar{\bar{d}}\in\mathcal{D}} y_{s\bar{\bar{d}}\bar{d}h} \qquad \forall s\in\mathcal{S}, \forall d,\bar{d}\in\mathcal{D}^0, \forall h\in\mathcal{H} \tag{15}$$

The time W_{sd} that a student has to wait before starting a discipline is set through Constraints (16-17). We recall that we assume that the duration Du_d of the dummy discipline is zero.

$$\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} t\cdot v_{sdth} \geq W_{sd} + \sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} t\cdot v_{s\bar{d}th} + \text{Du}_{\bar{d}} - M\left(1-\sum_{h\in\mathcal{H}} y_{s\bar{d}dh}\right) \quad \forall s\in\mathcal{S}, d\in\mathcal{D}^0, \bar{d}\in\mathcal{D} \tag{16}$$

$$\sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} t\cdot v_{sdth} \leq W_{sd} + \sum_{t\in\mathcal{T}}\sum_{h\in\mathcal{H}} t\cdot v_{s\bar{d}th} + \text{Du}_{\bar{d}} + M\left(1-\sum_{h\in\mathcal{H}} y_{s\bar{d}dh}\right) \quad \forall s\in\mathcal{S}, d\in\mathcal{D}^0, \bar{d}\in\mathcal{D} \tag{17}$$

The aggregated desire score of a student Des_s takes into account five aspects. The first three are positive ones, i.e., the student preference related to a specific discipline $\text{Pr}^{\text{disc}}_{\text{sd}}$, the student preference related to a specific hospital $\text{Pr}^{\text{hosp}}_{\text{sh}}$, and a general preference of all students toward a specific discipline $\text{Pr}^{\text{trPr}}_{d}$. On the contrary, the last two components are undesired, and therefore, they are negative in order to be minimized: the waiting time between two consecutive discipline assignments

W_{sd}, and the total number of hospital changes for a student Ch_{sd}. Notice that all the weights ω_s^* (except ω^{trPr}) are set by each student in order to represent their individual preferences.

$$\begin{aligned} Des_s \leq & \sum_{\bar{d}\in\mathcal{D}}\sum_{d\in\mathcal{D}}\sum_{h\in\mathcal{H}}\left(\omega_s^{disc}\cdot Pr_{sd}^{disc}\cdot y_{s\bar{d}dh}\right) \\ & + \sum_{\bar{d}\in\mathcal{D}}\sum_{d\in\mathcal{D}}\sum_{h\in\mathcal{H}}\left(\omega_s^{hosp}\cdot Pr_{sh}^{hosp}\cdot y_{s\bar{d}dh}\right) \\ & + \sum_{\bar{d}\in\mathcal{D}}\sum_{d\in\mathcal{D}}\sum_{h\in\mathcal{H}}\left(\omega^{trPr}\cdot Pr_{d}^{trPr}\cdot y_{s\bar{d}dh}\right) \\ & - \sum_{d\in\mathcal{D}}\left(\omega_s^{wait}\cdot W_{sd}\right) - \sum_{d\in\mathcal{D}}\left(\omega_s^{chng}\cdot Ch_{sd}\right) \\ & \forall s\in\mathcal{S} \end{aligned} \tag{18}$$

The fairness objective is obtained by maximizing the worst desire score Des^{min} acquired by a student.

$$Des^{min} \leq Des_s \qquad \forall s\in\mathcal{S} \tag{19}$$

In addition, we introduced two constraints that were missing in the original model by Akbarzadeh and Maenhout (2021b). In the first one, we force the dummy discipline to start at time period 0 to take into account also the possible waiting time before the first discipline assignment.

$$\sum_{h\in\mathcal{H}} v_{s00h} = 1 \quad \forall s\in\mathcal{S} \tag{20}$$

Finally, the dummy discipline is assigned to only one period and hospital for each student.

$$\sum_{h\in\mathcal{H}}\sum_{t\in\mathcal{T}} v_{s0th} = 1 \quad \forall s\in\mathcal{S} \tag{21}$$

The domains of the decision variables are reported in Table 2.

Notice that variables Des_s and Des^{min} are unconstrained as the aggregated desire can assume also negative values if the disadvantageous measures related to the hospital changes and waiting times are dominant.

4 Instances

We consider two sets of instances, that we name Dataset 1 and Dataset 2. The first is composed of one hundred synthetic instances generated by Akbarzadeh and Maenhout (2021b) to test their solution method (Sect. 4.1). The second one is created by our new generator, which activates more constraints and exhibits more variability in terms of size and feature values (Sect. 4.2).

4.1 Dataset 1

The instances of the dataset by Akbarzadeh and Maenhout (2021b) have a number of students equal to either 40 or 80 and a number of disciplines of either 12 or 24. This dataset refers to a version of the problem simpler than the one presented in Sect. 3, as it includes the following simplifications:

- *A* there are no requirements on the minimum number of students per ward/period, i.e., all values Dem_{wht}^{Min} are set to 0;
- *B* each student can follow an arbitrary number of disciplines in any of the hospitals, i.e., Λ is set to an arbitrarily high value (e.g., to the number of disciplines $|\mathcal{D}|$);
- *C* every student has the ability to work in any ward, i.e., $Ab_{sdh} = 1$ for all students for all disciplines and hospitals;
- *D* the preferences of the manager are fixed, i.e., $Pr_d^{trPr} = 1$ for all disciplines;
- *E* groups of disciplines are fixed, not depending on the specific student; in addition there are no *mutual* disciplines, i.e., every discipline belongs to at most one group;
- *F* the duration of the disciplines is equal for all disciplines in each instance, and the fixed duration in the various instances is equal to 1, 2, or 4 periods;
- *G* each discipline can be assigned to exactly one ward in each hospital, i.e., for each hospital h the matrix $\xi_{\mathbf{h}}$ is the identity matrix.

In addition, in all instances the weights of the objectives are fixed to $\alpha = 1$ and $\beta = 1$.

4.2 Dataset 2

To create the new dataset we developed a parameterized generator that takes as input the number of students, the number of disciplines, and the duration of the disciplines, and delivers a random instance.

We now briefly explain the design choices we made in the development of the generator. In particular, we discuss which of the assumptions *A*–*G* we keep and which we remove.

First, we assume that the duration is equal for all disciplines in an instance and that the length of the horizon is equal to the number of disciplines times the duration, so that a student, if needed, has the possibility to follow all the disciplines available.

As was previously done by Akbarzadeh and Maenhout (2021b), we assume that each hospital has a number of wards equal to the number of disciplines and that each ward covers, exclusively, only one discipline. This implies that, notwithstanding other constraints, a student can attend a discipline in any of the hospitals available.

The number of hospitals is randomly chosen in the interval [2, 5], while the maximum number of disciplines that a student can follow in a single hospital is a random number in the range $[\lfloor \frac{|\mathcal{D}|}{|\mathcal{H}|+1} \rfloor, |\mathcal{D}|]$.

The number of disciplines that each student has to follow in a group g is randomly chosen in the interval $[0, |D_g|]$ where D_g is the set of disciplines belonging to group g; each student is guaranteed to follow at least one discipline (regardless of the group). In this definition of the problem, it is assumed that the partition of disciplines into groups is the same for each student.

The precedences between the disciplines are generated as follows. Given an enumeration, ordered by group, of the disciplines available, a discipline i, belonging to group g, will be required to be preceded by a number of disciplines k in the range $[0, i - j]$ where j is the index of the first discipline in group g. Then, the k indexes, representing the disciplines that must be completed before starting discipline i, will be chosen in the range $[j, i - 1]$. In principle, there is no limit to the depth of the direct acyclic graph representing the precedences.

Moreover, a student can be unavailable only for at most a number of periods that still allows them to complete all the disciplines required. The probability that a student does not possess the ability to work at a certain ward is set at 5%; however, by design, each student has to be able to attend each of the disciplines in at least one hospital.

To generate the minimum number of students requested by the hospitals, at first a number in $[1, |\mathcal{H}|]$ is randomly chosen, and it represents the number of hospitals that will require, at some point, a minimum number of students. Then, the number of students required for each pair (ward, period) is, in principle, a random number in the range $[0, k]$ where k is a raw estimate of the mean number of slots that can be filled without exceeding the number of slots that the students can cover according to their internship plan. However, if a ward covers a discipline that needs to be preceded by one or more disciplines, then no minimum requirement can be set until a number of periods sufficient to complete the preceding discipline(s) have passed.

Similarly, the maximum number of students allowed in a ward in a given period is a random number in the range $[k/2, k + 1]$ where k is a raw estimate of the mean number of students that have to be allowed so that each student can fulfill his/her internship requirements within the end of the horizon.

For the generation of both hospital and discipline preferences, first a random number representing the weight of the student preferences is drawn in the range [1, 3] then for each hospital and each discipline the preference expressed by a student is simulated by choosing a random number in the range [1, 5]. On the other hand, to generate the preferences that a student expresses relating to the presence of waiting times and hospital changes in the schedule a random number in the range [1, 3] is selected. The manager's preferences are produced by randomly choosing a number in the range [0, 5] for each of the disciplines involved in the internship, while the weight of the manager's preferences is assumed to be 1.

Table 3 Generated instances cardinality

Parameters	Cardinality
Students ($\|\mathcal{S}\|$)	40, 80, 160, 240, 320
Disciplines ($\|\mathcal{D}\|$)	12, 24
Duration (Du_d)	1, 2, 4

According to the design choices described above, we removed the simplifications introduced in Sect. 4.1, except for the last three (points *E*, *F*, and *G*). The reason is that these latter limitations are the most realistic ones and they do not prejudice the complexity of the problem. The most important limitation that we removed is the one on the minimum students per ward (point *A*), which makes the instances more constrained, and consequently the feasibility harder to be obtained.

Using the generator we produced Dataset 2 composed of 30 instances, one for each combination of the parameters in Table 3. In addition, 5 instances for each combination (150 in total) have been generated for the tuning procedure.

While the design choices of the generator aim to produce realistic and satisfiable instances, the satisfiability of a generated instance is not guaranteed. In order to have only feasible instances, for each instance (including tuning ones) the generation process is repeated until it shows to be satisfiable.

4.3 Instance features

For both Datasets 1 and 2, several features were computed. The first type of features are the scalars corresponding to the number of students $|\mathcal{S}|$, the number of disciplines $|\mathcal{D}|$, the (fixed) duration of each discipline Du_d, the length of the horizon $|\mathcal{T}|$, and the number of hospitals $|\mathcal{H}|$.

The second type of features are real-valued indicators computed starting from the input data. With the term *packing* (P), we indicate the maximum number of disciplines to be followed in a single hospital divided by the number of disciplines. On the other hand, the feature *stiffness* (St) is computed by dividing the sum of the minimum number of students required in any of the hospitals throughout the entire horizon by the number of slots to be filled. With *busyness* (B) we refer to the number of slots to be filled in the schedule divided by the total number of available slots in the schedule.

The last two features are related to the precedences' direct acyclic graph (DAG), calculated as the length of the longest

path between two disciplines. Specifically, Ds and Dp show, respectively, the *density* and *depth* of the DAG.

Tables 4 and 5 recap the features for Datasets 1 and 2, respectively. For Dataset 1, instances that are homogeneous in terms of the scalar values are grouped. For feature B, which changes among instances, the table reports min and max values. For Dataset 2, each row of the table represents a single instance.

First, we see that Dataset 2 includes instances that are much bigger in terms of number of students and periods. We also see that for Dataset 1, P is always equal to 1, meaning that there are no limits on the number of assignments of a student to the same hospital, whereas for Dataset 2 we have diverse values for P ranging from 0.292 to 1. Similarly, St is always 0 for Dataset 1, given that there are no minimum requirements, while it has a wide range (from 0.022 to 0.81) for Dataset 2. Finally, we can observe that the values of B found in Dataset 1 are higher than those recorded in Dataset 2. Moreover, the values recorded on Dataset 1 are on many occasions close to 1, meaning that the number of slots to be filled is very close to the number of slots available in the schedule. As a consequence, the students' schedule is very dense and the waiting component in the objective function has very little impact on the solution of those instances because it is essentially a constant. In contrast, the values found in Dataset 2 allow for a higher degree of freedom in building the schedule, while still ensuring that the number of disciplines followed by a student is high enough.

We designed a new format (extension `.dzn`) to store the instances, based on the data file format of MiniZinc (Stuckey et al., 2022). This choice is motivated by the will of making the instances easily readable by any class of users as well as highly structured and standardized to facilitate their parsing with respect to the original text-only format.

5 Solution method

In this section, we introduce the key features of local search in stages. In Sect. 5.1, we discuss the search space and the initial solution generation. In Sect. 5.2, we describe our neighborhood relation. In Sects. 5.3 and 5.4, we present the simulated annealing procedure and its integration with the MIP model, respectively.

5.1 Search space and initial solution

Our local search technique uses, for the search space, an integer-valued matrix of size $|\mathcal{S}| \times |\mathcal{T}|$ that assigns to each student $s \in \mathcal{S}$ in each period $t \in \mathcal{T}$ an index that represents a specific ward of a specific hospital. That is, we use a single value that encodes a pair ⟨hospital, ward⟩.

According to the simplification G discussed in Sect. 4, only one discipline can be undertaken in any specific ward, so the elements of the matrix also specify the discipline. The conventional value -1 is used when the student is not assigned to any internship in a specific period.

Given that finding a feasible solution is difficult in some instances, we decided to also include in our search space solutions that may violate the three hard constraints that turned out to be the most difficult to satisfy, namely the minimum (Constraints 11) and maximum (Constraints 12) number of students per ward per period and the precedences among disciplines (Constraints 9). These constraints are taken care of by the cost function along with the objectives (to be maximized). As customary, the minimization of the hard constraint violations (also called *distance to feasibility*) is assigned a higher weight, in order to favor feasibility over optimality.

The initial solution is generated at random. In detail, for each student the procedure iterates on all groups of disciplines selecting the necessary number within the group. For each selected discipline, it selects the hospital and the period, among the student's free ones. Each selection is repeated until all constraints, but 9, 11, and 12 are satisfied. These three constraints are not checked given that, as mentioned above, they can be violated.

5.2 Neighborhood relations

We consider two distinct neighborhood relations, that will be used in combination as explained below. In detail, the first one considers a student and changes their assignment of one discipline, whereas the second one swaps two distinct assignments of a student. They are defined as follows:

- Change (C). The move $\mathsf{C}\langle s, p, w, p', w'\rangle$ reassigns the student s from the period p at ward w to a new period p' and a new ward w'. The move has the precondition that s is currently idle in p', unless $p = p'$; in the latter case the move represents a reassignment of the ward in the current period p. It is also possible that $w = w'$, so that the student remains in the same ward, but at a different time. Conversely, it is not possible that $p = p'$ and $w = w'$, which would result in a null move.
- Swap (S). The move $\mathsf{S}\langle s, p, w, p', w'\rangle$ swaps the assignments w and w' of student s in the two distinct periods p and p'. The precondition here is that the student is assigned in both periods, i.e., $w \neq -1$ and $w' \neq -1$.

The neighborhood relation employed is the set union of Change and Swap, and the random move selection is guided by a parameter σ (called *swap rate*), such that a Swap move is drawn with probability σ and a Change move with probability $1 - \sigma$.

Table 4 Instances features of Dataset 1

Instances	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{D}\rvert$	Du_d	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{H}\rvert$	P	St	B	Ds	Dp
Instance_10–14	40	12	1	12	3	1	0.000	0.796–0.863	0.136	3
Instance_20–24	80	12	1	12	6	1	0.000	0.790–0.849	0.136	3
Instance_30–34	40	12	1	12	3	1	0.000	0.879–0.921	0.106	2
Instance_40–44	80	12	1	12	6	1	0.000	0.889–0.923	0.106	2
Instance_50–54	40	12	1	12	3	1	0.000	0.794–0.878	0.091	2
Instance_60–64	80	12	1	12	6	1	0.000	0.837–0.862	0.091	2
Instance_70–74	40	12	1	12	3	1	0.000	0.808–0.851	0.136	3
Instance_80–84	80	12	1	12	5	1	0.000	0.816–0.851	0.136	3
Instance_90–94	40	12	1	12	3	1	0.000	0.869–0.942	0.106	2
Instance_100–104	80	12	1	12	5	1	0.000	0.900–0.906	0.106	2
Instance_110–114	40	12	1	12	3	1	0.000	0.841–0.909	0.091	2
Instance_120–124	80	12	1	12	5	1	0.000	0.860–0.902	0.091	2
Instance_L10–14	40	12	2	12	4	1	0.000	0.948–0.970	0.136	3
Instance_L20–24	40	12	4	24	4	1	0.000	0.933–0.971	0.136	3
Instance_L30–34	40	24	2	24	2	1	0.000	0.977–0.986	0.047	3
Instance_L40–44	40	24	4	48	2	1	0.000	0.970–0.985	0.047	3
Instance_L50–54	80	12	2	12	8	1	0.000	0.942–0.956	0.136	3
Instance_L60–64	80	12	4	24	8	1	0.000	0.938–0.957	0.136	3
Instance_L70–74	80	24	2	24	4	1	0.000	0.977–0.981	0.047	3
Instance_L80–84	80	24	4	48	5	1	0.000	0.971–0.985	0.047	3

Note that the Swap neighborhood alone would not be suitable as it does not allow for introducing new disciplines in a student's schedule, meaning that the search space is not connected under such neighborhood. On the other hand, Change alone has problems traversing the search space effectively when the schedule of a student is particularly dense because it does not have enough empty spots to move the disciplines. For these reasons, the use of the combined neighborhood is more effective for this problem.

We use several auxiliary data structures in order to accelerate the computation of the difference of cost between two neighbor states (the so-called *delta costs*). In detail, we employ redundancy within the data to represent both the information stored in the input and the state. This redundancy enables us to choose the most appropriate data representation for each situation.

As an example, we define an integer-valued matrix that stores the number of students working in a particular ward during a specific period. This specific representation enables us to calculate the delta costs associated with the violations of Constraints 11 and 12 using a constant number of operations. More specifically, since both neighborhood relationships change at most two assignments at a time, the effect of a potential move can be measured by performing four lookups of this matrix. Without this representation, we would need to scan an entire column of the state representation to compute the number of students available in a particular ward during a specific period, leading to a number $O(|S|)$ of operations for each delta cost computation.

The presence of redundancy in data structures results in an increased number of operations required when moving to a new state, as well as greater memory usage. However, considering that the frequency of state updating is significantly lower compared to the overall number of evaluated delta costs, this trade-off is advantageous.

5.3 Simulated Annealing

As the guiding metaheuristic, we use simulated annealing (SA), which already turned out quite effective in several timetabling problems (see, e.g., Bellio et al., 2021).

Our SA procedure starts from a random initial solution and draws, at each iteration, a random move. As customary for SA, the move is always accepted if it is improving or sideways (i.e., same cost), whereas when worsening it is accepted based on time-decreasing exponential distribution.

The temperature starts from a high value T_0 and is decreased according to the classical geometric cooling scheme $T_{i+1} \leftarrow \alpha T_i$. However, in order to speed up the early stages of the SA procedure, we use the *cut-off* mechanism: The temperature is decreased when either the number of sampled moves reaches its threshold (N_s) or the number of accepted moves is reached. This latter number is expressed as a ratio of N_s, i.e. equal to $N_s \rho$ with $0 \leq \rho < 1$ as a new parameter.

Table 5 Instances features of Dataset 2

Instance	$\|\mathcal{S}\|$	$\|\mathcal{D}\|$	Du_d	$\|\mathcal{T}\|$	$\|\mathcal{H}\|$	P	St	B	Ds	Dp
40_12_1	40	12	1	12	3	0.917	0.221	0.602	0.076	1
40_12_2	40	12	2	24	2	0.583	0.267	0.653	0.182	2
40_12_4	40	12	4	48	3	1.000	0.074	0.563	0.121	2
40_24_1	40	24	1	24	4	0.875	0.810	0.642	0.018	1
40_24_2	40	24	2	48	3	0.792	0.488	0.639	0.018	1
40_24_4	40	24	4	96	5	0.500	0.180	0.640	0.014	1
80_12_1	80	12	1	12	4	0.750	0.176	0.609	0.121	2
80_12_2	80	12	2	24	2	0.917	0.069	0.606	0.136	2
80_12_4	80	12	4	48	4	0.500	0.165	0.651	0.121	2
80_24_1	80	24	1	24	3	0.792	0.188	0.648	0.011	1
80_24_2	80	24	2	48	4	0.292	0.387	0.633	0.004	1
80_24_4	80	24	4	96	3	0.458	0.299	0.681	0.014	1
160_12_1	160	12	1	12	4	0.750	0.056	0.640	0.167	3
160_12_2	160	12	2	24	2	0.917	0.102	0.635	0.091	2
160_12_4	160	12	4	48	2	0.583	0.023	0.638	0.212	4
160_24_1	160	24	1	24	2	0.583	0.396	0.602	0.004	1
160_24_2	160	24	2	48	4	0.792	0.092	0.654	0.007	1
160_24_4	160	24	4	96	2	0.542	0.173	0.633	0.011	1
240_12_1	240	12	1	12	2	0.917	0.731	0.603	0.061	2
240_12_2	240	12	2	24	4	1.000	0.107	0.599	0.076	3
240_12_4	240	12	4	48	2	0.583	0.070	0.666	0.076	2
240_24_1	240	24	1	24	3	0.583	0.390	0.618	0.011	1
240_24_2	240	24	2	48	3	0.625	0.178	0.623	0.007	1
240_24_4	240	24	4	96	5	0.583	0.052	0.623	0.004	1
320_12_1	320	12	1	12	3	0.917	0.382	0.632	0.152	2
320_12_2	320	12	2	24	2	0.750	0.022	0.663	0.212	2
320_12_4	320	12	4	48	3	0.833	0.056	0.645	0.152	2
320_24_1	320	24	1	24	3	0.833	0.064	0.625	0.007	1
320_24_2	320	24	2	48	4	0.667	0.063	0.641	0.014	1
320_24_4	320	24	4	96	4	0.875	0.062	0.665	0.014	1

The search is stopped when a total number of iterations $\mathcal{I}$ has been performed, which guarantees that the running time is the same for all configurations of the parameters. To this aim, we compute N_s from I and the other parameters, using the following formula where T_f is the parameter representing the final temperature.

$$N_s = \mathcal{I} \Big/ \left(\frac{\log\left(T_f / T_0\right)}{\log \alpha} \right)$$

5.4 Integration with the MIP model

The MIP model shown in Sect. 3 has been implemented in CPLEX, using the Concert Technology which provides a C++ interface. This interface has been used to warm-start the CPLEX solver with the solutions provided by local search. We could not use the models developed by Akbarzadeh and Maenhout (2021b) because their source code unfortunately has not been made available to us.

The integration between the MIP solver and SA is obtained by executing a short run of SA to obtain a feasible solution, which is supplied to CPLEX as the initial solution. This has the twofold aim to speed up the CPLEX solver with respect to solving from scratch, and to obtain some lower bounds for assessing the quality of our SA solutions.

6 Experimental results

The software was implemented in C++ and compiled using g++ (v. 9.4). The experiments were run on an AMD Ryzen Threadripper PRO 3975WX 32-Cores (3.50 GHz, 64GB RAM) with Ubuntu Linux 20.4. One single core was dedicated to each experiment.

Table 6 Parameter tuning

Parameter	Description	Range	Winning
T_0	Start temperature	[10 – 80]	18.64
T_f	Final temperature	[0.5 – 1.5]	0.87
α	Cooling rate	[0.985 – 0.995]	0.99
ρ	Accepted moves ratio	[0.05 – 0.15]	0.08
σ	swap rate	[0.1 – 0.3]	0.11

Table 7 Comparative results on Dataset 1 between different solution methods and time limits

	CPU [s]	Gap (%)	Opt (%)
MIP	8054	9	50
CP	2630	1	88
DP	166	0	100
SA_5	2.5	0.09	98.1
SA_{10}	4.9	0.07	99.3
SA_{20}	9.7	0.07	99.8
SA_{50}	24.1	0.06	99.9
SA_{100}	48.2	0.00	100.0

The source code is available at https://bitbucket.org/satt/mss_data, along with all instances (in.dzn format) and their best solutions. The repository also contains a Constraint Programming (CP) model in MiniZinc, that, as customary for CP also makes use of integer-value variables rather than only the binary ones of the MIP.

6.1 Parameter tuning

The tuning procedure was performed on dedicated instances of various sizes, created by our generator specifically for this purpose. We used the tool JSON2RUN (Urli, 2013), which performs the F-Race procedure (Birattari et al., 2010) to select the best configuration. The winning configuration is shown in Table 6, which also reports the initial ranges, set based on preliminary experiments.

6.2 Comparative results

Table 7 shows the comparison between the methods presented by Akbarzadeh and Maenhout (2021b, Table 8), i.e. a mixed integer programming (MIP) formulation, a constraint programming (CP) formulation and a dynamic programming (DP) method, and our method based on simulated annealing with increasing number of iterations ($\mathcal{I} = 10^6 k$ with $k \in [5, 10, 20, 50, 100]$), denoted by SA_k. The table reports for each method the average results obtained on all original instances in terms of computational time in seconds (CPU), final optimality gap (Gap) and percentage of instances solved to optimality (Opt) within the time limit.[1]

6.3 Results on larger instances

The instances of the Dataset 2 were solved using both Simulated Annealing with a number of iterations $\mathcal{I}$ equal to 500 M (SA_{500M}) and the integrated approach (SA_{50M}+MIP) discussed in Sect. 5.4 that combines simulated annealing with $\mathcal{I} = 50M$ and mixed integer programming with timeout set to 1 h.

We run 30 repetitions for SA_{500M}, and we report in Table 8 best and average values of the objective function (all runs ended with feasible solutions). We made one single run of SA_{50M}+MIP_{1h}, and we report the initial value obtained by SA_{50M}, the final value of the integrated method, and the upper bound (UB) computed by CPLEX. Column MIPgap reports the percentage gap between the best result of SA_{500M} and the value of SA_{50M}+MIP_{1h}, and column UBgap reports the gap between the best result of SA_{500M} and the UB.

The table shows that MIP improves compared to the result reached by SA_{50M} in only three instances, and it never improves the best results found by SA_{500} in any of the execution (see that MIPgap is never negative). Moreover, the longest execution of SA_{500M} requires 739 s, considerably less than the 3600 s time limit set for the combined search.

We also notice that the average gap with the upper bound UBgap is less than 10%, for the instances in which a UB is reached. This can be considered a good achievement, given that the upper bounds are obtained by the standard procedure of CPLEX, without any guarantee of being tight.

We do not include the results obtained by the MIP model starting from scratch, given that it was not able to produce any feasible solution within 1 h of computation.

6.4 Analyses and insights

In this section, we analyze the importance of the objective component related to the fairness. To this aim, we run experiments with $\beta = 0$, i.e., without considering the fairness component in the objective function, and we compare the results with the ones obtained with the standard configuration ($\beta = 1$).

In Dataset 1, it turned out that the values of both Des_s and Des_{min} remain unchanged regardless of whether the fairness component is included in the objective or not. This means that there is no actual trade-off between satisfying the

[1] The methods by Akbarzadeh and Maenhout (2021b) are implemented in the ILOG-OPL IBM (v. 12.9) environment and the time limit imposed is 5760 s for instances with 40 students and 11520 s for those with 80 students (on a Windows i7 PC at 3.40GHz with 8GB RAM). Their source code is not available so we could not run it on our environment.

Table 8 Results on Dataset 2. The — means that CPLEX has run out of memory, the × is used when the ratio cannot be computed

Instance	SA_{500M}			SA_{50M}+MIP_{1h}			MIPgap	UBgap
	best (max)	avg	time [s]	z_{SA}	z_{SA+MIP}	UB	(%)	(%)
40_12_1	4127	4126.7	243	4127	4127	4237	0.00	2.60
40_12_2	3920	3908.1	379	3899	3899	4236	0.54	7.46
40_12_4	2862	2856.1	383	2850	2850	3361	0.42	14.85
40_24_1	7799	7776.6	289	7698	7698	8377	1.31	6.90
40_24_2	7303	7264.8	410	7129	7129	8700	2.44	16.06
40_24_4	7096	7061.3	574	6984	6984	8680	1.60	18.25
80_12_1	8614	8612	254	8608	8608	8884	0.07	3.04
80_12_2	7206	7205	307	7206	7206	7634	0.00	5.61
80_12_4	8097	8075.8	478	8006	8006	9306	1.14	12.99
80_24_1	16453	16447.1	275	16437	16437	17087	0.10	3.71
80_24_2	15906	15874.1	466	15716	15716	17581	1.21	9.53
80_24_4	12704	12616.7	730	12069	12069	17327	5.26	26.68
160_12_1	17748	17742.2	266	17726	17728	18316	0.11	3.10
160_12_2	15085	15073	331	15057	15057	16200	0.19	6.88
160_12_4	14002	13989.3	477	13940	13942	15995	0.43	12.46
160_24_1	31470	31446.5	304	31381	31381	32824	0.28	4.13
160_24_2	35579	35561	359	35486	35486	37631	0.26	5.45
160_24_4	28412	28329.4	704	27943	27943	34138	1.68	16.77
240_12_1	20099	20059	275	19822	19829	21842	1.36	7.98
240_12_2	23183	23167.4	321	23144	23144	24552	0.17	5.58
240_12_4	21167	21150.7	529	21060	21060	24337	0.51	13.03
240_24_1	50278	50241.2	305	50066	50066	52642	0.42	4.49
240_24_2	45864	45840.4	382	45701	45701	$> 10^6$	0.36	×
240_24_4	48471	48421.1	481	—	—	—	×	×
320_12_1	31897	31869.9	284	31789	31789	33763	0.34	5.53
320_12_2	29895	29885.4	359	29873	29873	32261	0.07	7.33
320_12_4	32281	32259.6	447	32201	32201	36006	0.25	10.35
320_24_1	65943	65937.7	294	65920	65920	68011	0.03	3.04
320_24_2	67772	67740.5	375	—	—	—	×	×
320_24_4	66372	66333.9	519	—	—	—	×	×

general preferences and obtaining a fair distribution. This phenomenon is explained by the fact that the value of Des_{min} reflects the score achievable by a specific student that has a low number of disciplines to attend and has given low weights to their preferences.

These observations are also confirmed by the results obtained on Dataset 2, in which, however, as shown by Table 9, there are some exceptions. While the differences between Des_{min} values are relatively limited on most instances there are a few of them for which it is remarkable. Notice also that there are some negative fairness values due to the predominance of the subtractive components (hospital changes and waiting times) in the objective function.

In conclusion, we believe that the notion of fairness proposed by Akbarzadeh and Maenhout (2021b) and used in this work, which is based on the minimization of the maximum discomfort (min–max), needs to be refined or replaced by more complex definitions of fairness (see, e.g., Bertsimas et al., 2011).

Finally, we analyzed the impact of minimum requirements (constraint 11). To this aim, we performed additional experiments with SA on Dataset 2 only, given that minimum requirements are absent in Dataset 1. In particular, we investigated the hardness of instances and the quality of the solution with and without such constraints. We do not report the full results here, but the general output is that the time to obtain a feasible solution is on average 6 times shorter removing the constraint, going from 0.49s to 0.08s, with a maximum time that drops from 12.2s to 0.47s. On the contrary, the effect on the objective function of removing constraint 11 is rather limited, and we record an average improvement of just 2.8%.

Table 9 Average results on Dataset 2 with and without the fairness component

Instance	$\beta = 1$		$\beta = 0$		Des_{min} gap
	Des_s	Des_{min}	Des_s	Des_{min}	
40_12_1	4114.5	12.0	4115.0	12.0	0.0
40_12_2	3903.5	3.0	3906.4	3.0	0.0
40_12_4	2850.8	4.0	2851.2	4.0	0.0
40_24_1	7754.9	40.0	7757.7	37.7	2.3
40_24_2	7306.7	−17.3	7311.3	−20.4	3.1
40_24_4	7060.8	3.5	7065.9	−17.5	21.0
80_12_1	8598.2	14.0	8597.5	14.0	0.0
80_12_2	7196.6	9.0	7195.9	9.0	0.0
80_12_4	8076.4	−2.0	8073.9	−2.0	0.0
80_24_1	16411.1	35.0	16412.8	35.0	0.0
80_24_2	15859.6	12.0	15864.8	12.0	0.0
80_24_4	12663.4	14.5	12670.6	−15.7	30.2
160_12_1	17735.4	8.0	17733.4	8.0	0.0
160_12_2	15076.8	−3.0	15078.4	−3.0	0.0
160_12_4	14011.4	−14.0	1412.4	−14.0	0.0
160_24_1	31415.9	30.0	31418.3	30.0	0.0
160_24_2	35538.3	25.0	35539.6	25.0	0.0
160_24_4	28349.5	−29.0	28348.9	−29.0	0.0
240_12_1	20077.7	9.0	20095.3	7.8	16.8
240_12_2	23155.7	11.0	23153.3	11.0	0.0
240_12_4	21198.1	−47.0	21196.2	−47.0	0.0
240_24_1	50230.4	12.0	50245.9	12.0	0.0
240_24_2	45834.2	9.0	45830.9	9.0	0.0
240_24_4	48495.0	−54.0	48492.6	−54.0	0.0
320_12_1	31871.8	9.8	31865.2	6.5	16.3
320_12_2	29882.8	1.0	29889.2	1.0	0.0
320_12_4	32299.9	−40.0	32303.4	−40.0	0.0
320_24_1	65924.2	15.0	65922.6	15.0	0.0
320_24_2	67740.0	10.0	67728.0	10.0	0.0
320_24_4	66387.2	−47.0	66382.4	−47.0	0.0

7 Conclusions

We have developed a local search approach for the MSS problem, which turned out to be able to find the optimal solution for all publicly available instances. To test it on more competitive ground, we have developed an instance generator and have created challenging instances. These novel instances are more difficult not only because they are considerably larger, but also because they activate constraints that are not used in the original dataset. Both for verification and for hybridization purposes, we have also implemented the mathematical model in CPLEX, which provided some interesting upper bounds.

As a by-product of this work, we have designed a new data format based on MiniZinc, which is both more robust and more human-readable than the original (text-only) one. All instances and solutions are available in the new format on BitBucket, along with the source code of our search methods.

Our future work will follow mainly three directions. First, we plan to improve the efficacy of our local search approach by designing additional neighborhood relations, that could potentially improve the exploration of the search space.

Secondly, we plan to design new forms of hybridization between metaheuristic and exact search method, more sophisticated than the simple warm-start of MIP experimented in this paper.

Finally, we plan to extend our approach to other versions of the problem, in order to enlarge the set of real-world cases that our search method can solve. For example, we plan to consider the case in which the durations of the internships differ from each other. Similarly, we plan to scale to a larger (regional) level, in which traveling and accommodation of students in different regions need to be taken into account.

Acknowledgements We thank Babak Akbarzadeh and Broos Maenhout for answering our questions about their work. This research has been funded by the Italian Ministry of University and Research under the action PRIN 2020, project "Models and algorithms for the optimization of integrated healthcare management".

Funding Open access funding provided by Universitá degli Studi di Udine within the CRUI-CARE Agreement.

Appendix

In this section, we report the detailed list of all corrections we made to the mathematical model formulated in Akbarzadeh and Maenhout (2021b, Section 3.2).

- The domain of the decision variables Des_s and Des^{min} is not $\mathbb{R}_+$, indeed Des_s and Des^{min} can also assume negative values given that in Constraint (18) the last two components related to waiting times and hospital changes are negative.
- In Constraints (3, 5, 7, 10, 11, 12), in the first summation on the set of disciplines, the dummy discipline must be excluded (i.e., $\sum_{d \in \mathcal{D}} \rightarrow \sum_{d \in \mathcal{D}^0}$).

- Similarly, Constraints (8, 9, 14, 16, 17) must be imposed $\forall d \in \mathcal{D}^0$ (excluding the dummy discipline) instead of $\forall d \in \mathcal{D}$. For Constraint (15), the same correction must be done on $\overline{d}$.
- The RHS part of Constraint (9) has been extended with the term $Sk_{\overline{d}d} \cdot M \left(1 - \sum_{t \in \mathcal{T}} \sum_{h \in \mathcal{H}} v_{s\overline{d}th}\right)$, which was missing.
- In Constraints (11–12), the summation on time periods has been modified as follows $\sum_{\overline{t}=t-\mathrm{Du}_d+1}^{t} \rightarrow \sum_{\overline{t}=\max(0,t-\mathrm{Du}_d+1)}^{t}$ to avoid negative values of t.
- Constraint (20) has been added to the model to force the dummy discipline to start at time period 0.
- Constraint (21) has been added to guarantee that the dummy discipline is assigned to only one period and hospital for each student.

References

Akbarzadeh, B., & Maenhout, B. (2021). A decomposition-based heuristic procedure for the medical student scheduling problem. *European Journal of Operational Research, 288*(1), 63–79.

Akbarzadeh, B., & Maenhout, B. (2021). An exact branch-and-price approach for the medical student scheduling problem. *Computers and Operations Research, 129*, 105209.

Akbarzadeh, B., Wouters, J., Sys, C., & Maenhout, B. (2022). The scheduling of medical students at Ghent university. *INFORMS Journal on Applied Analytics, 52*(4), 303–323.

Bard, J. F., Shu, Z., Morrice, D. J., & Leykum, L. K. (2016). Annual block scheduling for internal medicine residents with 4+1 templates. *Journal of the Operational Research Society, 67*(7), 911–927.

Beliën, J., & Demeulemeester, E. (2007). On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem. *Annals of Operations Research, 155*(1), 143–166.

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

Bertsimas, D., Farias, V. F., & Trichakis, N. (2011). The price of fairness. *Operations research, 59*(1), 17–31.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated F-race: An overview. *Experimental Methods for the Analysis of Optimization Algorithms* (pp. 311–336). Berlin: Springer.

Brech, C. H., Ernst, A., & Kolisch, R. (2019). Scheduling medical residents' training at university hospitals. *European Journal of Operational Research, 274*(1), 253–266.

Cire, A. A., Diamant, A., Yunes, T., & Carrasco, A. (2019). A network-based formulation for scheduling clinical rotations. *Production and Operations Management, 28*(5), 1186–1205.

Cohn, A., Root, S., Kymissis, C., Esses, J., & Westmoreland, N. (2009). Scheduling medical residents at Boston university school of medicine. *Interfaces, 39*(3), 186–195.

Franz, L. S., & Miller, J. L. (1993). Scheduling medical residents to rotations: Solving the large-scale multiperiod staff assignment problem. *Operations Research, 41*(2), 269–279.

Goodman, M. D., Dowsland, K. A., & Thompson, J. M. (2012). Hybridising GRASP and network flows in the solution of a medical school scheduling problem. *Journal of Scheduling, 15*(6), 717–731.

Güler, M. G., Idi, K., Güler, E. Y., et al. (2013). A goal programming model for scheduling residents in an anesthesia and reanimation department. *Expert Systems with Applications, 40*(6), 2117–2126.

Guo, J., Morrison, D. R., Jacobson, S. H., & Jokela, J. A. (2014). Complexity results for the basic residency scheduling problem. *Journal of Scheduling, 17*(3), 211–223.

Kraul, S. (2020). Annual scheduling for anesthesiology medicine residents in task-related programs with a focus on continuity of care. *Flexible Services and Manufacturing Journal, 32*(1), 181–212.

Kraul, S., Fügener, A., Brunner, J. O., & Blobner, M. (2019). A robust framework for task-related resident scheduling. *European Journal of Operational Research, 276*(2), 656–675.

Lemay, B., Cohn, A., Epelman, M., & Gorga, S. (2017). New methods for resolving conflicting requests with examples from medical residency scheduling. *Production and Operations Management, 26*(9), 1778–1793.

Proano, R. A., & Agarwal, A. (2018). Scheduling internal medicine resident rotations to ensure fairness and facilitate continuity of care. *Health Care Management Science, 21*(4), 461–474.

Resende, M. G., & Ribeiro, C. C. (2016). *Optimization by GRASP*. New york: Springer.

Seizinger, M., & Brunner, J. O. (2023). Optimized planning of nursing curricula in dual vocational schools focusing on the German health care system. *European Journal of Operational Research, 304*(3), 1223–1241.

Smalley, H. K., & Keskinocak, P. (2016). Automated medical resident rotation and shift scheduling to ensure quality resident education and patient care. *Health Care Management Science, 19*(1), 66–88.

Stuckey, P.J., Marriott, K., & Tack, G. (2022). The minizinc handbook. https://www.minizinc.org/.

Topaloglu, S. (2006). A multi-objective programming model for scheduling emergency medicine residents. *Computers & Industrial Engineering, 51*(3), 375–388.

Topaloglu, S., & Ozkarahan, I. (2011). A constraint programming-based solution approach for medical resident scheduling problems. *Computers & Operations Research, 38*(1), 246–255.

Urli, T. (2013). json2run: a tool for experiment design & analysis. CoRR arXiv:1305.1112

White, C.A., & White, G.M. (2003). Scheduling doctors for clinical training unit rounds using tabu optimization. In: Practice and Theory of Automated Timetabling IV, Springer, pp 120–128.

Zheng, Z., Gong, X., & Liu, X. (2016). A two-phase heuristic approach for solving trainee rotation assignment problem at a local school of nursing. In: Proc. of the International Conference on the Practice and Theory of Automated Timetabling (PATAT-2016), pp 421–438.