Annals of Operations Research
https://doi.org/10.1007/s10479-025-06678-6

ORIGINAL - OR MODELING/CASE STUDY

Computing job-tailored degree plans towards the acquisition of professional skills

Roger X. Lera-Leri[1] · **Filippo Bistaffa**[1] · **Tomas Trescak**[2] · **Juan A. Rodríguez-Aguilar**[1]

Received: 15 June 2024 / Accepted: 14 May 2025

Abstract
Sensibly planning the subjects to study during a university degree is one of the most crucial tasks that impact the future professional life of a student. Nonetheless, to the best of our knowledge, no automated solution is available for students who want to plan their desired degree path and maximize the skills required by desired or target job(s). In this paper, we consider the *Degree Planning Problem* (DPP), which aims at computing degree plans composed of university subjects for students during the completion of an undergraduate degree. Specifically, we aim to obtain the best set of skills matching the requirements of students' preferred job(s). To achieve this objective, we propose a flexible and scalable approach that solves the DPP in real-time by means of a non-trivial formalization as an optimization problem that can be solved with standard solvers. Finally, we employ real data from our University's Bachelor in Information and Communications Technology to show, through several use cases, that our approach can be a valuable decision-support tool for students and curriculum designers.

Keywords OR in education · Decision support systems · Degree planning

✉ Roger X. Lera-Leri
rlera@iiia.csic.es

Filippo Bistaffa
filippo.bistaffa@iiia.csic.es

Tomas Trescak
t.trescak@westernsydney.edu.au

Juan A. Rodríguez-Aguilar
jar@iiia.csic.es

[1] Artificial Intelligence Research Institute (IIIA-CSIC), 08193 Bellaterra, Catalonia, Spain

[2] Western Sydney University, Sydney 2751, Australia

Published online: 02 June 2025

1 Introduction

The United Nations' 2030 Agenda for Sustainable Development includes quality education as a Sustainable Development Goal. This goal calls for the development of technologies and tools that make education management and provision more equitable, inclusive, and *personalized* (Trescak et al., 2022). Indeed, personalization in education has recently been the focus of significant research efforts within the scientific community, mainly from the perspective of *learning analytics* (Siemens and Baker, 2012; Gašević et al., 2015; Tsai and Gasevic, 2017). Learning analytics research focuses on measuring, collecting, analyzing, and reporting data about learners and their contexts to understand and optimize learning and the environments in which it occurs. In this context, one of the most active and relevant areas of research concerns *Educational Recommender Systems* (EdRecSys) (Drachsler et al., 2015). One of the main topics of interest for EdRecSys research has been recommending university courses since planning the set of courses to study during a university degree is one of the most crucial tasks that impact the future professional life of a student. Indeed, in a recent survey carried out by the educational platform Forage,[1] 1000 U.S. university students were interviewed about several aspects of education in relation to their professional prospects. One of the most important findings of this survey was that 71% of the students evaluated as *extremely important* to "identify the most important skills [needed] to land [their] dream job", but only 39% considered the "tools or information to do this" as adequate. Along these lines, developing an automated approach to guide students throughout their educational pathway toward the acquisition of the skills required by their preferred job(s) is key.

Of course, the problem of planning a set of activities within a time horizon has also been extensively studied in the planning and scheduling literature. However, despite the abundance of works in these fields (especially on the *Resource-Constraint Project Scheduling Problem* (RCPSP) (Pass-Lanneau et al., 2023; Karnebogen and Zimmermann, 2024; Etminaniesfahani et al., 2023) and the *Balanced Academic Curriculum Problem* (BACP) (Castro and Manzano, 2001; Hnich et al., 2002; Ceschia et al., 2014), as discussed in detail in Sect. 7), to the best of our knowledge, the problem of computing a degree path that allows a student to acquire the skills required by their desired job(s) has not been studied yet.

Against this background, in this paper, we propose an automated solution (outlined in Fig. 1) able to provide recommendations to students during the completion of an undergraduate university degree by solving a novel planning problem, a problem that we denote as *Degree Planning Problem* (DPP). Specifically, the goal of the DPP is to compute degree plans that achieve the set of skills that best cover the requirements of the job(s) preferred by a student, i.e., maximizing the so-called *job affinity*. More precisely, we make the following contributions to the state-of-the-art:

- We propose a formal definition of the DPP as an optimization problem. Solving such optimization problem yields the degree paths that maximize the "coverage" of the professional skills required by a student's desired job(s). While some aspects of our model are based on our real-world test case, it is straightforward to adapt it to other university curricula without changing the proposed solution method.
- We present a non-trivial linearization of our optimization problem so that we can solve it with standard optimization tools.

[1] https://www.theforage.com/blog/news/forage-career-readiness-survey, accessed on 23/05/2024.

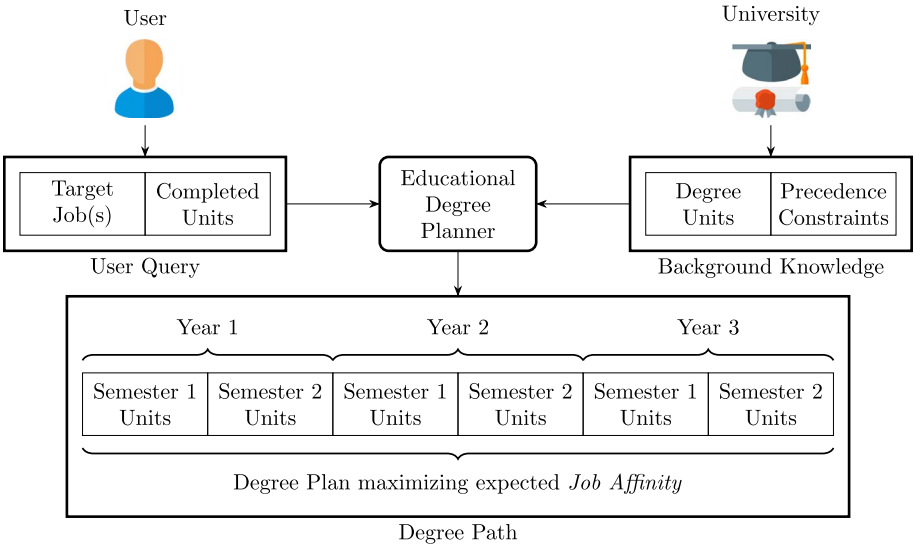

Fig. 1 Overview of our automated solution approach. We refer the reader to Sects. 3 and 4 for the technical details

- We show, through several use cases, that our approach can be a valuable decision-support tool for (i) students planning their academic pathways (even when they are uncertain about their desired job or they have already initiated their academic career) and (ii) curriculum designers assessing how well a university's degrees satisfy the skill requirements of the job market. With this aim, we employ real-world data from the Bachelor Degree of Information and Communications Technology of Western Sydney University and the *Skills Framework for the Information Age* (SFIA) dataset (British Computer Society, 2021).*Organization.* In Sect. 2, we discuss the concepts of *Mixed-Integer Programming* (MIP) and p-norms, which provide the basis for our formalization of the DPP discussed in Sect. 3. We show how we solve the DPP by casting it as a MIP problem in Sect. 4. In Sect. 5, we empirically analyze real-world data considering different potential use cases. In Sect. 6, we discuss the real-world applicability of our approach and we propose some extensions to other university curricula. Sect. 7 positions our work with respect to the existing literature, and Sect. 8 concludes the paper and discusses future research directions.

2 Background

In this section, we discuss some background concepts that we employ to formalize the DPP and are useful to position our theoretical contributions. Specifically, in Sect. 2.1, we introduce MIP and the subclasses of optimization problems that are interesting in the context of our work. Then, in Sect. 2.2, we discuss the concept of p-norms, which we employ in the formulation of the objective function of the DPP, and we present different approaches to solving p-norm minimization problems.

2.1 MIPs and related sub-classes of optimization problems

An MIP is an optimization problem in the form

$$\text{minimize} \quad f_0(\mathbf{x}) \tag{1}$$

$$\text{subject to} \quad f_i(\mathbf{x}) \leq 0, \quad \forall i = 1, \ldots, m, \tag{2}$$

$$f_i(\mathbf{x}) \sim \mathbf{0}, \quad \forall i = m+1, \ldots, m+k, \tag{3}$$

where $\mathbf{x} \in \mathbb{Z}^n$, and $f_i : \mathbb{Z}^n \to \mathbb{R}$ is any real-valued function. In general, MIP problems are NP-Hard (Wolsey, 2020) and are extremely expensive to solve since off-the-shelf solvers such as CPLEX or Gurobi cannot tackle them directly. On the other hand, CPLEX and Gurobi can tackle some particular sub-classes of optimization problems that, while remaining NP-Hard, can be solved much more efficiently.

Specifically, *Mixed-Integer Linear Programming* (MILP) is the class of MIP problems whose objective and constraints are linear, i.e.,

$$\text{minimize} \quad \mathbf{c}^T\mathbf{x}, \tag{4}$$

$$\text{subject to} \quad D\mathbf{x} \leq \mathbf{d}, \tag{5}$$

$$A\mathbf{x} \sim \mathbf{b}, \tag{6}$$

where $\mathbf{c} \in \mathbb{R}^n$, $D \in \mathbb{R}^{m \times n}$, $\mathbf{d} \in \mathbb{R}^m$, $A \in \mathbb{R}^{k \times n}$, and $\mathbf{b} \in \mathbb{R}^k$. Other classes of MIPs are *Mixed-Integer Quadratic Programming* (MIQP) problems, which minimize a quadratic objective function, and *Mixed-Integer Quadratically Constrained Programming* (MIQCP) problems, whose solutions are constrained to quadratic functions such that

$$\text{minimize} \quad \mathbf{x}^T P\mathbf{x} + \mathbf{c}^T\mathbf{x}, \tag{7}$$

$$\text{subject to} \quad D\mathbf{x} + \mathbf{x}^T Q\mathbf{x} \leq \mathbf{d}, , \tag{8}$$

$$A\mathbf{x} \sim \mathbf{b}, \tag{9}$$

where $Q, P \in \mathbb{R}^{n \times n}$ are positive semi-definite matrices.

2.2 Objective functions involving *p*-norms

A p-norm (Boyd and Vandenberghe, 2004) is defined as

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}, \tag{10}$$

where $p \geq 1$. *Norm Minimization Problems* (NMPs) are a family of optimization problems well-known in the literature (Boyd and Vandenberghe, 2004; Chakrabarty and Swamy, 2019) defined as

$$\text{minimize} \quad \|A\mathbf{x} - \mathbf{b}\|_p, \tag{11}$$

$$\text{subject to} \quad \mathbf{x} \in \mathcal{F}, \tag{12}$$

where $A \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathcal{F}$ is the set of feasible solutions. For a better understanding of our discussion in Sect. 4, it is particularly important to note that the argument of the p-norm must be of linear form. NMPs are particularly interesting because they can be transformed either to a MILP, a MIQPs, or a MIQCP depending on the value of p (see Fig. 2), i.e., they are solvable with off-the-shelf solvers for any p.[2]

The most common p-norms used in the literature are $p = \{1, 2, \infty\}$(Boyd and Vandenberghe, 2004; Salas-Molina et al., 2023). In the case of minimizing the *Manhattan* norm ($p = 1$), i.e., $\|A\mathbf{x} - \mathbf{b}\|_1 = |A_1\mathbf{x} - b_1| + \cdots + |A_m\mathbf{x} - b_m|$, the problem can be cast as an MILP such that

$$\text{minimize} \quad \mathbf{1}^T\mathbf{t}, \tag{13}$$

$$\text{subject to} \quad -\mathbf{t} \leq A\mathbf{x} - \mathbf{b} \leq \mathbf{t}, \tag{14}$$

$$\mathbf{x} \in \mathcal{F}, \tag{15}$$

where $\mathbf{t} \in \mathbb{R}^m$ is a vector of auxiliary variables (Boyd and Vandenberghe, 2004). In the minimization of the Maximum norm ($p = \infty$), i.e., $\|A\mathbf{x} - \mathbf{b}\|_\infty = \max_{i=1,\ldots,m} \left(|A_i\mathbf{x} - b_i|\right)$, the problem can be also cast as an MILP such that

$$\text{minimize} \quad t, \tag{16}$$

$$\text{subject to} \quad -t\mathbf{1} \leq A\mathbf{x} - \mathbf{b} \leq t\mathbf{1}, \tag{17}$$

$$\mathbf{x} \in \mathcal{F}, \tag{18}$$

where $t \in \mathbb{R}$ is an auxiliary variable (Boyd and Vandenberghe, 2004). In addition, in the minimization of the Euclidean norm ($p = 2$), i.e., $\|A\mathbf{x} - \mathbf{b}\|_2$, the problem can be cast as an MIQP by squaring the 2-norm such that

$$\text{minimize} \quad \|A\mathbf{x} - \mathbf{b}\|_2^2 = \mathbf{x}^T A^T A\mathbf{x} - 2\mathbf{b}^T A\mathbf{x} + \mathbf{b}^T\mathbf{b}, \tag{19}$$

$$\text{subject to} \quad \mathbf{x} \in \mathcal{F}. \tag{20}$$

The optimization problems involving the minimization of norms with $p \notin \{1, 2, \infty\}$ can be solved by employing the techniques discussed by Alizadeh and Goldfarb (2001, Sec-

[2] The widely used optimization library CVXPY can automatically transform a NMP to the appropriate problem depending on p.

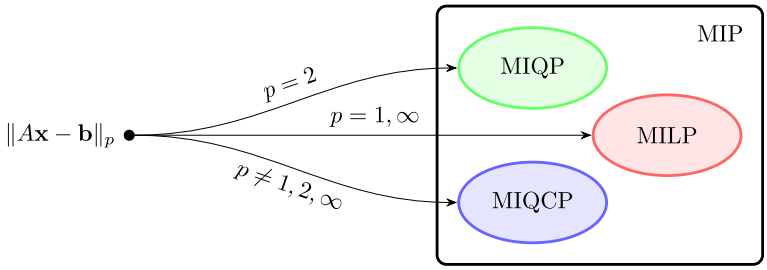

Fig. 2 Subclasses of MIPs and p-norm minimization problems depending on p

tion 2.3g), who show how to represent inequalities involving p-norms as MIQCP, hence enabling the use of off-the-shelf solvers such as CPLEX or Gurobi. Figure 2 summarizes the above discussion, showing the resulting optimization problems depending on the choice of p-norm.

Having provided the necessary background on the different optimization problems that appear in the rest of our paper, we now proceed, in Sect. 3, to formalize the DPP as an MIP. Then, in Sect. 4 we show how we transform our MIP formalization to a more manageable optimization problem involving a p-norm with a linear argument, as in (11).

3 Formalizing the degree planning problem as a MIP

The objective of the DPP is planning a feasible degree path (i.e., one that complies with the rules imposed by a university)[3] for a student to maximize the level of skills required by the job (or a set of jobs) preferred by the student.

3.1 Basic definitions

First, we consider a sequence of semesters $\mathcal{L} = \langle l_1, \ldots, l_n \rangle$. Each semester is associated with a season of the year in the set $\Gamma = \{\text{winter, spring, summer, autumn}\}$.[4] To represent such an association, we define function $\rho : \mathcal{L} \to \Gamma$, which assigns a season to each semester. We also consider a set of skills $\mathcal{S} = \{s_1, \ldots, s_w\}$. Then, we denote a university subject as *unit*, representing the fundamental element used to build a student's degree plan.

Definition 3.1 (*Unit*) A unit is a tuple $\langle S, c, \sigma, \gamma \rangle$, where $S \subseteq \mathcal{S}$ is the set of skills that a student obtains when coursing the unit, $c \in \mathbb{N}$ is the number of credits rewarded to a student completing the unit, $\sigma : S \to [1..lev_{max}]$ is a function that indicates the level for a given skill acquired when completing a unit (lev_{max} denotes the maximum skill level that can be

[3] Some aspects of our model (e.g., the major definition, the precedence relations, or the credits specification restrictions) are based on our real-world test case. Nonetheless, it is straightforward to adapt it to other university curricula without changing the proposed solution method.

[4] Notice that universities usually organize courses during two semesters: autumn and spring. However, some universities, such as the one considered in our test case, offer shorter semesters during summer vacations. Thus, for the sake of generality, we consider all the seasons.

achieved) and $\gamma : \mathcal{L} \to \{\top, \bot\}$ is a Boolean function that indicates whether a unit is offered during a given semester l_i(i.e., $\gamma(l_i) = \top$) or it is not offered (i.e., $\gamma(l_i) = \bot$).[5]

Henceforth, we denote the set of all units as $\mathcal{U}$,[6] We refer to the components of a unit $u \in \mathcal{U}$ by subscripts (e.g., S_u and c_u denote the skills and number of credits of unit *u*, while σ_u and γ_u refer to its skill level and offer functions).

It is common for degrees' syllabi to have requisites to course some subjects, as students may be required to possess some basic knowledge before pursuing a given subject. Hence, we introduce precedence relationships that hold between units. Following the structure of the real-world syllabi, we consider two types of precedence relationships:

- AND: a unit requires that *all* its prerequisite units have been completed.
- OR: a unit can be undertaken by a student if *at least one* of its prerequisite units has been completed.

Formally, we capture these two types of precedence relationships as follows.

Definition 3.2 (*AND precedence*) The AND precedence graph is a directed acyclic graph $(\mathcal{U}, E_{and})$, where $E_{and} \subseteq \mathcal{U} \times \mathcal{U}$. If $(u, u') \in E_{and}$, we say that unit *u* is a prerequisite of u'.

Definition 3.3 (*OR precedence*) The OR precedence between units is a function $or : 2^{|\mathcal{U}|} \to \mathcal{U}$. If $or(U) = u'$, with $U \subseteq \mathcal{U}$, we say that all units in U are an OR prerequisite of u'.

Notice that a unit can have multiple precedence relations of both the previously defined types.

In addition, we consider a subset of units $\mathcal{U}^{core} \subset \mathcal{U}$ called *core units*, which are mandatory for a student during their degree. Moreover, each student must pursue at least one *major*. A *major* is a specialization within the degree. We formally define a major as follows.

Definition 3.4 (*Major*) A major is a set of units $M \subset \mathcal{U}$ such that: (i) there is a set of core units $M^{core} \subseteq M$ to be completed; and (ii) there is a set of eligible units $M^{elig} \subset M$ distributed into subsets of units $SM^{elig} \subseteq M^{elig}$, such that a student has to choose a certain number of units $n_{SM^{elig}}$ for each SM^{elig}.

We denote the set of all majors by $\mathcal{M}$. Moreover, we introduce the notion of *credits specification* as a tuple $\langle c_T, \mathbf{d} \rangle$ where $c_T \in \mathbb{N}$ is the total number of credits to complete to obtain

[5] Universities have limited resources. Hence it is reasonable to assume that not all units will be offered during all semesters.

[6] In this work, we are taking the perspective of a single student, hence we assume that each unit that appears in the set $\mathcal{U}$ has the capacity of hosting such a student, i.e., we do not enforce any capacity constraint. With minimal changes to our model, one could easily consider the problem of recommending degree plans to *several students* in which capacity constraints could be enforced straightforwardly. Nonetheless, by taking such a multi-student perspective, the problem would resemble more of an *assignment* problem of students to units, which is not our purpose here. For this reason, we leave this exercise out of the current paper.

a bachelor and $\mathbf{d} \in \mathbb{N}^{|\mathcal{L}|}$ is a vector containing the number of credits that a student has to complete during a semester.

Finally, we define the notion of *job*.

Definition 3.5 (*Job*) A job j is a tuple $\langle S_j, \sigma'_j \rangle$, where $S_j \subseteq \mathcal{S}$ is the set of skills required to take on the job, and $\sigma'_j : S_j \rightarrow [1..lev_{max}]$ is a skill requirement function defining the minimum level required per skill to undertake the job.

3.2 Objective of the degree planning problem

We now characterize the objective of the DPP. Specifically, the DPP aims at finding the degree plan whose units help a student achieve the skills required by the job(s) pursued by the student. Formally, we define a target as follows.

Definition 3.6 (*Target*) Given a set $\mathcal{J}$ of jobs defined according to Definition 3.5, the target $\mathbf{t} \in \mathbb{N}^{|\mathcal{S}|}$ is a vector such that t_s represents the target level required by each skill s required by a job $j \in \mathcal{J}$. Hence, $t_s = \max_{j \in \mathcal{J}} \sigma'_j(s)$. We assume $t_s = 0$ for the skills that are not required.

Therefore, given a student who pursues the target $\mathbf{t}$, solving the DPP amounts to finding a degree plan as an *allocation of units to semesters*, such that: (i) the skills acquired by the student after completing the degree plan bring them as close as possible to the target, i.e., the degree plan maximizes a similarity measure (*job affinity*) between the achieved skills and the skills required by the job(s), and (ii) the student is not penalized for acquiring further skills not required by the job(s).[7] Therefore, the DPP amounts to solving

$$\text{minimize} \quad \left(\sum_{s \in \mathcal{S}} \left| \min \left(\max_{u,l} \left(\sigma_u(s) \cdot x_{u,l} \right) - t_s, 0 \right) \right|^p \right)^{\frac{1}{p}}, \tag{21}$$

where the binary decision variable $x_{u,l}$ denotes whether the unit u has been scheduled or not during semester l, and $\sigma_u(s)$ is the level of skill s for the unit corresponding to the variable $x_{u,l}$. The $(\sum |\cdot|^p)^{\frac{1}{p}}$ notation represents a p-norm that measures the *distance* between the vector of acquired skills and the target, where $p \in [1, \infty]$ is a metric parameter.

3.3 Constraints

After defining the objective of DPP, we now specify a number of feasibility constraints that must be included to comply with university rules. We remark that these constraints are specific to the degree structure of Western Sydney University. Nonetheless, adapting our formalization to other scenarios and implementing additional constraints by adopting a similar methodology is easy.

[7] Not penalizing further skills is reasonable because degrees are conceived to provide a wider range of skills than those strictly required. Furthermore, from the point of view of a student, acquiring further skills than required is an advantage: they make the student more competitive in the market, and they are valuable to change or evolve their professional career.

1. A unit cannot be scheduled more than once in a degree plan. We achieve this by defining a matrix $B \in \mathbb{N}^{|\mathcal{U}|\times|X|}$ where each row corresponds to a unit $u \in \mathcal{U}$ and each column to a decision variable $x_{u,l} \in X$ such that

$$B_{u,(u',l)} = \begin{cases} 1, & \text{if } u = u', \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

and by imposing the linear constraint

$$B\mathbf{x} \leq \mathbf{1}, \tag{23}$$

where $\mathbf{x} \in \{0,1\}^{|X|}$ is the vector of decision variables, and $\mathbf{1} \in \mathbb{N}^{|\mathcal{U}|}$ is a vector with all components equal to 1.

2. As mentioned in the basic definitions, some units are not offered during some semesters. We fulfil such constraint by defining a diagonal matrix $P \in \mathbb{N}^{|X|\times|X|}$ where each row corresponds to a decision variable $x_{u,l} \in X$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$P_{(u,l),(u',l')} = \begin{cases} 1, & \text{if } u = u' \text{ and } l = l' \text{ and } \gamma_u(l) = \perp, \\ 0, & \text{otherwise,} \end{cases} \tag{24}$$

and by imposing the linear constraint

$$P\mathbf{x} = \mathbf{0}, \tag{25}$$

where $\mathbf{0} \in \mathbb{N}^{|\mathcal{P}|}$ is a null vector.

3. The degree plan has to fulfill a total number of credits. We achieve this by defining a vector $\mathbf{c} \in \mathbb{N}^{|X|}$ where each position corresponds to the number of credits assigned to the unit associated with variable $x_{u,l} \in X$ such that $\mathbf{c}_{(u,l)} = c_u$, and by imposing the linear constraint

$$\mathbf{c}^T\mathbf{x} = c_T, \tag{26}$$

where c_T is the total number of credits.

4. In addition, during each semester, we have to assign a certain number of credits. We achieve this by defining a matrix $D \in \mathbb{N}^{|\mathcal{L}|\times|X|}$ where each row corresponds to a semester $l \in \mathcal{L}$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$D_{l,(u',l')} = \begin{cases} c(u'), & \text{if } l = l', \\ 0, & \text{otherwise,} \end{cases} \tag{27}$$

and by imposing the linear constraint

$$D\mathbf{x} = \mathbf{d}, \tag{28}$$

where $\mathbf{d} \in \mathbb{N}^{|\mathcal{L}|}$ is a vector whose components are the total amount of credits per semester.[8]

5. Now, we introduce the constraints related to the precedence relations between units. As mentioned in Sect. 3.1, two types of precedence relations exist: AND and OR. However, we can formalize the AND precedence relation as a specific case of OR precedence, where the set of prerequisites contains one unit for each AND precedence relation. Consequently, we denote $\mathcal{R}$ as the set of all precedence relations. To formally code precedence relations we define matrices $R, r \in \mathbb{N}^{|\mathcal{R}|\cdot|\mathcal{L}|\times|X|}$ where each row corresponds to a precedence relation $or(U) \in \mathcal{R}$ and a semester $l \in \mathcal{L}$ and each column to a decision variable $x_{u,l} \in X$ such that

$$R_{(or(U),l'),(u,l)} = \begin{cases} 1, & \text{if } u \in U \text{ and } l' > l, \\ 0, & \text{otherwise,} \end{cases} \tag{29}$$

$$r_{(or(U),l'),(u',l)} = \begin{cases} 1, & \text{if } u' = or(U) \text{ and } l' = l, \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

Then, we impose

$$R\mathbf{x} \geq r\mathbf{x}. \tag{31}$$

Concretely, this linear constraint enforces assigning one or more prerequisites $u \in U$ of unit $u' = or(U)$ in previous semesters than when this unit is assigned, in case u' is scheduled. If u' is not scheduled, the constraint does not enforce assigning any prerequisite.

6. All core units (i.e., $u \in \mathcal{U}^{core}$) must be scheduled. We achieve this by defining a matrix $E \in \mathbb{N}^{|\mathcal{U}^{core}|\times|X|}$ where each row corresponds to a unit $u \in \mathcal{U}^{core}$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$E_{u,(u',l')} = \begin{cases} 1, & \text{if } u = u', \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

and by imposing the linear constraint

$$E\mathbf{x} = \mathbf{1}, \tag{33}$$

where $\mathbf{1} \in \mathbb{N}^{|\mathcal{U}^{core}|}$ is a vector of 1.

7. Finally, a student has to complete at least one major. To enforce this constraint, we define three sub-constraints. First, we guarantee the assignment of at least one major by imposing the linear constraint

$$\mathbf{1}^T\mathbf{v} \geq 1, \tag{34}$$

[8] Notice that (26) is self-contained in (28). However, we impose both constraints in case a user wants to impose a lower or upper bound threshold for constraint (28) such that $d_{min} \leq D\mathbf{x} \leq d_{max}$, which is a common real-world case. In that case, constraint (26) is needed.

where the auxiliary variable $v_M = 1$ indicates that the plan will contain the requirements to complete major M. Then, we enforce to schedule all core units of the major we assign, namely M^{core}. We achieve this by defining a matrix $F \in \mathbb{N}^{|\mathcal{U}_{\mathcal{M}}^{core}| \times |X|}$, where $\mathcal{U}_{\mathcal{M}}^{core}$ denote the set of core units of all majors, and each row corresponds to a core unit $u \in M^{core}$ for a major $M \in \mathcal{M}$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$F_{u,(u',l')} = \begin{cases} 1, & \text{if } u = u', \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

We then define a matrix $f \in \mathbb{N}^{|\mathcal{U}_{\mathcal{M}}^{core}| \times |\mathcal{M}|}$ where each each row corresponds to a core unit $u \in M^{core}$ for a major $M \in \mathcal{M}$ and each column to a decision variable v_M such that

$$f_{u,v_M} = \begin{cases} 1, & \text{if } u \in M^{core}, \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

Then, we impose the linear constraint

$$F\mathbf{x} \geq f\mathbf{v}. \tag{37}$$

Concretely, this constraint enforces scheduling core units of a major when this is coursed. Otherwise, the units can be undertaken, but it is not a requirement. Furthermore, we consider that major have different subsets of elective units, namely $\mathcal{SM}^{elig}$. By $\mathcal{SM}^{elig}$, we denote the set of elective unit subsets. In order to enforce that at least $n_{SM^{elig}}$ units $u \in SM^{elig}$ are assigned, we define a matrix $G \in \mathbb{N}^{|\mathcal{SM}^{elig}| \times |x|}$ where each row corresponds to a subset of electives units $SM^{elig} \in \mathcal{SM}^{elig}$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$G_{SM^{elig},(u,l')} = \begin{cases} 1, & \text{if } u \in SM^{elig}, \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

Then, we define a vector $g \in \mathbb{N}^{|\mathcal{SM}^{elig}| \times |\mathcal{M}|}$ where each row corresponds to a subset of electives units $SM^{elig} \in \mathcal{SM}^{elig}$ and each column to a decision variable $x_{u',l'} \in X$ such that

$$g_{SM^{elig},v_M} = \begin{cases} n_{SM^{elig}}, & \text{if } SM^{elig} \text{ is associated with } M, \\ 0, & \text{otherwise,} \end{cases} \tag{39}$$

where $n_{SM^{elig}}$ is the minimum number of elective units completed per subset SM^{elig}. Then, we impose

$$G\mathbf{x} \geq g\mathbf{v}. \tag{40}$$

Concretely, this linear constraint enforces that the required number of elective units are scheduled if a given major M is coursed. Otherwise, elective units can be scheduled, but it is not a requirement.

3.4 Completed units

In our formalization discussed so far, we assume that (i) no unit had been completed and (ii) there were no changes in the academic curriculum along the degree plan. However, in practice, students might have completed some units and want to receive a new plan for the remaining part of their bachelor's degree. The reasons for computing a new plan might be multiple: students might want to target a different desired job(s), conditions in the curriculum might change, and students might need to compute a new valid degree plan (some units might not be offered anymore, new units might be offered, skill levels acquired by units might change, prerequisites of the units have changed, etc.), or the required skills for a given job might change. Thus, we must consider the scenario where students are in the middle of the degree and have completed some units. To do so, we first formally define the concept of a completed unit.

Definition 3.7 (*Completed units*) A completed unit is a tuple $\langle u, l\rangle$ where u is the completed unit and l is the semester when the unit has been completed. We denote the set of all completed units by $\mathcal{Q}$.

Then, we define a matrix $Q \in \mathcal{N}^{|\mathcal{Q}|\times|X|}$ where each row corresponds to a completed unit $\langle u, l\rangle \in \mathcal{Q}$ and each column to a variable $x_{u',l'} \in X$ such that

$$Q_{(u,l),(u',l')} = \begin{cases} 1, & \text{if } u = u' \text{ and } l = l, \\ 0, & \text{otherwise.} \end{cases} \tag{41}$$

Finally, we consider completed units by imposing the linear constraint

$$Q\mathbf{x} = \mathbf{1}. \tag{42}$$

As a conclusion, by combining the objective function in (21) with the collection of constraints (23), (25), (26), (28), (31), (33), (34), (37), (40), and (42), we obtain the following MIP formalization of the DPP:

$$\begin{aligned} \text{minimize} \quad & \left(\sum_{s\in\mathcal{S}} \left| \min\left(\max_{u,l} \left(\sigma_u(s)\cdot x_{u,l}\right) - t_s, 0\right)\right|^p \right)^{\frac{1}{p}}, \\ \text{subject to} \quad & B\mathbf{x} \le \mathbf{1}, \quad P\mathbf{x} = \mathbf{0}, \quad \mathbf{c}^T\mathbf{x} = c_T, \quad D\mathbf{x} = \mathbf{d}, \\ & R\mathbf{x} \ge r\mathbf{x}, \quad E\mathbf{x} = \mathbf{1}, \quad \mathbf{1}^T\mathbf{v} \ge 1, \\ & F\mathbf{x} \ge f\mathbf{v}, \quad G\mathbf{x} \ge g\mathbf{v}, \quad Q\mathbf{x} = \mathbf{1}. \end{aligned} \tag{43}$$

4 Solving DPP by linearizing the argument of the *p*-norm

We are now ready to detail our approach to solve (43). With that aim, we show how to cast the MIP formulation of the DPP as a NMP problem with a linear argument within a p-norm objective function. Specifically, our goal is to linearize the argument of the p-norm, i.e., $\min\left(\max_{u,l}\left(\sigma_u(s)\cdot x_{u,l}\right)-t_s,0\right)$, in the objective function of (43). This will allow us to solve the DPP as one of the manageable optimization problems discussed in Sect. 2.2.

In detail, we discuss how we reformulate the DPP in two steps. First, we consider the goal of ensuring that the skills acquired by a student are as close as possible to their target job(s). Hence we simplify (21) to

$$\text{minimize} \quad \left(\sum_{s\in\mathcal{S}}\left|\max_{u,l}\left(\sigma_u(s)\cdot x_{u,l}\right)-t_s\right|^p\right)^{\frac{1}{p}}, \tag{44}$$

for the sake of clarity. Second, we consider the goal of not penalizing the acquisition of non-compulsory skills for the target job(s), whose objective function is (21). To address the first goal, we introduce a new set of auxiliary, binary decision variables $y_{s,lev}$ with $s=1,\ldots,|\mathcal{S}|$ and $lev=1,\ldots,lev_{max}$. Given a degree path (obtained by setting some decision variables $x_{u,l}$ to 1), setting $y_{s,lev}$ to 1 indicates that the degree plan achieves at least level *lev* for skill s. Furthermore, on the one hand, we introduce a matrix of binary values $A\in\mathbb{N}^{lev_{max}\cdot|\mathcal{S}|\times|X|}$ that encodes the skill levels acquired by each unit. Thus, by setting $A_{(s,lev),(u,l)}$ to 1, we indicate that if unit u is allocated to semester l, it helps a student achieve at least level *lev* for skill s (this is because $\sigma_u(s)\geq lev$). Otherwise, we set $A_{(s,lev),(u,l)}$ to 0. On the other hand, we introduce another matrix of binary values $K\in\mathbb{N}^{|\mathcal{S}|\times lev_{max}\cdot|\mathcal{S}|}$ to encode that decision variables $\{y_{s,lev}\mid lev=1,\ldots,lev_{max}\}$ refer to skill s. Thus, we set $K_{s',(s,level)}$ to 1 when $s=s'$, and to 0 otherwise.

We are now ready to formulate a version of the DPP that only addresses the goal of selecting a degree path *as close as possible to* the target job(s):

$$\text{minimize} \quad \|K\mathbf{y}-\mathbf{t}\|_p, \tag{45}$$

$$\text{subject to} \quad A\mathbf{x}\leq\Lambda\cdot\mathbf{y}, \tag{46}$$

$$A\mathbf{x}\geq\Lambda\cdot(\mathbf{y}-\mathbf{1})+\mathbf{1}. \tag{47}$$

and also subject to the side constraints in the DPP defined in the previous section, i.e., constraints (23), (25), (26), (28), (31), (33), (34), (37), (40), and (42). Notice that $\Lambda\in\mathbb{R}$ is a large enough number to ensure the non-violation of constraints (Griva et al., 2009).[9] Notice also that the introduction of constraints (46) and (47) force to set an auxiliary variable $y_{s,lev}$ to 1 when there is some unit u scheduled during some semester l in the degree plan ($x_{u,l}=1$) whose completion leads to level *lev* for skill s (i.e., $\sigma_u(s)\geq lev$); and to 0 otherwise. Then, the product $K\mathbf{y}$ obtains the maximum level for each skill in the degree plan.

[9] To ensure this condition, it is enough to consider $\Lambda=|X|$.

Now, the objective function above is already an p-norm with a linear argument. However, this formulation penalizes acquiring skills not required by the target job(s). We will address this issue next. We achieve this by reformulating the cost function in (45) as follows:

$$\text{minimize} \quad \|\min(K\mathbf{y} - \mathbf{t}, \mathbf{0})\|_p. \tag{48}$$

This new formulation ensures that acquiring skills not required by the target job(s) does not contribute to the cost function. However, notice that we have introduced non-linearity in the argument of the p-norm of the objective function in (48), which we must eliminate to have a convex objective function. We achieve that by introducing a new set of auxiliary decision variables $z_s \in \mathbb{Z}$, with $s = 1, \ldots, |S|$. Decision variable z_s takes the value of the *level shortage* for skills s considering the acquired level of s thanks to a degree plan and the required level of s for the target job(s). Therefore, we enforce that z_s is set to 0 when a given degree plan leads to acquiring a greater (or equal) level for skill s than required by the target job(s) ($Ky_s \geq t_s$). Otherwise, z_s is set to difference, shortage, between acquired skills and required skills ($Ky_s - t_s$).

With these new variables, we reformulate the objective in (48) as follows:

$$\text{minimize} \quad \|\mathbf{z}\|_p, \tag{49}$$

$$\text{subject to} \quad \mathbf{z} \leq K\mathbf{y} - \mathbf{t}, \tag{50}$$

$$\mathbf{z} \leq \mathbf{0}. \tag{51}$$

Notice that the argument of the p-norm in the objective function (49) is linear. Therefore, we are in the position of formalizing the DPP as an NMP, which considers the objective function (49), the constraints required by the original formulation of the DPP, and the new constraints added by the reformulation in this section, i.e., (46), (47), (50), (51). Formally:

$$\begin{aligned}
&\text{minimize} \quad \|\mathbf{z}\|_p, \\
&\text{subject to} \quad \mathbf{z} \leq \mathbf{0}, \quad \mathbf{z} \leq K\mathbf{y} - \mathbf{t}, \quad A\mathbf{x} \leq \Lambda \cdot \mathbf{y}, \\
&\qquad\qquad A\mathbf{x} \geq \Lambda \cdot (\mathbf{y} - \mathbf{1}) + \mathbf{1}, \quad B\mathbf{x} \leq \mathbf{1}, \quad P\mathbf{x} = \mathbf{0}, \\
&\qquad\qquad \mathbf{c}^T\mathbf{x} = c_T, \quad D\mathbf{x} = \mathbf{d}, \quad R\mathbf{x} \geq r\mathbf{x}, \quad E\mathbf{x} = \mathbf{1}, \\
&\qquad\qquad \mathbf{1}^T\mathbf{v} \geq 1, \quad F\mathbf{x} \geq f\mathbf{v}, \quad G\mathbf{x} \geq g\mathbf{v}, \quad Q\mathbf{x} = \mathbf{1}.
\end{aligned} \tag{52}$$

Although the formulation above uses four different sets of decision variables ($\mathbf{x}$, $\mathbf{v}$, $\mathbf{y}$ and $\mathbf{z}$), we remark that our primary decision variables are $\mathbf{x}$, which are the ones employed to build a degree plan.

Since (52) is an NMP, we can solve it with the solution techniques discussed in Sect. 2.2.

4.1 Equivalence between MIP and NMP

Having presented such a non-trivial linearization of our optimization problem, we now provide the formal proof of the equivalence of the initial MIP (43) and the final NMP (52). To do so, we must prove that: (i) the formalization in (52) does not change the space of feasible

solutions of $\mathbf{x}$; and (ii) the ordering of each pair of feasible solutions in both problems is the same. The latter statement is necessary to guarantee that our transformation preserves optimal solutions.

To prove statement (i), we formally define the space of feasible solutions.

Definition 4.1 (*Space of feasible solutions*) The space of feasible solutions $\mathcal{F}$ is the set of points within the domain of the optimization problem for which all the constraints are satisfied (Boyd and Vandenberghe, 2004).

Formally, for the formalization (43) of DPP, we define $\mathcal{F}$ as follows.

$$\mathcal{F}_{\mathcal{C}} = \Big\{ \mathbf{x} \in \{0,1\}^{|X|} \,|\, \mathbf{x} \quad \text{satisfies} \quad \mathcal{C} \Big\}, \tag{53}$$

where $\mathcal{C}$ is the set of constraints (23), (25), (26), (28), (31), (33), (34), (37), (40), and (42). To prove (ii), we demonstrate that both objective functions are equivalent, which is a *sufficient* condition to guarantee (ii).

For clarity, we proceed in two steps as done in Sect. 4. We first demonstrate that the formalization (45) is equivalent to (44). Secondly, we demonstrate that the formalization (49) is equivalent to (48).

Step 1. First, we introduce Lemma 4.1.

Lemma 4.1 *The space of feasible solutions of problems* (44) *and* (45) *are the same.*

Proof Notice that the final problem (45) considers the set of constraints in the initial problem $\mathcal{C}$ plus constraints (46) and (47). For the sake of simplicity we use the notation $\mathcal{C}'$ for the set of constraints

$$\mathcal{C}' = \mathcal{C} \cup \{A\mathbf{x} \leq \Lambda \cdot \mathbf{y}\,,\; A\mathbf{x} \geq \Lambda \cdot (\mathbf{y} - 1) + \mathbf{1}\}.$$

Formally, we define the space of feasible solutions of problem (45) as

$$\mathcal{F}_{\mathcal{C}'} = \Big\{ \mathbf{x} \in \{0,1\}^{|X|} \,|\, \mathbf{x} \;\text{ satisfy} \quad \mathcal{C}' \Big\}. \tag{54}$$

Due to the fact that problem (45) considers two additional constraints, $\mathcal{F}_{\mathcal{C}'} \subseteq \mathcal{F}_{\mathcal{C}}$ is ensured. Now, we evaluate whether constraints (46) and (47) make that $\mathcal{F}_{\mathcal{C}'}$ contains less feasible solutions than $\mathcal{F}_{\mathcal{C}}$. Therefore we examine constraints (46) and (47), which can be expressed as

$$\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} \leq \Lambda \cdot y_{s,lev}, \tag{55}$$

$$\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} \geq \Lambda \cdot (y_{s,lev} - 1) + 1, \tag{56}$$

for all $s \in \mathcal{S}$; $lev = 1, \ldots, lev_{max}$, where $U_{s,lev} \subseteq \mathcal{U}$ is the set of units u that provides skill s such that when u is completed, the student acquires a skill level higher or equal than *lev*, i.e., $\sigma_u(s) \geq lev$. Thus, $\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} > 0$ indicates that after completing the considered degree plan, a student has acquired at least level *lev* for skill s. We remark that $y_{s,lev} \in \{0, 1\}$ is a binary auxiliary variable which is set to 1 only when the degree plan has scheduled a unit whose completion leads to at least level *lev* for skill s. Considering that $\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} \in [0..\Lambda]$, we derive (55) and (56) for both possible values of $y_{s,lev}$ such that

$$\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} \leq \begin{cases} 0, & \text{if } y_{s,lev} = 0, \\ \Lambda, & \text{if } y_{s,lev} = 1. \end{cases}$$

$$\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} \geq \begin{cases} 1 - \Lambda, & \text{if } y_{s,lev} = 0, \\ 1, & \text{if } y_{s,lev} = 1. \end{cases}$$

for all $s \in \mathcal{S}$; $lev = 1, \ldots, lev_{max}$. As we can see, for any value $\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l}$ there exists only one value of $y_{s,lev}$ that satisfies the constraints. Hence, problem (33) does not lose solutions when constraints (46) and (47) are added.[10] Thus, we conclude that $\mathcal{F}_{\mathcal{C}'} = \mathcal{F}_{\mathcal{C}}$. □

Now, we prove (ii): that the ordering of each pair of feasible solutions in both problems is the same. To do so, it is sufficient to prove Lemma 4.2.

Lemma 4.2 *The objective function* (45) *is equivalent to* (44)*, i.e.*

$$\|K\mathbf{y} - \mathbf{t}\|_p = \left(\sum_{s \in \mathcal{S}} \left| \max_{u,l} \left(\sigma_u(s) \cdot x_{u,l} \right) - t_s \right|^p \right)^{\frac{1}{p}}$$

Proof First, we develop (45):

$$\begin{aligned} \|K\mathbf{y} - \mathbf{t}\|_p &= \left(\sum_{s \in \mathcal{S}} \left| K_s\mathbf{y} - t_s \right|^p \right)^{\frac{1}{p}} \\ &= \left(\sum_{s \in \mathcal{S}} \left| \sum_{lev=1}^{lev_{max}} y_{s,lev} - t_s \right|^p \right)^{\frac{1}{p}}. \end{aligned}$$

From the discussion in Proof 4.1, we remark that $y_{s,lev}$ can only be set to 1 when at least one unit that delivers level *lev* (or higher) for skill s is recommended, namely is part of the solution. For example, if there is planned a unit u that delivers level i for skill s (and no other scheduled unit gives a higher level for skill s), i.e., $\sigma_u(s) = i$, then $y_{s,lev} = 1$, for $lev = 1, \ldots, i$; and $y_{s,lev} = 0$, for $lev = i + 1, \ldots, lev_{max}$. Hence, $\sum_{lev=1}^{lev_{max}} y_{s,lev} = i$ for skill s, which indicates that at least there is a unit u which $\sigma_u(s) = i$ is part of the solution. If we consider all scheduled units in the degree plan instead of only a unit, $\sum_{lev=1}^{lev_{max}} y_{s,lev}$

[10] Notice that when $\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} > 0$, only $y_{s,lev} = 1$ satisfies constraints (55) and (56). Otherwise, when $\sum_{u \in U_{s,lev}} \sum_{l \in \mathcal{L}} x_{u,l} = 0$, only $y_{s,lev} = 0$ satisfies constraints (55) and (56).

is equivalent to obtaining the maximum level of skill s of all scheduled units, which can be encoded as $\max_{u,l}\left(\sigma_u(s)\cdot x_{u,l}\right)$. Thus,

$$\begin{aligned}\|K\mathbf{y}-\mathbf{t}\|_p &= \left(\sum_{s\in\mathcal{S}}\left|\sum_{lev=1}^{lev_{max}} y_{s,lev}-t_s\right|^p\right)^{\frac{1}{p}}\\ &= \left(\sum_{s\in\mathcal{S}}\left|\max_{u,l}\left(\sigma_u(s)\cdot x_{u,l}\right)-t_s\right|^p\right)^{\frac{1}{p}}.\end{aligned}$$

□

Then, we can introduce Theorem 4.3.

Theorem 4.3 *Problems* (44) *and* (45) *are equivalent.*

Proof The proof is immediate by applying Lemmas 4.1 and 4.2. □

Step 2. Now, we discuss that both (48) and (52) formalizations are equivalent. First, we remark that the space of feasible solutions $\mathcal{F}$ of formalization (48) is equal to the one of (45). Second, we define the space of feasible solutions $\mathcal{F}_{\mathcal{C}''}$ of formalization (52) such that

$$\mathcal{F}_{\mathcal{C}''}=\left\{\mathbf{x}\in\{0,1\}^{|X|}\,|\,\mathbf{x}\ \text{satisfy}\quad \mathcal{C}''\right\}, \tag{57}$$

where $\mathcal{C}''$ is the set of constraints such that

$$\mathcal{C}''=\mathcal{C}'\cup\{\mathbf{z}\leq\mathbf{0}\,,\,\mathbf{z}\leq K\mathbf{y}-\mathbf{t}\}.$$

Then, we introduce Lemma 4.4.

Lemma 4.4 *The space of feasible solutions of problems* (48) *and* (52) *are the same.*

Proof Thus, we must evaluate whether constraints (50) and (51) make $\mathcal{F}_{\mathcal{C}''}$ contain less feasible solutions over auxiliary variable $\mathbf{y}$. Note that both constraints can be expressed as

$$z_s\leq\sum_{lev=1}^{lev_{max}} y_{s,lev}-t_s, \tag{58}$$

$$z_s\leq 0, \tag{59}$$

$\forall\, s\in\mathcal{S}$. From (59) we remark that we encode $\mathbf{z}$ as a non-positive vector, i.e., $\mathbf{z}\in\mathbb{Z}^{-|\mathcal{S}|}$. From the discussion in Proof 4.1, we notice that $0\leq\sum_{lev=1}^{lev_{max}} y_{s,lev}\leq lev_{max}$. Notice that from Sect. 3 we remark that $0\leq t_s\leq lev_{max}$. Hence, for any value $-lev_{max}\leq\sum_{lev=1}^{lev_{max}} y_{s,lev}-t_s\leq lev_{max}$ there exist a value z_s that

satisfies the constraints. Thus, the formalization (52) does not lose feasible solutions when adding (50) and (51), i.e., $\mathcal{F}_{\mathcal{C}''} = \mathcal{F}_{\mathcal{C}'} = \mathcal{F}_{\mathcal{C}}$. □

Having proved Lemma 4.4 and guaranteed (i), we now move to Lemma 4.5.

Lemma 4.5 *The objective function* (48) *is equivalent to* (49)*, i.e.,*

$$\| \min(K\mathbf{y} - \mathbf{t}, \mathbf{0})\|_p = \|\mathbf{z}\|_p.$$

Proof To prove so, we examine both the constraints (58), (59) and the objective function (49). We remark that $\mathbf{z} \in \mathbb{Z}^{|\mathcal{S}|}$ is a non-positive set of auxiliary variables due to constraint (59). However, the domain of $\mathbf{z}$ can be even more restricted by (58) when $\sum_{lev=1}^{lev_{max}} y_{s,lev} - t_s < 0$. In addition, we know that we aim at minimizing the absolute values of vector $\mathbf{z}$ to a power. Therefore, $\mathbf{z}$ variables' values will be the maximum values of their domain restricted by constraints (58) and (59). Formally:

$$\begin{aligned} z_s &= \begin{cases} \sum_{lev=1}^{lev_{max}} y_{s,lev} - t_s, & \text{if } \sum_{lev=1}^{lev_{max}} y_{s,lev} - t_s < 0, \\ 0, & \text{otherwise,} \end{cases} \\ &= \min\left(\sum_{lev=1}^{lev_{max}} y_{s,lev} - t_s, 0 \right) \\ &= \min(K_s\mathbf{y} - t_s, 0). \end{aligned}$$

$\forall\, s \in \mathcal{S}$. As we have shown, both objective functions are equivalent and, consequently, (ii) is guaranteed. □

Finally, we can prove Theorem 4.6.

Theorem 4.6 *Problems* (43) *and* (52) *are equivalent.*

Proof By direct applications of Lemmas 4.1, 4.2, 4.4 and 4.5. □

5 Experimental analysis

The main objective of this section is to show the potential of our approach as a valuable decision-supporting tool for students and curriculum designers. To this end, we consider four scenarios we deem representative and realistic use cases for our approach. First, we focus on a student at the beginning of their degree who wants to plan the best path to achieve the skills required by some desired job(s). Secondly, we consider a student who wants to identify the jobs that are more *attainable* in the job market after completing the degree. Thirdly, we focus on the case when a student may be uncertain about the desired target job, hence they specify a set of multiple candidate jobs to receive a broader recommendation by our algorithm. Finally, we consider a student who has already started their degree and requests recommendations to move ahead. The student will request degree plans that lead to maximizing the acquisition of the skills required by some desired job.

5.1 Dataset

Our use cases employ a real-world dataset of skills, i.e., the SFIA skill framework (British Computer Society, 2021). We consider version 8 of this dataset (with 121 skills), adopted by governments and institutions in Australia,[11] the USA, the UK, and the EU to define jobs and positions that require a certain level of professional skills. As to degree data, we consider a real-world bachelor's degree, i.e., the Bachelor of Information and Communication Technology (ICT) of the Western Sydney University,[12] which involves $|\mathcal{U}| = 82$ units ($\mathcal{U}^{core} = 16$ of which are core units) and 7 different majors, together with the corresponding precedence constraints among units. Importantly, our dataset contains the courses in the syllabus of the Bachelor of ICT together with information on the skills acquired when completing each course. For that, we tasked university experts with associating, by hand, each unit with the skills in the SFIA framework that a student would acquire after completing the unit. Moreover, experts were asked to set the level of each skill acquired when completing a unit. Each unit rewards a student with 10 credits (i.e., $c_u = 10\ \forall u \in \mathcal{U}$). To obtain a Bachelor of ICT, a student must obtain $c_T = 240$ credits, completing 40 credits each semester. Therefore, a student completes a bachelor's degree in 6 semesters. Finally, we consider a dataset of 118 jobs obtained from the real-world *APS Digital Career Pathways* dataset (Australian Digital Transformation Agency, 2019), which specifies the level of SFIA skills required by each job.

5.2 Implementation

Even though our model can be used with any p-norm, following the discussion by Salas-Molina et al. (2023, Section 6), we restrict our attention to $p \in \{1, 2, \infty\}$, since those are the values that can be better semantically characterized. Specifically, the optimal solution for $p = 1$ aims to minimize the *total* (or, equivalently, the *average*) residual $r_i = z_i$, while for $p = 2$ we aim to minimize the *standard deviation* of the residual. Finally, the optimal solution for $p = \infty$ aims to minimize the maximum residual. Among the three above-mentioned values of p, we argue that $p = 1$ is the best-suited for our cases since we aim to reduce the total gap between the acquired skills and those required by the target job. For $p = 1$, the DPP amounts to a MILP. We remark that solving DPP for different values of p is indeed technically possible, but such an exercise has been left out for the sake of simplicity.

We implement and solve our problem in (52) with CPLEX v22.1.0.[13] We run our tests on a machine with an 8-core 1.00GHz CPU and 16.0GB of RAM. The average runtime for computing the optimal solution is 0.025 ± 0.001 seconds in all our experiments for $p = 1$. We also evaluate the computational benefit of solving the initial problem in (43) to solving our transformed problem in (52). We do so by comparing the runtime to solve the problem in (52) with the ILOG CPLEX solver to the runtime required to solve the original problem in (43) with the ILOG CP solver. We observe that while the optimal solution is computed in 0.025 seconds on average with our approach, the average runtime using the CP solver is 10.4 ± 2.2 seconds, i.e., our approach outperforms the CP solver by 2 orders of magnitude.

[11] https://sfia-online.org/en/tools-and-resources/standard-industry-skills-profiles.

[12] https://hbook.westernsydney.edu.au/programs/bachelor-information-communications-technology/.

[13] Our implementation is available at https://github.com/RogerXLera/DegreePlanning.

At this point, is important to mention that computing degree plans in real time has value from an application perspective. Indeed, we conceive our DPP as part of an *interactive* software application for students and curriculum designers. Therefore, such an application must rapidly respond to requests issued by students and curriculum designers.

5.3 Use case: beginning-of-degree planning

In our first use case, we consider students at the beginning of their degree (i.e., $\mathcal{Q} = \emptyset$); thus, they need a plan for all semesters of their degrees. Therefore, the goal of our first experiment is to show the degree paths computed by our approach depending on the skills required by two selected example jobs: *Penetration Tester* (a highly-specific and technical job) and *Software Engineer* (a more general job). Table 1 shows the elective units of the computed degree paths coursed in each semester,[14] where the numbers are semesters' labels and "Au" and "Sp" stand for Autumn and Spring semesters, respectively. Our results show that, for the *Penetration Tester* job, the *Cybersecurity* major is planned to be coursed as the one that best matches the skills required. For the *Software Engineer* job, our approach plans to pursue the *Mobile Computing* major. Table 2 shows the skills acquired in each of the above-mentioned degree paths and the units providing such skills after completion. For both degree paths, we see that there are units related to the major that provide the required skills, justifying the pursuit of that major. For the *Penetration Tester* degree path example, we show that the *Ethical Hacking Principles and Practices* unit, which is compulsory to complete the *Cybersecurity*

Table 1 Elective subjects computed for the degree plans targeting *Penetration Tester* and *Software Engineer* jobs

Penetration Tester		*Major: Cybersecurity*
Semester	Unit	Unit type
2 (Sp)	Object Oriented Programming	Elective
3 (Au)	**Data Structures and Algorithms**	Major
4 (Sp)	**Network Security**	Major
5 (Au)	**Discrete Mathematics**	Elective
5 (Au)	Ethical Hacking Principles and Practice	Major
6 (Sp)	Computer Organisation	Major
6 (Sp)	Information Security	Major
6 (Sp)	**Formal Software Engineering**	Elective
Software Engineer		*Major: Mobile Computing*
Semester	Unit	Unit type
2 (Sp)	Social Computing	Elective
3 (Au)	**Data Structures and Algorithms**	Elective
3 (Au)	**Discrete Mathematics**	Elective
4 (Sp)	Information Systems Deployment and Management	Major
4 (Sp)	Mobile Applications Development	Major
5 (Au)	Wireless and Mobile Networks	Major
6 (Sp)	**Network Security**	Major
6 (Sp)	**Formal Software Engineering**	Elective

Units within both degree plans are in **bold**

[14] We do not show the degree path with the core units for the sake of brevity.

Table 2 Skills required by the considered jobs and their Required Level (RqL), along with the Achieved Level (AcL) after completing the computed degree path and the unit that provides such skill level

Penetration Tester ($\alpha = 93\%$)			
Skill	RqL	AcL	Unit
Systems Design	4	4	Professional Experience
Software Design	3	3	**Formal Software Engineering**
Programming	3	3	**Object Oriented Programming** & *Data Structures and Algorithms*
Penetration Testing	4	3	*Ethical Hacking Princ. and Practice*
Software Engineer ($\alpha = 85\%$)			
Skill	RqL	AcL	Unit
Porting	4	3	*Mobile Applications Development* & Web Systems Development
Testing	3	2	*Mobile Applications Development* & Web Systems Development
Methods and Tools	4	4	**Formal Software Engineering**
Systems Integration	5	3	*Info. Systs. Depl & Manag.* & Web Systems Development
Systems Design	4	4	Professional Experience
Software Design	3	3	**Formal Software Engineering**
Programming	4	4	*Mobile Applications Development*

Units in black are core units. Units in italic are electives that belong to the major program selected to complete by the approach. Units in bold are electives that do not belong to the major program selected by the approach

major, is the only unit that gives 3 out of 4 levels of the *Penetration Testing* skill. In addition, there are elective units that help students to achieve the required level for some skills. It is the case for the *Formal Software Engineering* unit, which provides students with level 3 of the *Software Design* skill and level 4 of the *Methods and Tools* skill. Other elective units are planned since they are the prerequisite of other electives. For instance, *Discrete Mathematics* is a prerequisite of *Formal Software Engineering*.

In such examples, a student would achieve almost all required skills. However, we observe that some skills required by more multidisciplinary jobs cannot be covered. This result is expected due to the lack of units that can provide some skills or their required level. Indeed, it is reasonable to expect that, for some technical and multidisciplinary jobs, students might not reach the required level of skills after completing their bachelor's. Nonetheless, our approach allows students to start their professional careers in the best position based on the curriculum offered by the university.

5.4 Use case: most attainable jobs on the market

In our second use case, the student's goal is to identify the degree plans (computed by our algorithm) whose completion leads to the most attainable jobs on the market. For this purpose, we first consider a *job affinity* function that measures the achieved skills with respect to the ones required by the target job(s).

Definition 5.1 (*Job affinity*) Given a target vector of skills $\mathbf{t}$ and a degree plan encoded by the decision variable $\mathbf{z}$, we define the job affinity function $\alpha(\mathbf{t}, \mathbf{z})$ as

$$\alpha(\mathbf{t}, \mathbf{z}) = 100 \cdot \frac{\|\mathbf{t} + \mathbf{z}\|_1}{\|\mathbf{t}\|_1}. \tag{60}$$

We define α in terms of $\mathbf{z}$, which measures the *shortage* between the achieved level of skills and the target $\mathbf{t}$. Hence, when $\alpha = 100\%$, the student has fully achieved the desired job requirements. On the other hand, when $\alpha = 0\%$, the student does not possess any skills out of those required by the job.

Given the above-defined job affinity function, we solve the DPP for all the jobs in our dataset and compute the corresponding job affinity values. Notice that the capability of our approach of solving the DPP within seconds is fundamental for this kind of analysis, for which we must solve the DPP multiple times (as many as jobs). Figure 3 reports the histogram of job affinity values. Results show that for almost 10% of target jobs (11 out of 118 target jobs), the optimal degree paths achieve a job affinity greater than 90%. Furthermore, for 58% of the jobs, the computed degree paths achieve job affinities greater than 50%. In more detail, for 8 jobs, the job affinity for their optimal degree plans is 100%, i.e., students acquire all the skills required by the jobs when completing the recommended optimal degree plans.

Furthermore, we illustrate the evolution of skills learned along semesters in Fig. 4. We show how job affinity progressively increases over semesters. In more detail, by the end of the third semester (half of the degree plan), students acquire 40% of the required skills on average. Moreover, once the degree plan is completed, the average job affinity is 56% as shown in Fig. 5. Therefore, 29% of job affinity value is gained in the second part of a degree plan, where most of the elective units are completed. We argue that most of the skills not acquired can not be learned with any of the units of the degree curriculum. This is usual since some jobs require high-level skills that can only be acquired with additional specialized education or professional experience.

To better understand and illustrate the value of electives, in Fig. 5, we show in greater detail the job affinity values for the degree plans computed considering each job in our data-

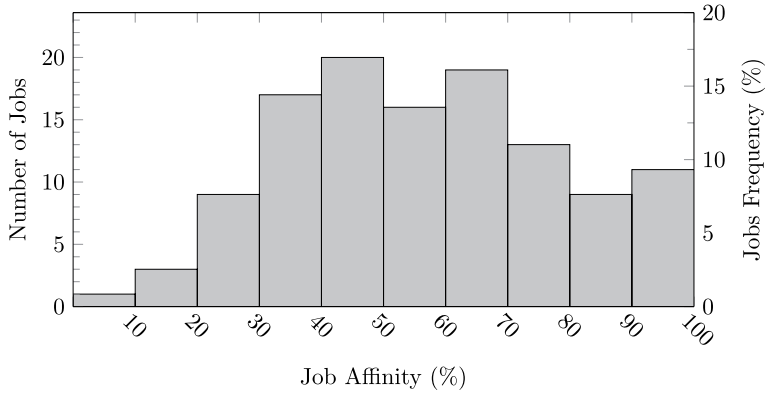

Fig. 3 Histogram of job affinities for all jobs. We report the distribution of the job affinity values in two frequency metrics: the absolute number of jobs (left axis), and the percentage of jobs (right axis)

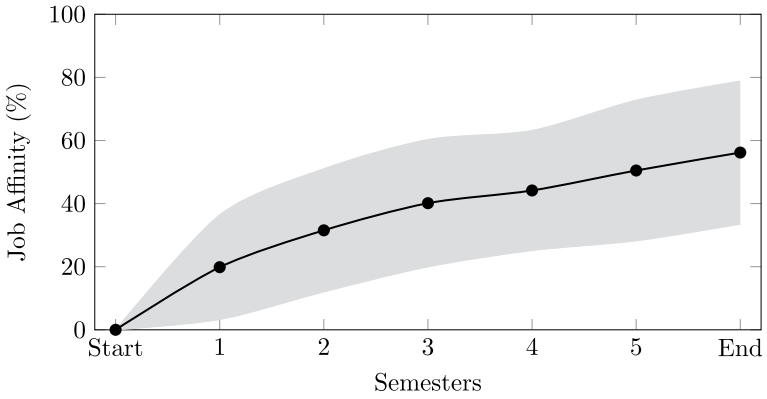

Fig. 4 Evolution of the job affinity during the degree plan. The shadowed region represents the standard deviation of the job affinity for the different target jobs

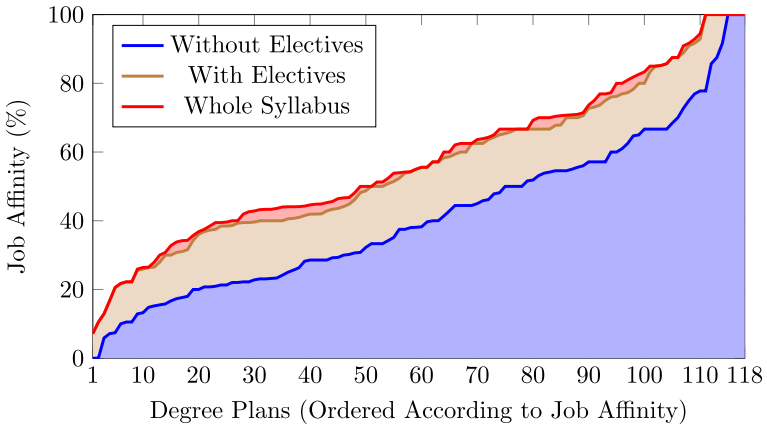

Fig. 5 Cactus plot showing the job affinity for the degree plan of each target job. We order degree plans from the lowest job affinity to the highest on the x-axis. We plot the job affinity considering degree plans without the elective units (in blue); whole degree plans (in brown); and all units from the degree syllabus, $\mathcal{U}$(in red)

set. Specifically, we plot the job affinity values considering (i) degree plans without elective units (considering only core and major units), (ii) whole degree plans (with electives), and (iii) all units from the syllabus (all $u \in \mathcal{U}$). Notice that elective units significantly increase the job affinity values of degree plans. In detail, the average job affinity without considering electives is 41.5%, while the average job affinity is 56.2%. Thus, elective units lead to a 26% relative increase in job affinity. Furthermore, the recommended degree plans have the highest job affinity values possible in most cases since coursing all available courses in the degree syllabus does not significantly increase job affinity. This indicates that most of the skills not acquired in the recommended degree plans can not be learned with any of the units of the degree curriculum. As mentioned above, this is usual since some jobs require

high-level skills that can only be acquired with additional specialized education or professional experience.[15]

We believe these analyses could be useful for students and curriculum designers. On the one hand, students can identify which jobs are more aligned with their degree, helping them better understand whether it is appropriate for the professional career they aim to pursue. Moreover, students can track their skill acquisition during their studies. On the other hand, curriculum designers can evaluate which portion of the job market is "covered" by the degree in terms of professional skills and, if deemed appropriate, adjust the offered units in a principled way.

5.5 Use case: multi-job target

Our third use case focuses on a scenario in which a student is uncertain about the specific job to pursue but has several potential candidate jobs in mind. Along these lines, our algorithm could benefit the student since it could recommend a *joint* degree path to cover the professional skills required by such potential jobs. Nonetheless, such a joint degree plan aiming at several target jobs at once could provide, for each individual job, a lower skill coverage compared to the degree paths specifically aimed at each job separately.

Thus, this experiment aims to quantify the skill coverage of joint degree plans *with respect to* individual degree plans. Hence, the larger the skill coverage of a joint degree plan *relative* to such individual plans, the better, because it is closer to the best possible individual plans.

To quantify the skill coverage of joint degree plans, we first define an auxiliary metric to compute the skill coverage of each job in a candidate job set given a joint degree plan. Recall that a degree plan (as well as a joint degree plan) is characterized by a set of decision variables. Along these lines, we use the vector $\mathbf{z}$ encoding the computed plan to determine the skill coverage of a joint degree plan. In detail, $z_s \in \mathbb{Z}^-$ encodes the level shortage of skill s of a joint degree plan, i.e., the difference between the target skill level t_s and the skill level that would be achieved by following the plan.

Definition 5.2 (*Single Job Affinity (SJA)*) Given a candidate job $j = \langle S_j, {\sigma'}_j \rangle$, a candidate job set $\mathcal{J}$ such that $j \in \mathcal{J}$, a target vector of the candidate job set required skills $\mathbf{t}$, and a joint degree plan for the jobs in $\mathcal{J}$ whose skill level shortage is encoded by the decision variable vector $\mathbf{z}$, we define the Single Job Affinity of job j for the joint degree plan as:

$$\alpha_{j,\mathcal{J}}(\mathbf{t}, \mathbf{z}) = 100 \cdot \frac{\sum_{s \in S_j} \min({\sigma'}_j(s), t_s + z_s)}{\sum_{s \in S_j} {\sigma'}_j(s)}, \tag{61}$$

where $t_s + z_s$ is the achieved level of skill s after completing the joint degree plan.
After that, we define the notion of *Relative* Job Affinity to compute the skill coverage of a job in a joint degree plan with respect to an individual degree plan.

[15] For example, the role of "*Software Engineer*" requires level 5 of skill "*Systems Integration*" (see Table 2). According to the SFIA, such a high level indicates that "*the employee is capable of providing authoritative guidance in such skill and is accountable for delivering significant work outcomes*". Therefore, such a skill level can only be acquired after years of professional experience.

Definition 5.3 (*Relative Job Affinity (RJA)*) Given a candidate job $j \in \mathcal{J}$, the target vector $\mathbf{t}$ of the skills required by the candidate jobs in $\mathcal{J}$, the skill level shortage $\mathbf{z}$ of a joint degree plan for $\mathcal{J}$, the target vector $\hat{\mathbf{t}}^j$ of the skills required by job j, and the skill level shortage $\hat{\mathbf{z}}^j$ of an individual job degree plan for job j, we define the Relative Job Affinity of the joint degree plan with respect to the individual degree plan as

$$RJA(j, \mathbf{t}, \mathbf{z}, \hat{\mathbf{t}}^j, \hat{\mathbf{z}}^j) = 100 \cdot \frac{\alpha_{j,\mathcal{J}}(\mathbf{t}, \mathbf{z})}{\alpha_j(\hat{\mathbf{t}}^j, \hat{\mathbf{z}}^j)}, \tag{62}$$

where $\alpha_{j,\mathcal{J}}(\mathbf{t}, \mathbf{z})$ is the SJA of job j and $\alpha_j(\hat{\mathbf{t}}^j, \hat{\mathbf{z}}^j)$ is the job affinity of job j individual job degree plan.[16]

Given a set of candidate jobs and a joint degree plan for it, we quantify its skill coverage in three steps. First, we compute the degree plan for each job in the candidate job set. Second, we compute the RJA for each job in the candidate job set. Finally, we need a global measure to quantify the skill coverage of a joint degree plan with respect to all the individual plans. Thus, we aggregate the candidate jobs' RJAs to obtain the skill coverage of the joint degree plan. To this end, we define the Average RJA as follows.

Definition 5.4 (*Average Relative Job Affinity (ARJA)*) Given a candidate job set $\mathcal{J}$, the target vector $\mathbf{t}$ of the skills required by the candidate jobs in $\mathcal{J}$, the skill level shortage $\mathbf{z}$ of a joint degree plan for $\mathcal{J}$, the set of target vectors (denoted as $\hat{T}$) of the skills required by each job $j \in \mathcal{J}$, and the set of skill level shortages (denoted as $\hat{Z}$) of an individual job degree plan for each job $j \in \mathcal{J}$, we define the Average Relative Job Affinity of the joint degree plan as

$$ARJA(\mathcal{J}, \mathbf{t}, \mathbf{z}, \hat{T}, \hat{Z}) = \frac{1}{|\mathcal{J}|} \cdot \sum_{j \in \mathcal{J}} RJA(j, \mathbf{t}, \mathbf{z}, \hat{\mathbf{t}}^j, \hat{\mathbf{z}}^j) = \frac{100}{|\mathcal{J}|} \cdot \sum_{j \in \mathcal{J}} \frac{\alpha_{j,\mathcal{J}}(\mathbf{t}, \mathbf{z})}{\alpha_j(\hat{\mathbf{t}}^j, \hat{\mathbf{z}}^j)}. \tag{63}$$

Once we have defined our metric for skill coverage of a joint degree plan, we describe our experimental setting. We aim to quantify how well a joint degree plan covers a student's requirements as their uncertainty increases, assuming that if students are more uncertain, they will specify more candidate jobs. Thus, we consider up to $|\mathcal{J}| = 10$ for cases with maximum uncertainty, and we refer to cases with minimum uncertainty when $|\mathcal{J}| = 2$. To compute the ARJA for different sizes of candidate job sets, we solve the DPP for 4500 unique candidate job sets of size $|\mathcal{J}| = 2, 3, \ldots, 10$, i.e., 500 combinations of jobs per set size. Notice that computing the degree plan for all the possible job combinations is impossible because of the so-called "Curse of Dimensionality". For this experimental setting, the total number of combinations is $\sum_{k=1}^{10} \binom{118}{k} = 1.07 \times 10^{14}$. Therefore, given the impossibility of computing the degree plan due to the unmanageable number of possible combinations of jobs, the capability of our approach to compute degree plans in real time is crucial.

Figure 6 shows how the skill coverage decreases as the student's uncertainty increases. In detail, we notice a decrease in the ARJA as the number of target jobs increases, i.e., as the number of target jobs (i.e., the student's uncertainty) increases. We do not plot the statistical error since it is negligible ($\approx 0.2\%$). Overall, even in the case with maximum uncertainty,

[16] Notice that RJA $\in [0, 100]$.

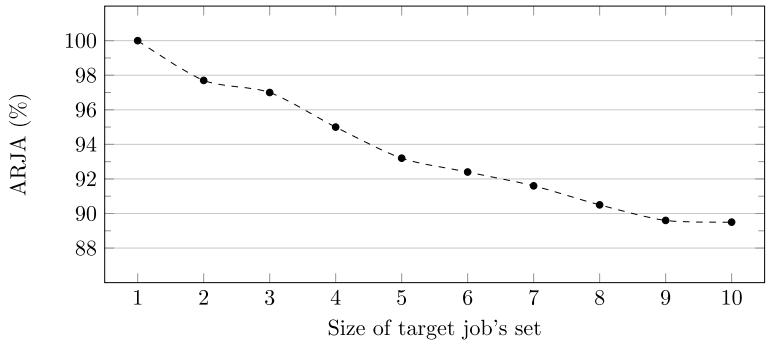

Fig. 6 Average RJA along the size of the set of target jobs

i.e., the case where candidate job sets contain 10 jobs, the ARJA is high, i.e., 10.5% decrease with respect to no uncertainty about their target job (i.e., $|\mathcal{J}| = 1$). This shows that the joint degree plans that DPP computes are valuable recommendations for a student despite uncertainty. Specifically, for instances considering up to 4 target jobs, the average ARJA is 95%. Moreover, the average ARJA decreases below the 90% exclusively for candidate job sets with a big number of jobs, i.e., $|\mathcal{J}| \geq 9$. Therefore, when a student is uncertain about their desired career path, our approach can compute degree paths for multiple target jobs with a minimum decrease in skill coverage with respect to the case where students know their desired job. This is due to the fact that, in our dataset, different jobs may require common skills, as well as to the fact that units may provide skills that are required by multiple jobs.

5.6 Use case: mid-degree planning

In our fourth use case, we consider that a student has already started their bachelor's degree and they have already completed some units. Henceforth, we employ our approach to plan a path for the remaining part of their degree. We consider 12 "reference" degree paths based on the recommended ones in the bachelor's syllabus. Then, for each reference degree path, we consider that there is a student who partially completed such degree path (i.e., 1 semester, 2 semesters, etc.). We then complete each partial path with our approach and calculate the job affinity increase by comparing the computed path with the reference one. We analyzed the 20 jobs with the highest job affinity for each "reference" degree path.

Figure 7 shows the increase of the job affinity's mean and standard deviation across all the above-mentioned experiments. Results indicate that our approach can indeed provide an improvement with respect to the reference path indicated by the bachelor's syllabus. We also observe that the maximum increase is obtained when our approach is employed early in the degree path. This result is expected since, in this case, the optimization is less constrained, thus it can reach a higher job affinity. Moreover, we observe a sharp drop in the job affinity increase when students complete half of their degree. As expected, choosing a target during the first half of the degree is crucial for skill achievement.

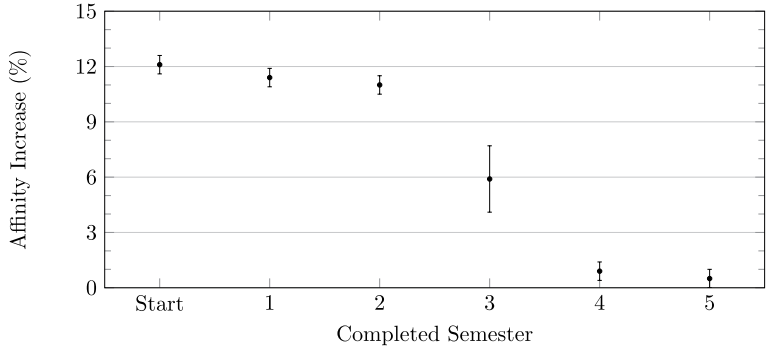

Fig. 7 Evolution of job affinity increase along semesters

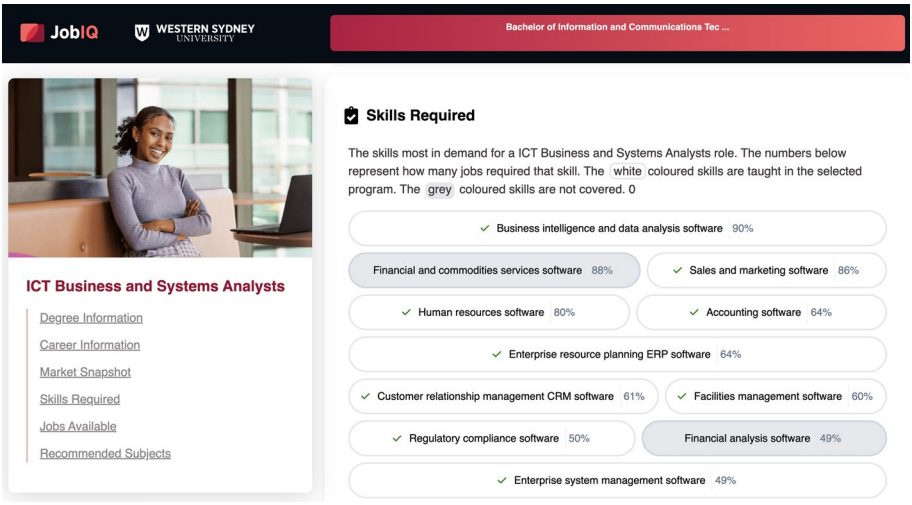

Fig. 8 Screenshot of JobIQ showing the skills required by the *ICT Business and Systems Analysts* job and which skills are acquired after completing the Bachelor's Degree in ICT

6 Real-world applicability of our approach

6.1 Current deployment

To showcase the applicability of our approach to real-world university systems, we developed JobIQ[17] based on the approach proposed in this paper. Such a system recommends Western Sydney University degrees based on students' career preferences. The website informs students which skills they acquire during their study (as shown in Fig. 8) and which job markets they can target after graduation. Furthermore, it shows them current job offers from the countries where such a university is present, i.e., Australia, New Zealand, and Indonesia, including financial prospects or demanding employers. This approach provides

[17] https://www.jobiq.com.au. Last Consulted: February 2025.

an idea of what career prospects there are for our graduates. Therefore, JobIQ has become one of the main recruitment tools for our hiring strategy.

6.2 Extending our DPP model to other curricula

Even though our formulation discussed in Sect. 3 involves some aspects specific to the Bachelor's Degree in ICT from Western Sydney University (such as the major requirements in Sect. 3.1), we will now discuss how it can be adapted to other common scenarios by means of additional linear constraints.

As a first example, it is common for students to be forced to enroll in a minimum number of credits each semester. Furthermore, to avoid students suffering from an excessive workload, degree curricula usually specify a maximum number of credits (Castro and Manzano, 2001). In such cases, we would replace constraint (16) by the set of constraints $\mathbf{d_{min}} \leq D\mathbf{x} \leq \mathbf{d_{max}}$, where $\mathbf{d_{min}}, \mathbf{d_{max}} \in \mathbb{N}^{|\mathcal{L}|}$ are vectors whose components are the minimum and maximum number of credits allowed to enroll per semester. Furthermore, it is common to require that students complete a minimum number of credits to graduate instead of allowing them to just enroll in a given number of credits. In such case, we would modify constraint (14) such that $\mathbf{c}^T\mathbf{x} \geq c_T$.

Another common scenario is when academic curricula require to complete a given number of "*basic-level*" units to access "*higher-level*" units. In such a case, we formalize such a restriction with a new type of prerequisite constraint that enforces having completed a certain number of basic-level units before enrolling in a higher-level course. We define a subset of units containing the basic-level units $\mathcal{B} \subset \mathcal{U}$ and a second subset of units containing the higher-level units $\mathcal{H} \subset \mathcal{U}$. Then, we impose a set of linear constraints such that

$$\sum_{u \in \mathcal{B}} \sum_{i=1}^{l-1} x_{u,i} \geq n_h \cdot x_{h,l}, \quad \text{for all } h \in \mathcal{H}, l \in \mathcal{L},$$

where n_h is the number of basic-level units to be completed to enroll in unit h.

Moreover, some institutions require students to complete special courses (or a certain number) to finish their bachelor's degree. For example, in the US, some degrees require the completion of "*Liberal Education*" courses (Roth, 2014). In such a case, we define a special subset of units $\mathcal{I} \subset \mathcal{U}$. Then, we impose a set of linear constraints such that

$$\sum_{u \in \mathcal{I}} \sum_{l \in \mathcal{L}} x_{u,l} \geq n_{\mathcal{I}}$$

where $n_{\mathcal{I}}$ is the number of special units to be completed to finish a bachelor's degree.

7 Related work

Historically, Operational Research (OR) has proven to be a very useful tool for the solution of several challenges related to education (Johnes, 2015), especially to help governments in the allocation and planning of the resources aimed at the education of the citizens. OR has

also been used to solve scheduling problems such as *High School Timetabling*, *University Course Timetabling* and *Audit Scheduling* (Brucker and Knust, 2000), by formalizing them as an RCPSP (Brucker et al., 1999; Hartmann and Briskorn, 2022). RCPSP is a well-known NP-hard problem (Blazewicz et al., 1983) that consists of activities that must be scheduled within a planning horizon to meet precedence and resource constraints and to minimize the project's duration, i.e., the *makespan*.

While there exist similarities between DPP and RCPSP,[18] the two are fundamentally different problems, preventing us from employing any of the algorithms proposed in the above-cited literature. Firstly, DPP and RCPSP have different optimization objectives: the goal of DPP is obtaining the set of skills that best matches the student's job preferences, whereas the main goal of the RCPSP is to minimize the makespan. In other words, in the DPP formalization, the student or the university set the *available* time they count on completing the degree, but minimizing such a time is not the optimization objective, in contrast with RCPSP. Secondly, in the DPP, not all units must be scheduled since the goal is selecting the ones that best fit with the skill set required by the student's job preferences, in contrast with the RCPSP for which all activities must be scheduled. Finally, DPP's constraints are specific to the university-course context and cannot be accommodated within the RCPSP framework.

Another closely related problem is the *Balanced Academic Curriculum Problem* (BACP), which is the planning problem aiming at assigning a set of courses from an academic curriculum to a set of teaching periods (e.g. semesters, trimesters, etc.). However, unlike the DPP —whose goal is to maximize the affinity between the skills acquired by the student and the requirements by the desired job—, the BACP's goal is to balance the workload of a plan among teaching periods while meeting the constraints of the academic curriculum.

The BACP has received attention in the last decades since it is a key task universities tackle every academic year. Castro and Manzano (2001), Castro et al. (2007) initially propose both *Integer Programming* (IP) and *Constraint Programming* (CP) models that minimize the maximum workload per teaching period, hence balancing the academic workload of the academic curriculum. Hnich et al. (2002, 2004) combine IP and CP techniques to build a hybrid approach that efficiently solves the BACP. Lambert et al. (2005, 2006) propose a hybrid algorithm that integrates a genetic algorithm with CP techniques to outperform state-of-the-art optimization solvers. Moreover, Monette et al. (2007) define new balance criteria as objective functions of the BACP. Then, they solve the BACP employing a set of local search strategies and constraint programming techniques. Di Gaspero and Schaerf (2008), Chiarandini et al. (2012), and Ceschia et al. (2014) propose a generalization of the BACP. They consider that not all courses must be completed in a curriculum since students can choose which courses to take. In addition, they allow professors to set their preferences on which teaching periods they give their courses. Ünal and Uysal (2014) propose to formalize the BACP as a *Generalised Quadratic Assignment Problem*. They model such a problem with the aim of planning related courses as closely as possible in the degree plan. Similarly, Slim et al. (2015) propose to schedule the most *crucial* courses as early as possible in the plan. More recently, Christou et al. (2024) propose an *Integer Linear Programming* (ILP) formalization to plan degrees for the American College of Greece. However, their formalization has different optimization goals with respect to the generalization

[18] University units can be thought as activities with precedence constraints. Moreover, the sequence of semesters is analogous to the planning horizon, and credits can be considered as resources.

of the BACP (Ceschia et al., 2014). Their approach to compute degree plans has three goals: (i) minimize the total duration of a bachelor's degree, (ii) balance the courses' difficulty in each semester, and (iii) maximize the expected average score during the degree. Finally, Heileman and Zhang (2024) propose a related problem, the so-called *Optimal Learning Outcome Assignment* (OLOA) problem. Such a problem consists of assigning learning outcomes, i.e., learning concepts that students must acquire, into the courses of an academic curriculum. Such a problem aims to reduce the difficulty of the curriculum by optimizing the arrangement of learning outcomes.

Even though the BACP is a problem closely related to the DPP, our goal is different from all the above-mentioned approaches. By addressing DPP, our work focuses on maximizing the required skills for a student's desired job. Moreover, we introduce new realistic constraints that have not been modeled in the literature so far, such as the introduction of majors in a degree curriculum and their related restrictions to complete them.

Alongside BACP approaches, commercial tools capable of planning degree paths have also been proposed in the last few years. The most notable example is *Prepler*,[19] a web application to plan degree paths for multiple universities of the United States according to a desired study load, i.e., hours that students are willing to devote in each semester. Nonetheless, the objective of Prepler is minimizing the amount of "off track" time experienced by the student to reduce the number of tuition fees, which is fundamentally different from the DPP.

Researchers have also developed methods or applications as part of *Career Planning* (CaP) intending to guide future job positions given the individual's job experience (Ghosh et al., 2020; Liu et al., 2016). Despite the abundance of works focusing on CaP, to the best of our knowledge, the DPP is a novel problem that has not been proposed before in the above-discussed literature. One major difference is that instead of predicting future job positions of an individual, the DPP's objective is to plan the optimal degree path to acquire the required skills for a desired future job.

Finally, the problem of recommending university courses or subjects to students has been addressed by EdRecSys, which plays a crucial role in the context of *Technology Enhanced Learning* (TEL) (Drachsler et al., 2015). Several EdRecSys have been developed, as discussed in several surveys (Drachsler et al., 2015; Dascalu et al., 2016). In contrast with the above-mentioned literature, whose main goal is providing recommendations to students about courses or subjects based on other students' choices and preferences without a specific optimization goal, our approach computes the degree path that maximizes the acquisition of skills according to the desired job.

8 Conclusions and future work

As motivated above, developing decision-support tools for guiding students throughout their educational pathways toward the acquisition of the skills required by their preferred job(s) is key. In this paper, our goal was to contribute to the development of such decision-support tools.

First, we formalized the problem of computing the degree path for a student to be employable in their desired jobs as an optimization problem. Solving such an optimization problem

[19] Online at https://www.prepler.com/degree-plans (last consulted: January 2024).

yields the degree paths that maximize the "coverage" of the professional skills required by a student's desired job(s). Second, because of the non-linearity of the objective function in that optimization problem, we introduced a non-trivial linearization of the problem so that we can solve it with standard integer optimization tools. We formally prove that our transformation produces an equivalent optimization problem.

Throughout our experimental evaluation with real-world data from the Bachelor of ICT of the University of Western Sydney, we showed that our approach can be a valuable decision-support tool for (i) students planning their academic pathways and (ii) curriculum designers assessing the matching between a university's degrees and the job market. Despite showcasing it for an actual real-world example, our model is straightforward to adapt to other university curricula without changing the proposed solution method.

In spite of the above-mentioned improvements to the state of the art, our work can be improved according to the following open research lines. On the one hand, we plan to explain the degree plans provided by our approach since presenting explanations is key to increasing users' trust and satisfaction within an educational planning system, as suggested by Barria-Pineda (2020). On the other hand, we plan to *personalize* degree paths by also taking into account a student's preferences and their academic performance in already-completed units. Notice that, in contrast with existing related work (Drachsler et al., 2015; Shao et al., 2021) in the educational recommender systems literature, which provides personalized recommendations on courses without planning them, in future work, we will combine optimization and learning to provide individualized degree plans by predicting the *expected performance* of students. To do that, we plan to employ: (i) actual data on the student's performance following the recommended degree plan; and (ii) observed performance of other students who completed their degrees. Predicting such expected performance is important as an indicator for students of the difficulty of the degree plan they are about to follow. As students progress, the expected performance can be refined, and our approach can learn to anticipate problems and recommend variations of a degree plan.

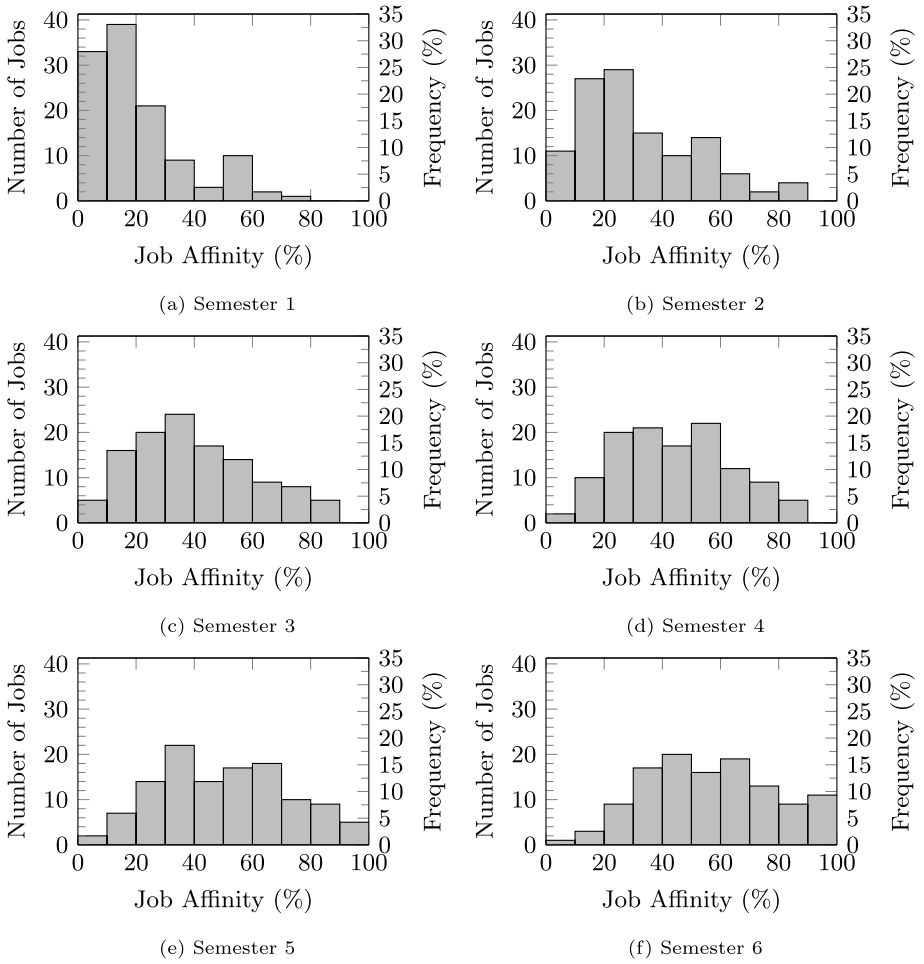

Fig. 9 Evolution of the distribution of job affinities for all jobs after completing each semester

Appendix: Job affinity evolution along semesters

Figure 9 shows the evolution of the distribution of job affinities for all jobs after each semester. Notice that most jobs have low job affinity values after completing the first semesters. This indicates that students have not acquired many of the skills required for those jobs, which is usual since, in the first semesters, students complete core units (which are the ones that have no prerequisites). However, job affinity increases over semesters as more jobs have higher job affinity values. Figure 9f shows the distribution of the job affinity after completing semester 6, i.e., after finishing the degree. Thus, note that Fig. 9f is the same as Fig. 3, whose results are discussed in Sect. 5.4.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.
The authors were supported by the research projects ACISUD (PID2022-136787NB-I00) and Yoma Operational Research (OPE02570).

Declarations

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

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Alizadeh, F., & Goldfarb, D. (2001). Second-order cone programming. *Mathematical Programming,95*, 3–51.

Australian Digital Transformation Agency. (2019). Aps digital career pathways dataset. https://data.gov.au/data/dataset/aps-digital-career-pathways. Online.

Barria-Pineda, J. (2020). Exploring the need for transparency in educational recommender systems. In *ACM conference on user modeling, adaptation and personalization* (pp. 376–379).

Blazewicz, J., Lenstra, J. K., & Kan, R. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics,5*(1), 11–24.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

British Computer Society. (2021). Skills framework for the information age. https://sfia-online.or. Online.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research,112*(1), 3–41.

Brucker, P., & Knust, S. (2000). Resource-constrained project scheduling and timetabling. In *International conference on the practice and theory of automated timetabling* (pp. 277–293). Springer.

Castro, C., Crawford, B., & Monfroy, E. (2007). A quantitative approach for the design of academic curricula. In *Human interface and the management of information. Interacting in information environments: symposium on human interface 2007, held as part of HCI international 2007, Beijing, China, July 22–27, 2007, proceedings, part II* (pp. 279–288). Springer.

Castro, C., & Manzano, S. (2001). Variable and value ordering when solving balanced academic curriculum problems. In *6th workshop of the ERCIM WG on constraints*.

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2014). The generalized balanced academic curriculum problem with heterogeneous classes. *Annals of Operations Research,218*, 147–163.

Chakrabarty, D., & Swamy, C. (2019). Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing* (pp. 126–137).

Chiarandini, M., Di Gaspero, L., Gualandi, S., & Schaerf, A. (2012). The balanced academic curriculum problem revisited. *Journal of Heuristics,18*, 119–148.

Christou, I. T., Vagianou, E., & Vardoulias, G. (2024). Planning courses for student success at the American College of Greece. *INFORMS Journal on Applied Analytics,54*, 365–379.

Dascalu, M.-I., Bodea, C.-N., Mihailescu, M. N., Tanase, E. A., & Ordoñez de Pablos, P. (2016). Educational recommender systems and their application in lifelong learning. *Behaviour & Information Technology,35*(4), 290–297.

Di Gaspero, L., & Schaerf, A. (2008). Hybrid local search techniques for the generalized balanced academic curriculum problem. In *Hybrid metaheuristics: 5th international workshop, HM 2008, Málaga, Spain, October 8–9, 2008. Proceedings 5* (pp. 146–157). Springer.

Drachsler, H., Verbert, K., Santos, O., & Manouselis, N. (2015). Panorama of recommender systems to support learning. In *Recommender systems handbook* (pp. 421–451). Springer.

Etminaniesfahani, A., Gu, H., Naeni, L. M., & Salehipour, A. (2023). An efficient relax-and-solve method for the multi-mode resource constrained project scheduling problem. *Annals of Operations Research* 1–28.

Gašević, D., Dawson, S., & Siemens, G. (2015). Let's not forget: Learning analytics are about learning. *TechTrends,59*(1), 64–71.

Ghosh, A., Woolf, B., Zilberstein, S., & Lan, A. (2020). Skill-based career path modeling and recommendation. In *International conference on big data (big data)* (pp. 1156–1165).

Griva, I., Nash, S., & Sofer, A. (2009). *Linear and nonlinear optimization* (Vol. 108). SIAM.

Hartmann, S., & Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research,297*(1), 1–14.

Heileman, G. L., & Zhang, Y. (2024). Minimizing curricular complexity through backwards design. In *2024 ASEE annual conference & exposition*.

Hnich, B., Kiziltan, Z., Miguel, I., & Walsh, T. (2004). Hybrid modelling for robust solving. *Annals of Operations Research,130*, 19–39.

Hnich, B., Kiziltan, Z., & Walsh, T. (2002). Modelling a balanced academic curriculum problem. In *Proceedings of CP-AI-OR-2002* (pp. 121–131). sn.

Johnes, J. (2015). Operational research in education. *European Journal of Operational Research,243*(3), 683–696.

Karnebogen, M., & Zimmermann, J. (2024). Generation schemes for the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints. *Annals of Operations Research,338*, 173–192.

Lambert, T., Castro, C., Monfroy, E., Riff, M. C., & Saubion, F. (2005). Hybridization of genetic algorithms and constraint propagation for the bacp. In *Logic programming: 21st international conference, ICLP 2005, Sitges, Spain, October 2–5, 2005. Proceedings 21* (pp. 421–423). Springer.

Lambert, T., Castro, C., Monfroy, E., & Saubion, F. (2006). Solving the balanced academic curriculum problem with an hybridization of genetic algorithm and constraint propagation. In *Artificial intelligence and soft computing—ICAISC 2006: 8th international conference, Zakopane, Poland, June 25–29, 2006. Proceedings 8* (pp. 410–419). Springer.

Liu, Y., Zhang, L., Nie, L., Yan, Y., & Rosenblum, D. S. (2016). Fortune teller: Predicting your career path. In *AAAI conference on artificial intelligence* (pp. 201–207).

Monette, J.-N., Schaus, P., Zampelli, S., Deville, Y., Dupont, P., et al. (2007). A cp approach to the balanced academic curriculum problem. In *Seventh international workshop on symmetry and constraint satisfaction problems* (Vol. 7).

Pass-Lanneau, A., Bendotti, P., & Brunod-Indrigo, L. (2023). Exact and heuristic methods for anchor-robust and adjustable-robust rcpsp. *Annals of Operations Research* 1–34.

Roth, M. S. (2014). *Beyond the university: Why liberal education matters*. Yale University Press.

Salas-Molina, F., Bistaffa, F., & Rodríguez-Aguilar, J. A. (2023). A general approach for computing a consensus in group decision making that integrates multiple ethical principles. *Socio-Economic Planning Sciences,89*, Article 101694.

Shao, E., Guo, S., & Pardos, Z. A. (2021). Degree planning with plan-bert: Multi-semester recommendation using future courses of interest. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 14920–14929).

Siemens, G., & Baker, R. S. d. (2012). Learning analytics and educational data mining: Towards communication and collaboration. In *International conference on learning analytics and knowledge* (pp. 252–254).

Slim, A., Heileman, G. L., Lopez, E., Al Yusuf, H., & Abdallah, C. T. (2015). Crucial based curriculum balancing: A new model for curriculum balancing. In *2015 10th international conference on computer science & education (ICCSE)* (pp. 243–248). IEEE.

Trescak, T., Lera-Leri, R., Bistaffa, F., and Rodriguez-Aguilar, J. A. (2022). Agent-assisted life-long education and learning. In *International conference on autonomous agents and multi-agent systems* (pp. 1819–1823).

Tsai, Y.-S., & Gasevic, D. (2017). Learning analytics in higher education—challenges and policies: a review of eight learning analytics policies. In *International conference on learning analytics and knowledge* (pp. 233–242).

Ünal, Y. Z., & Uysal, Ö. (2014). A new mixed integer programming model for curriculum balancing: Application to a Turkish University. *European Journal of Operational Research,238*(1), 339–347.

Wolsey, L. A. (2020). *Integer programming*. Wiley.