Discrete Optimization 29 (2018) 174–195

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Pareto optimal matchings of students to courses in the presence of prerequisites[☆]

Katarína Cechlárová[a], Bettina Klaus[b], David F. Manlove[c,*]

[a] *Institute of Mathematics, Faculty of Science, P.J. Šafárik University, Jesenná 5, 040 01 Košice, Slovakia*
[b] *Faculty of Business and Economics, University of Lausanne, Internef 538, CH-1015 Lausanne, Switzerland*
[c] *School of Computing Science, Sir Alwyn Williams Building, University of Glasgow, Glasgow, G12 8QQ, UK*

ARTICLE INFO

Article history:
Received 8 July 2017
Received in revised form 13 February 2018
Accepted 22 April 2018
Available online 27 July 2018

Keywords:
Course allocation
Prerequisites
Pareto optimality
Sequential mechanism
NP-hardness

ABSTRACT

We consider the problem of allocating applicants to courses, where each applicant has a subset of acceptable courses that she ranks in strict order of preference. Each applicant and course has a *capacity*, indicating the maximum number of courses and applicants they can be assigned to, respectively. We thus essentially have a many-to-many bipartite matching problem with one-sided preferences, which has applications to the assignment of students to optional courses at a university.

We consider additive preferences and lexicographic preferences as two means of extending preferences over individual courses to preferences over bundles of courses. We additionally focus on the case that courses have prerequisite constraints: we will mainly treat these constraints as compulsory, but we also allow alternative prerequisites. We further study the case where courses may be corequisites.

For these extensions to the basic problem, we present the following algorithmic results, which are mainly concerned with the computation of Pareto optimal matchings (POMs). Firstly, we consider compulsory prerequisites. For additive preferences, we show that the problem of finding a POM is NP-hard. On the other hand, in the case of lexicographic preferences we give a polynomial-time algorithm for finding a POM, based on the well-known sequential mechanism. However we show that the problem of deciding whether a given matching is Pareto optimal is co-NP-complete. We further prove that finding a maximum cardinality (Pareto optimal) matching is NP-hard. Under alternative prerequisites, we show that finding a POM is NP-hard for either additive or lexicographic preferences. Finally we consider corequisites. We prove that, as in the case of compulsory prerequisites, finding a POM is NP-hard for additive preferences, though solvable in polynomial

☆ Supported by grant APVV-15-0091 from the Slovak Research and Development agency (Cechlárová), by grant 100018_150086 from the Swiss National Science Foundation SNFS (Klaus), by grants EP/P028306/1 and EP/K010042/1 from the Engineering and Physical Sciences Research Council (Manlove), and by a Short-Term Scientific Mission from COST Action IC1205 on Computational Social Choice (Manlove). Part of this work was carried out whilst the authors visited Corvinus University of Budapest, and whilst the third author visited P.J. Šafárik University. This research was initiated during Dagstuhl Seminar 15241 on Computational Social Choice: Theory and Applications [1]. We would like to thank the following Dagstuhl participants who were involved in initial discussions regarding the results in this paper: Haris Aziz, Péter Biró, Jiehua Chen and Nicholas Mattei.

* Corresponding author.

E-mail addresses: katarina.cechlarova@upjs.sk (K. Cechlárová), Bettina.Klaus@unil.ch (B. Klaus), david.manlove@glasgow.ac.uk (D.F. Manlove).

https://doi.org/10.1016/j.disopt.2018.04.004

time for lexicographic preferences. In the latter case, the problem of finding a maximum cardinality POM is NP-hard and very difficult to approximate.

1. Introduction

Problems involving the allocation of indivisible goods to agents have gained a lot of attention in the literature, since they model many real scenarios, including the allocation of pupils to study places [2], workers to positions [3], researchers to projects [4], tenants to houses [5] and students to courses [6], etc. We assume that agents on one side of the market (pupils, workers, researchers, tenants, students) have preferences over objects on the other side of the market (study places, positions, projects, courses, etc.) but not vice versa. In such a setting where the choices of agents are in general conflicting, economists regard Pareto optimality (or Pareto efficiency) as a basic, fundamental criterion to be satisfied by an allocation. This concept guarantees that no agent can be made better off without another agent becoming worse off. A popular and very intuitive approach to finding Pareto optimal matchings is represented by the class of *sequential allocation mechanisms* [7–10].

In the one-to-one case (each agent receives at most one object, and each object can be assigned to at most one agent) this mechanism has been given several different names in the literature, including serial dictatorship [5,11], queue allocation [12], Greedy-POM [13] and sequential mechanism [9,10], etc. Several authors independently proved that a matching is Pareto optimal if and only if it can be obtained by the serial dictatorship mechanism (Svensson in 1994 [12], Abdulkadiroğlu and Sönmez in 1998 [5], Abraham et al. in 2004 [13], and Brams and King in 2005 [8]).

In general many-to-many matching problems (agents can receive more than one object, and objects can be assigned to more than one agent), the sequential allocation mechanism works as follows: a central authority decides on an ordering of agents (often called a *policy*) that can contain multiple copies of an agent (up to her capacity). According to the selected policy, an agent who has her turn chooses her most preferred object among those that still have a free slot. This approach was used in [9,10], where the properties of the obtained allocation with respect to the chosen policy and strategic issues are studied.

The serial dictatorship mechanism is a special case of the sequential allocation mechanism where the policy contains each agent exactly once, and when agent a is dealt with, she chooses her entire most-preferred bundle of objects. The difficulty with serial dictatorship is that it can output a matching that is highly unfair. For example, it is easy to see that if there are two agents, each of whom finds acceptable all objects and has capacity equal to the number of objects, and each object has capacity 1, then the serial dictatorship mechanism will assign all objects to the first agent specified by the policy and no object to the other agent.

In this paper we shall concentrate on one real-life application of this allocation problem that arises in education, and so our terminology will involve *applicants* (students) for the agents and *courses* for the objects. In most universities students have some freedom in their choice of courses, but at the same time they are bound by the rules of the particular university. A detailed description of the rules of the allocation process and the analysis of the behaviour of students at Harvard Business School, based on real data, is provided by Budish and Cantillon [6]. They assume that students have a linear ordering of individual courses and their preferences over bundles of courses are responsive to these orderings. The emphasis in [6] was on strategic questions. The empirical results confirmed the theoretical findings that, loosely speaking, dictatorships (where students choose one at a time their entire most preferred available bundle) are the only mechanisms that are strategy-proof and ex-post Pareto optimal.

Another field experiment in course allocation is described by Diebold et al. [14]. The authors compared the properties of allocations obtained by the sequential allocation mechanism where the policy is determined by the arrival time of students (i.e., first-come first-served) and by two modifications of the Gale–Shapley student-optimal mechanism, i.e., they assumed that courses may also have preferences or priorities over students. Moreover, they only considered the case when each student can be assigned to at most one course.

In reality, a student can attend more courses, but not all possible bundles are feasible for her. Cechlárová et al. [15] considered explicitly-defined notions of feasibility for bundles of courses. For these feasibility concepts, a given bundle can be checked for feasibility for a given applicant in time polynomial in the number of courses. Such an algorithm may check for example if no two courses in the bundle are scheduled at the same time, or if the student has enough budget to pay the fees for all the courses in the bundle, etc. Cechlárová et al. [15] found that a sufficient condition for a general sequential allocation mechanism to output a Pareto optimal matching is that feasible bundles of courses form families that are closed with respect to subsets, and preferences of students over bundles are lexicographic. They also showed that under these assumptions a converse result holds, i.e., each Pareto optimal matching can be obtained by the sequential allocation mechanism if a suitable policy is chosen.

Prerequisites and corequisites

In this paper we deal with prerequisite and corequisite constraints. Prerequisite constraints model the situation where a student may be allowed to subscribe to a course c only if she subscribes to a set C' of other course(s). The courses in C' are usually called *prerequisite courses*, or *prerequisites*, for c. For example, at a School of Mathematics, an Optimal Control Theory course may have as its prerequisites a course on Differential Equations as well as a course in Linear Algebra; a prerequisite for a Differential Equations course could be a Calculus course, etc. In realistic situations, prerequisite constraints are *acyclic*, as they usually require that prerequisite courses for course c should be completed before the student starts course c itself. On the other hand, corequisite constraints model the situation where a student takes course c_1 if and only if she takes course c_2. These courses are referred to as *corequisite courses*, or *corequisites*. For example, a corequisite constraint may act on two courses: one that is a theoretical programming course and the other that is a series of corresponding programming lab sessions. We introduce a general model that includes acyclic prerequisites as well as corequisites.

We allow for the possibility that different students may have different prerequisite constraints. For example, for a doctoral study program in Economics, an economics graduate may have as a prerequisite a mathematical course and, on the other hand, a mathematics graduate may have as a prerequisite a course on microeconomics, etc. In the case of corequisites (symmetric prerequisites) we shall pay special attention to the situation when corequisite courses are identical for all applicants, as this case usually occurs in reality.

Finally, the third model considers *alternative prerequisites*. Here it is assumed that certain courses require that a student subscribes to at least one of a set of other courses. For example, a course in mathematical modelling may require that a student attends one of a range of courses that deal with a specific mathematical modelling software package, such as Maple, MATLAB or Mathematica.

As we assume that applicants express their preferences only over individual courses, a suitable extension of these preferences to preferences over bundles of courses has to be defined. Among the most popular extensions are *responsive* preferences [16]. That is, an applicant has responsive preferences over bundles of courses if bundle C' is preferred to bundle C'' whenever C' is obtained from C'' by replacing some course in C'' by a more preferred course not contained in C''. Responsiveness is a very mild requirement and responsive preferences form a very wide and variable class. Therefore we shall restrict our attention to two specific examples, namely *additive* [6,9,17] and *lexicographic* [17–21] preferences. Although lexicographic preferences can be modelled as additive preferences by choosing appropriate weights [9], we would like to avoid this approach as it requires very large numerical values, moreover, assuming lexicographic preferences from the outset leads to more straightforward algorithms.

To the best of our knowledge, matchings with prerequisite constraints have not been studied yet from an algorithmic perspective. Some connections can be found in the literature on scheduling with precedence constraints [22], but, unlike in the scheduling domain, there is no common optimality criterion for all the agents, since their individual choices are often conflicting and all have to be taken into account.

We would however like to draw the reader's attention to the works of Guerin et al. [23] and Dodson et al. [24], who analyse a version of a course selection problem in greater depth, using probabilistic methods. Their work is a part of a larger research programme that involves advising college students about what courses to study and when, taking into account not only the required course prerequisites, but also the students' course histories and obtained grades. Based on this information, the authors try to estimate a student's ability to take multiple courses concurrently, with the goal to optimise the student's total expected utility and her chances of moving successfully towards graduation. They consider also temporal factors, meaning that a student can only take a certain course in the current semester if she has completed the necessary prerequisites during previous semesters. By contrast, here we assume that students choose all their courses as well as their necessary prerequisites simultaneously, and we concentrate on computational problems connected with producing a matching that fulfils a global welfare criterion.

Our contribution

As mentioned above, we will formally introduce a general model of course allocation involving prerequisites. We assume that applicants have preferences over individual courses and we consider two extensions of these preferences to preferences over bundles of courses. For additive preferences we show that computing a Pareto optimal matching is an NP-hard problem. In fact, we will show a stronger result, namely, that the problem of finding a most-preferred feasible bundle of courses for a given applicant in this setting is NP-hard.

In the case of lexicographic preferences we illustrate that the simple sequential allocation mechanism and its natural modification may output a matching that either violates prerequisites or Pareto optimality. By contrast, if we stipulate that on her turn, an applicant chooses her most preferred course together with all the necessary prerequisites, a Pareto optimal matching will be produced. Still, it is impossible to obtain each possible Pareto optimal matching in this way. It is also unlikely that an efficient algorithm will be able to produce all Pareto optimal matchings, since, as we will show, the problem of checking whether a given matching admits a Pareto improvement is NP-complete.

Then we consider structural properties of Pareto optimal matchings. It is known that in the simplest one-to-one model (naturally without prerequisites) finding a Pareto optimal matching with minimum cardinality is NP-hard, but a Pareto optimal matching with maximum cardinality can be found in polynomial time [13]. By contrast, we show that the problem of finding such a matching with maximum cardinality is NP-hard in both special cases, when prerequisites are acyclic as well as when they are symmetric.

For the case with uniform corequisites, i.e., identical for all applicants, we propose another modification of the sequential allocation mechanism for finding Pareto optimal matchings. This model is closely related to matchings with sizes [25] or many-to-many matchings with price-budget constraints [15], and we strengthen the existing results by showing that the problem of finding a maximum size Pareto optimal matching is not approximable within $N^{1-\varepsilon}$, for any $\varepsilon > 0$, unless $P = NP$, where N is the total capacity of the applicants.

The final model involves alternative prerequisites (i.e., where a constraint takes the form that an applicant can attend course c_1 only if she also attends either course c_2 or course c_3) seems to be computationally the most challenging one. We show that although a Pareto optimal matching always exists, it cannot be computed efficiently unless P=NP, both under additive as well as lexicographic preferences of applicants.

Our paper constitutes a first attempt to assess the computational complexity of Pareto optimal course allocation in the presence of prerequisites/corequisites. In terms of computational complexity we obtain mostly negative results for the general domain of additive preferences and some positive results for the special case of lexicographic preferences. This means that in practice, and for further research, the type of preferences that applicants have (i.e., whether they are close to being lexicographic or additive) and further institutional details (such as a limitation on the number or structure of prerequisites/corequisites) are extremely important; we demonstrate that no "one size fits all" solutions are available.

The organisation of the paper is as follows. In Section 2 we give formal definitions of the problem models and define relevant notation and terminology. Section 3 explains hardness of the problem with additive preferences and Section 4 contains the description of the modification of the sequential allocation mechanism for finding a Pareto optimal matching in the general model with prerequisites. Sections 5–7 deal separately with the three different models, involving acyclic prerequisites, corequisites and alternative prerequisites respectively. Finally, Section 8 concludes with some open problems and possible directions for future research.

2. Definitions and notation

2.1. Basic course allocation problem

An instance of the *Course Allocation problem* (CA) involves a set $A = \{a_1, a_2, \dots, a_{n_1}\}$ of *applicants* and a set $C = \{c_1, c_2, \dots, c_{n_2}\}$ of *courses*. Each course $c_j \in C$ has a capacity $q(c_j)$ that denotes the maximum number of applicants that can be assigned to c_j. Similarly each applicant $a_i \in A$ has a capacity $q(a_i)$ denoting the maximum number of courses that she can attend. The vector $\mathbf{q}$ denotes applicants' and courses' capacities. Moreover a_i has a strict linear order (preference list) $P(a_i)$ over a subset of C. We shall represent a_i's preferences $P(a_i)$ as a simple ordered list of courses, from the most preferred to the least preferred one. With some abuse of notation, we shall say that a course c_j is *acceptable* to applicant a_i if $c_j \in P(a_i)$, otherwise c_j is *unacceptable* to a_i. $\mathcal{P}$ denotes the n_1-tuple of applicants' preferences. Thus altogether, the tuple $I = (A, C, \mathbf{q}, \mathcal{P})$ constitutes an instance of CA.

An *assignment* M is a subset of $A \times C$. The set of applicants assigned to a course $c_j \in C$ will be denoted by $M(c_j) = \{a_i \in A : (a_i, c_j) \in M\}$ and similarly, the *bundle* of courses assigned to an applicant a_i is $M(a_i) = \{c_j \in C : (a_i, c_j) \in M\}$. An assignment M is a *matching* if, for each $a_i \in A$, $M(a_i) \subseteq P(a_i)$ and $|M(a_i)| \le q(a_i)$, and for each $c_j \in C$, $|M(c_j)| \le q(c_j)$. In the presence of prerequisites, additional *feasibility* constraints are to be satisfied by a matching, which will be defined below. An applicant $a_i \in A$ is said to be *undersubscribed* (respectively *full*) in a matching M if $|M(a_i)| < q(a_i)$ (respectively $|M(a_i)| = q(a_i)$). Similarly we may define *undersubscribed* and *full* for a course c_j relative to M.

An applicant $a_i \in A$ has *additive preferences* over bundles of courses if she has a utility $u_{a_i}(c_j)$ for each course $c_j \in C$, and she prefers a bundle of courses $C_1 \subseteq C$ to another bundle $C_2 \subseteq C$ if and only if $\sum_{c_j \in C_1} u_{a_i}(c_j) > \sum_{c_j \in C_2} u_{a_i}(c_j)$.

Applicant a_i compares bundles of courses *lexicographically* if, given two different bundles $C_1 \subseteq C$ and $C_2 \subseteq C$ she prefers C_1 to C_2 if and only if her most preferred course in the symmetric difference $C_1 \oplus C_2$ belongs to C_1. Notice that the lexicographic ordering on bundles of courses generated by a strict preference order $P(a_i)$ is also strict.

Applicant a_i prefers matching M' to matching M if she prefers $M'(a_i)$ to $M(a_i)$. We say that a matching M' *(Pareto) dominates* a matching M if at least one applicant prefers M' to M and no applicant prefers M to M'.

A *Pareto optimal matching* (or *POM* for short), is a matching that is not (Pareto) dominated by any other matching. As the dominance relation is a partial order over $\mathcal{M}$, the set of all matchings in I, and $\mathcal{M}$ is finite, a POM exists for each instance of CA.

2.2. Prerequisites

We now define the extension of CA involving prerequisites. Suppose that for each applicant $a_i \in A$, there is a directed graph $D_{a_i} = (C, E_{a_i})$. If there is an arc $(c_j, c_k) \in E_{a_i}$ we say that course c_k is an *immediate prerequisite* of course c_j for applicant a_i. By transitivity, the prerequisite relation $\rightarrow_{a_i}$ of applicant a_i is then defined in the following way: course c_l is a *prerequisite* of course c_j if the corresponding vertex c_l is reachable from vertex c_j in the digraph D_{a_i}; we shall denote this by $c_j \rightarrow_{a_i} c_l$.

Now define the *feasibility* of a bundle of courses relative to the prerequisites constraints as follows.

Definition 1. A bundle of courses $C' \subseteq C$ is *feasible* for an applicant $a_i \in A$ if the following three conditions are fulfilled:

(i) $C' \subseteq P(a_i)$;
(ii) $|C'| \leq q(a_i)$;
(iii) C' fulfils a_i's prerequisites, i.e., for each $c_j, c_k \in C$, if $c_j \in C'$ and $c_j \rightarrow_{a_i} c_k$ then $c_k \in C'$.

We shall denote by $\overset{\rightarrow a_i}{c_j}$ the *set of courses that are reachable from vertex c_j in the prerequisites digraph D_{a_i}*, including vertex c_j itself, i.e., $\overset{\rightarrow a_i}{c_j} = \{c_k \in C; c_j \rightarrow_{a_i} c_k\} \cup \{c_j\}$. For technical reasons, we assume that $\overset{\rightarrow a_i}{c_j} \subseteq P(a_i)$ for each $a_i \in A$ and each $c_j \in P(a_i)$. If this is not the case then we can easily modify the preference list of applicant a_i either by deleting a course c_j if $P(a_i)$ does not contain all the courses in $\overset{\rightarrow a_i}{c_j}$, or we can append the missing courses to the end of $P(a_i)$.

An instance I of the *Course Allocation problem with PreRequisites* (CAPR) comprises a tuple $I = (A, C, \mathbf{q}, \mathcal{P}, \rightarrow)$, where $(A, C, \mathbf{q}, \mathcal{P})$ is an instance of CA and $\rightarrow$ is the n_1-tuple of prerequisite relations $\rightarrow_{a_i}$ for each applicant $a_i \in A$ (that are without loss of generality obtained from an n_1-tuple of prerequisite digraphs as described in the first paragraph). In an instance of CAPR, a *matching* M is as defined in the CA case, together with the additional property that, for each applicant $a_i \in A$, $M(a_i)$ is feasible for a_i.

In an instance of CAPR where the prerequisites are the same for all applicants, (i.e., when the prerequisites are *uniform*) we may drop the applicant subscript when referring to the prerequisite relation $\rightarrow$.

A special case of CAPR occurs when the prerequisites digraph D_{a_i} is acyclic for each applicant $a_i \in A$. This case corresponds to the situations that arise most often in reality.

2.3. Corequisites

Another special case of CAPR occurs when constraints on courses are given in the form of corequisites. We assume that, for each applicant $a_i \in A$, the digraph D_{a_i} is *symmetric*, i.e., $(c_j, c_k) \in E_{a_i}$ holds if and only if $(c_k, c_j) \in E_{a_i}$ for any two courses $c_j, c_k \in C$. This means that each digraph D_{a_i} is in fact an undirected graph (we keep the notation D_{a_i}) and it turns out that for our purposes only connected components are relevant. These connected components effectively partition the set of courses into equivalence classes $C^1_{a_i}, C^2_{a_i}, \ldots, C^r_{a_i}$. Two courses $c_j, c_k \in C$ in the same equivalence class are said to be each other's *corequisites*; we write $c_j \leftrightarrow_{a_i} c_k$. Hence applicant a_i can subscribe either to all courses in one equivalence class or to none.

Formally, we define a bundle of courses $C' \subseteq C$ to be *feasible* for a given applicant $a_i \in A$ if Conditions (i) and (ii) in Definition 1 are satisfied, and moreover, for any two courses $c_j, c_k \in C$, if $c_j \leftrightarrow_{a_i} c_k$ then $c_j \in C'$ if and only if $c_k \in C'$. An instance I of the *Course Allocation problem with CoRequisites* (CACR) is a tuple $I = (A, C, \mathbf{q}, \mathcal{P}, \leftrightarrow)$, where $(A, C, \mathbf{q}, \mathcal{P})$ is an instance of CA and $\leftrightarrow$ is an n_1-tuple of corequisite relations $\leftrightarrow_{a_i}$ for each applicant $a_i \in A$ (that are without loss of generality obtained from an n_1-tuple of corequisite digraphs as described above). In an instance of CACR, a *matching* M is as defined in the CA case, together with the additional property that, for each applicant $a_i \in A$, $M(a_i)$ is feasible for a_i.

We shall pay particular attention to the special case of CACR that arises when corequisites are *uniform*, i.e., $D_{a_i} = D_{a_j}$ for each a_i, $a_j \in A$ — we denote this restriction by CAUCR (the *Course Allocation problem with Uniform CoRequisites*). This reflects the fact that, often in practice, corequisite constraints on courses are not applicant-specific.

2.4. Alternative prerequisites

While our previous models correspond to the situations when the prerequisites are compulsory, our final model is a generalisation of CAPR in which prerequisites are in general presented in the form of alternatives. Formally, each applicant a_i has a mapping $\mapsto_{a_i} : C \to 2^C$ with the following meaning: if $c_j \mapsto_{a_i} \{c_{i_1}, c_{i_2}, \ldots, c_{i_k}\}$, it must then hold that if a_i wants to attend course c_j then she has to attend at least one of the courses $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ too.

We thus define a bundle of courses $C' \subseteq C$ to be *feasible* for a given applicant $a_i \in A$ if Conditions (i) and (ii) in Definition 1 are satisfied, and moreover, for any course $c_j \in C'$, if $c_j \mapsto_{a_i} \{c_{i_1}, c_{i_2}, \ldots, c_{i_k}\}$, then $c_{i_r} \in C'$ for some r $(1 \le r \le k)$.

An instance I of the *Course Allocation problem with Alternative PreRequisites* (CAAPR) comprises a tuple $I = (A, C, \mathbf{q}, \mathcal{P}, \mapsto)$, where $(A, C, \mathbf{q}, \mathcal{P})$ is an instance of CA and $\mapsto$ is the n_1-tuple of mappings $\mapsto_{a_i}$ for each applicant $a_i \in A$. In an instance of CAAPR, a *matching* M is as defined in the CA case, together with the additional property that, for each applicant $a_i \in A$, $M(a_i)$ is feasible for a_i.

3. Additive preferences

It turns out that the problem of finding a POM under additive preferences is NP-hard, even in the case when all applicants have acyclic or all have symmetric prerequisites. Note that in the NP-hardness results in this section it does not matter if additive preferences are implicitly given or need to be explicitly elicited, because a single applicant can be assumed to know her course utility values.

Lemma 2. *The problem of finding a most-preferred feasible bundle of courses of a given applicant with additive preferences in* CAPR *is NP-hard, even when the prerequisite digraph of the applicant is acyclic.*

Proof. We transform from the KNAPSACK problem, which is defined as follows. An instance I of KNAPSACK comprises a set of integers $w_1, w_2, \ldots, w_n, p_1, p_2, \ldots, p_n$, W, P. The problem is to decide whether there exists a set $K \subseteq \{1, 2 \ldots, n\}$ such that $\sum_{i \in K} w_i \le W$ and $\sum_{i \in K} p_i \ge P$. KNAPSACK is NP-complete [26].

Let us construct an instance J of CAPR as follows. There is a single applicant a_1 such that $q(a_1) = W$. The set of courses is $C \cup D$, where $C = \{c_1, c_2, \ldots, c_n\}$ and $D = \cup_{i=1}^{n} \{d_i^1, d_i^2, \ldots, d_i^{w_i - 1}\}$. For each course c_i $(1 \le i \le n)$, its utility for applicant a_1 is equal to $p_i + \delta_i$, where δ_i will be defined shortly. Moreover c_i has $w_i - 1$ immediate prerequisites $d_i^1, d_i^2, \ldots, d_i^{w_i - 1}$. For each i $(1 \le i \le n)$ and j $(1 \le j \le w_i - 1)$, d_i^j has utility ε_i^j for a_1, where the δ and ε values are all positive, distinct and add up to less than 1. They are selected simply to ensure that a_1's preferences over individual courses are strict. It is easy to see that in J, a_1 has a feasible bundle with utility at least P if and only if I is a yes-instance of KNAPSACK.

Clearly a polynomial-time algorithm for finding a most-preferred feasible bundle of courses for a_1 can be used to determine whether a_1 has a feasible bundle in J with utility at least P, hence the result. □

Notice that the arguments in the proof of the above lemma also remain valid if we stipulate that $c_i \leftrightarrow d_i^j$ holds for each i $(1 \le i \le n)$ and j $(1 \le j \le w_i - 1)$. So we also have the following assertion.

Lemma 3. *The problem of finding a most-preferred feasible bundle of courses of a given applicant with additive preferences in* CACR *is NP-hard.*

In the case of a single applicant a_1, a matching M is a POM if and only if a_1 is assigned in M a most-preferred feasible bundle of courses, otherwise M is dominated by assigning a_1 to this bundle. Hence the above lemma directly implies the following result.

Theorem 4. *Given an instance of* CAPR *with additive preferences, the problem of finding a POM is NP-hard even if there is only a single applicant and even when her prerequisites are acyclic or when they are symmetric.*

Given these negative results, we do not pursue additive preferences any further in this paper, and instead turn our attention to lexicographic preferences.

4. Finding a POM for lexicographic preferences

From now on we assume that preferences of applicants are lexicographic. In this section we explore variants of the sequential allocation mechanism, and show that one formulation allows us to find a POM in polynomial time. This mechanism, referred to as SM-CAPR, does however have some drawbacks: it is not truthful (see Section 8) and it is not able to compute all POMs in general (see Example 9).

In the context of course allocation when there are some dependencies among courses (for instance the constraints on prerequisites in CAPR) the standard sequential mechanism might output an allocation that does not fulfil some constraints on prerequisites. On the other hand, if we require that an applicant can only choose a course if she is already assigned all its prerequisites, the output may be a matching that is not Pareto optimal. This is illustrated by the following example.

Example 5. Construct a CAPR instance in which $A = \{a_1, a_2\}$ and $C = \{c_1, c_2, c_3, c_4\}$. Each applicant has capacity 2 and each course has capacity 1. The prerequisites of both applicants are the same, and are as follows:

$$c_1 \to c_3; \qquad c_2 \to c_4.$$

The applicants have the following preference lists:

$$P(a_1) : c_1, c_2, c_4, c_3$$
$$P(a_2) : c_2, c_1, c_3, c_4.$$

The sequential allocation mechanism with policy $\sigma = a_1, a_2, a_1, a_2$ will assign to applicant a_1 the bundle $\{c_1, c_4\}$ and to applicant a_2 the bundle $\{c_2, c_3\}$. Clearly, neither of the assigned bundles fulfils the prerequisites.

Now suppose that in the sequential allocation mechanism an applicant is allowed to choose the most-preferred undersubscribed course *for which she already has all the prerequisites*. Let the policy start with a_1, a_2. Applicant a_1 can choose neither c_1 nor c_2, as these courses require a prerequisite that she is not assigned yet. So she chooses c_4. Similarly, applicant a_2 will choose c_3. When these applicants are allowed to pick their next course, irrespective of the remainder of the policy, a_1 must choose c_2 and a_2 must choose c_1. So in the resulting matching M we have $M(a_1) = \{c_2, c_4\}$ and $M(a_2) = \{c_1, c_3\}$. This matching is clearly not Pareto optimal, since both applicants (having lexicographic preferences) will strictly improve by exchanging their assignments. □

Therefore we propose a variant of the sequential allocation mechanism, denoted by SM-CAPR, that can be regarded as being "in between" the serial dictatorship mechanism and the general sequential allocation

Algorithm 1 SM-CAPR

```
Require: CAPR instance $I$ and a policy $\sigma$
Ensure: return $M$, a POM in $I$
 1: $M := \emptyset$;
 2: for each applicant $a_i \in \sigma$ in turn do
 3:     feasible := false;
 4:     while $a_i$ has not exhausted her preference list and not feasible do
 5:         $c_j$ := next course in $a_i$'s list;
 6:         if $c_j \notin M(a_i)$ and $c_j$ is undersubscribed then
 7:             $S := \overset{\rightarrow a_i}{c_j}$;
 8:             feasible := true;
 9:             for each $c_k \in S$ do
10:                 if $c_k \in M(a_i)$ then
11:                     $S := S \backslash \{c_k\}$;
12:                 else if $c_k$ is full then
13:                     feasible := false;
14:             if feasible then
15:                 if $|M(a_i)| + |S| \leq q(a_i)$ then
16:                     for each $c_k \in S$ do
17:                         $M := M \cup \{(a_i, c_k)\}$;
18:                 else
19:                     feasible := false;
20: return $M$;
```

mechanism. Suppose a policy σ is fixed; again one applicant can appear in σ several times, up to her capacity. Applicant a_i on her turn identifies her most-preferred course c_j that she has not yet considered, and that she is not already assigned to (if no such course c_j exists then a_i is said to have *exhausted* her preference list and will be assigned no more courses). If c_j is full or a_i is already assigned to c_j then a_i considers the next course on her list (on the same turn). We then compute the reachable set $\overset{\rightarrow a_i}{c_j}$ of c_j (we shall explain how to do this efficiently in the proof of Theorem 6). If all courses c_k in $\overset{\rightarrow a_i}{c_j}$ satisfy the property that either (i) c_k has a free place or (ii) c_k is already assigned to a_i, and the number of courses in $\overset{\rightarrow a_i}{c_j}$ not already assigned to a_i does not exceed the remaining capacity of a_i, then a_i is assigned the bundle $\overset{\rightarrow a_i}{c_j}$. If it is impossible to assign bundle $\overset{\rightarrow a_i}{c_j}$ to a_i (as signified by the boolean *feasible* becoming `false`) then a_i moves to the next course in her preference list until either she is assigned to some bundle or her preference list is exhausted. This completes a single turn for a_i.

The pseudocode for SM-CAPR is given in Algorithm 1. This algorithm constructs a POM M in a given CAPR instance I relative to a given policy σ. Notice that serial dictatorship will be obtained as a special case of SM-CAPR if all the copies of one applicant form a substring (i.e., a contiguous subsequence) of the policy. We now show that SM-CAPR constructs a POM and is an efficient algorithm.

Theorem 6. *Algorithm SM-CAPR produces a POM for a given instance I of CAPR and for a given policy σ in I. The complexity of the algorithm is $O(N+L(n_2+|E|))$, where N is the sum of the applicants' capacities, n_2 is the number of courses, L is the total length of the applicants' preference lists and $|E| = \max_{a_i \in A}|E_{a_i}|$ where $|E_{a_i}|$ is the number of arcs in D_{a_i}.*

Proof. It is straightforward to verify that the assignment M produced by SM-CAPR is a matching in I. Suppose for a contradiction that M is not a POM. Then there exists a matching M' that dominates M. Let A' be the set of applicants who prefer their assignment in M' to their assignment in M. Define a *stage* to be an iteration of the **while** loop, and for a given stage s, define its *number* to be the integer k such that

s is the kth iteration of the **while** loop, taken over the entire execution of the algorithm. For each $a_j \in A'$, consider the first stage where a course, say c_{i_j}, was identified for a_j at line 5, such that $c_{i_j} \in M'(a_j) \setminus M(a_j)$, and let s_j be the number of this stage. Let $a_k = \arg\min_{a_j \in A'}\{s_j\}$.

As $c_{i_k} \in M'(a_k)$, all the prerequisites of c_{i_k} also belong to $M'(a_k)$. Since s_k is the first stage in which a course c_{i_k} was identified for a_k in line 5, such that $c_{i_k} \in M'(a_k) \setminus M(a_k)$, all the courses assigned in M to any applicant a_j in previous stages also belong to $M'(a_j)$, for otherwise M' does not dominate M. Also, clearly $c_{i_k} \notin M(a_k)$. Thus it was not the case that applicant a_k failed to receive course c_{i_k} in M at stage s_k because a_k did not have room for c_{i_k} and all of its prerequisites not already assigned to her in M. Rather, applicant a_k failed to receive course c_{i_k} in M at stage s_k because at least one course in $\overset{\rightarrow a_k}{c_{i_k}}$, say c_r, was already full in M before stage s_k. It follows from our previous remark that in M', all the places in c_r are occupied by applicants other than a_k. Thus c_{i_k} cannot be assigned to a_k in M' after all, a contradiction.

To derive the complexity bound of the algorithm, let us first observe that the number of iterations of the **for** loop in line 2 is $O(|\sigma|) = O(N)$, where N is the sum of the applicants' capacities, whilst the total number of iterations of the **while** loop in line 4, taken over the entire execution of the algorithm, is $O(L)$, where L is the total length of the applicants' preference lists. At line 5 we assume that the next course for an applicant a_i is maintained by a pointer that initially points to the head of a_i's preference list, and once the course c_j pointed to by a_i's pointer has been found, the pointer moves on one position (if c_j was the last course on a_i's list then the pointer becomes null, indicating that a_i has exhausted her list).

Now consider the steps needed when exploring a course c_j in the preference list of applicant a_i. To construct the list of courses in $S = \overset{\rightarrow a_i}{c_j}$ in line 7, we have to search the digraph D_{a_i}; by standard graph traversal techniques this requires $O(|E_{a_i}|)$ steps, where $|E_{a_i}|$ is the number of arcs in D_{a_i}. Each of the **for** loops occupying lines 9–13 and 16–17 has $O(n_2)$ overall complexity, as the size of S is $O(n_2)$. This means that the total time required by the algorithm is $O(N + L(n_2 + |E|))$, where $|E| = \max_{a_i \in A}|E_{a_i}|$ and L is the total length of the applicants' preference lists. $\square$

The complexity of SM-CAPR is no better than $O(N + L(n_2 + |E|))$ in the worst case, as the following example shows. Notice however that since the construction involves a long chain of prerequisites, in more realistic cases we can expect that SM-CAPR performs much better than the theoretical upper bound may suggest.

Example 7. Consider a CAPR instance in which $A = \{a_1, a_2, \ldots, a_n\}$ is the set of applicants and $C = \{c_1, c_2, \ldots, c_{2n}\}$ is the set of courses, for some $n \geq 1$. Assume that each course has capacity 1, whilst each applicant has capacity n and ranks all courses in increasing indicial order. Also suppose that the immediate prerequisites for each applicant are as follows:

$$c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_{2n}.$$

There are n POMs: in the POM M_i $(1 \leq i \leq n)$, a_i is assigned the set of courses $\{c_{n+1}, c_{n+2}, \ldots, c_{2n}\}$ and no course is assigned to any other applicant. Given any policy σ, let a_i be the first applicant considered during an execution of SM-CAPR. When $\overset{\rightarrow a_i}{c_1}$ is constructed in line 7, $2n$ courses are returned; likewise when $\overset{\rightarrow a_i}{c_2}$ is constructed, $2n - 1$ courses are returned, and so on. This continues until $\overset{\rightarrow a_i}{c_{n+1}}$ is constructed, leading to the matching M_i being formed at this **while** loop iteration. Note that even if the process of constructing $S = \overset{\rightarrow a_i}{c_j}$ were to halt as soon as $|S| > q(a_i)$, the total number of courses checked at this step of SM-CAPR is still $\Omega(n^2)$. Similarly, the number of courses checked at each other applicant's turn in the policy is also $\Omega(n^2)$; the only difference being that in each such case SM-CAPR determines that c_{n+i} is full, for each i $(1 \leq i \leq n)$. The overall number of steps used by SM-CAPR is then $\Omega(n^3) = O(N + L(n_2 + |E|))$. $\square$

Our next two examples indicate that in general, SM-CAPR is not capable of generating all POMs for a given CAPR instance, even if there are only two applicants and three courses and the capacity of each course is 1. In Example 8 the prerequisites of all applicants are the same. In Example 9 they are different, but each course has at most one prerequisite.

Example 8. Let $A = \{a_1, a_2\}$, $C = \{c_1, c_2, c_3\}$, capacity of each course is 1 and course c_1 have two prerequisites as follows:

$$c_1 \to c_2; \qquad c_1 \to c_3. \tag{1}$$

Each applicant has capacity 3 and the following preference list: c_1, c_2, c_3.

Depending on the policy, SM-CAPR outputs either the matching that assigns all three courses to a_1, or the matching that assigns all three courses to a_2. However, it is easy to see that the two matchings that assign c_2 to one applicant and c_3 to the other one are also Pareto optimal.

Example 9. The instance is the same as in the previous example; only the prerequisites of the applicants are different:

$$c_1 {\to_{a_1}} c_3; \qquad c_2 {\to_{a_2}} c_3. \tag{2}$$

Each applicant has capacity 2 and their preferences are as follows:

$$P(a_1) : c_1, c_2, c_3 \qquad P(a_2) : c_2, c_1, c_3.$$

There are 4 different POMs, as follows:

$$\begin{array}{ll} M_1(a_1) = \{c_1, c_3\}, & M_1(a_2) = \emptyset; \\ M_2(a_1) = \emptyset, & M_2(a_2) = \{c_2, c_3\}; \\ M_3(a_1) = \{c_2\}, & M_3(a_2) = \{c_1, c_3\}; \\ M_4(a_1) = \{c_2, c_3\}, & M_3(a_2) = \{c_1\}. \end{array}$$

SM-CAPR outputs M_1 with a policy in which a_1 is first, and M_2 with a policy in which a_2 is first. Notice that neither M_3 nor M_4 can be obtained by SM-CAPR.

Theorem 4 shows that finding a POM in the presence of additive preferences is NP-hard. It is instructive to show where SM-CAPR can fail to find a POM in this case.

Example 10. Let I be an instance of CAPR in which there are two applicants, a_1, a_2, each of which has capacity 2, and four courses, c_1, c_2, c_3, c_4, each of which has capacity 1. The prerequisites of both applicants are the same, and are as follows:

$$c_1 \to c_2; \qquad c_3 \to c_4.$$

The utilities of the courses for the applicants are as follows:

$$\begin{aligned} u_{a_1}(c_1) = u_{a_2}(c_3) = 6 \\ u_{a_1}(c_3) = u_{a_2}(c_1) = 4 \\ u_{a_1}(c_4) = u_{a_2}(c_2) = 3 \\ u_{a_1}(c_2) = u_{a_2}(c_4) = 0 \end{aligned}$$

and their ordinal preferences are consistent with these utilities. Regardless of the policy, SM-CAPR constructs the matching $M = \{(a_1, c_1), (a_1, c_2), (a_2, c_3), (a_2, c_4)\}$. M is not a POM as it is dominated by $M' = \{(a_1, c_3), (a_1, c_4), (a_2, c_1), (a_2, c_2)\}$. $\square$

In the following sections we consider separately the special cases of CAPR involving acyclic prerequisites and symmetric prerequisites (corequisites). We shall deal with the problems of testing a given matching for Pareto optimality and with the structure of the set of POMs.

5. Acyclic preferences

In the previous section we gave a polynomial-time algorithm for constructing a POM in a general instance of CAPR. However, this algorithm was not able to construct all POMs, even when prerequisite digraphs were acyclic. An alternative approach could involve starting with an arbitrary matching, and for as long as the current matching M is dominated, replace M by any matching that dominates it. However, the difficulty with this method is that the problem of determining whether a matching is Pareto optimal is computationally hard, as we demonstrate by our next result. This hardness result also shows that there is unlikely to be a "nice" (polynomial-time checkable) characterisation of a POM, in contrast to the case where there are no prerequisites [15]. We firstly define the following problems:

Name: EXACT 3- COVER
Instance: a set $X = \{x_1, x_2, \dots, x_{3n}\}$ and a set $\mathcal{T} = \{T_1, T_2, \dots, T_m\}$ such that for each i $(1 \le i \le m)$, $T_i \subseteq X$ and $|T_i| = 3$.
Question: is there a subset $\mathcal{T}'$ of $\mathcal{T}$ such that $T_i \cap T_j = \emptyset$ for each $T_i, T_j \in \mathcal{T}'$ and $\cup_{T_i \in \mathcal{T}'} T_i = X$?

Name: DOM CAPR
Instance: an instance I of CAPR and a matching M in I
Question: is there a matching M' that dominates M in I?

Theorem 11. DOM CAPR *is NP-complete even if prerequisites are acyclic, each course has capacity 1 and has at most one immediate prerequisite for each applicant.*

Proof. Clearly DOM CAPR is in NP. To show NP-hardness, we reduce from EXACT 3- COVER, which is NP-complete [26]. Let I be an instance of EXACT 3- COVER as defined above. For each $T_i \in \mathcal{T}$, let us denote the elements that belong to T_i by $x_{i_1}, x_{i_2}, x_{i_3}$. Obviously, we lose no generality by assuming that $m \ge n$.

We construct an instance J of DOM CAPR based on I in the following way. The set of applicants is $A = \{a_1, a_2, \dots, a_{m+1}\}$. The capacities are $q(a_i) = 4$ $(1 \le i \le m)$ and $q(a_{m+1}) = 2n + m$. The set of courses is $C = \mathcal{T} \cup X \cup Y \cup W$, where $\mathcal{T} = \{T_1, T_2, \dots, T_m\}$, $X = \{x_1, x_2, \dots, x_{3n}\}$, $Y = \{y_1, y_2, \dots, y_m\}$ and $W = \{w_1, w_2, \dots, w_{m-n}\}$. (Some of the courses in J derived from the elements and sets in I are denoted by identical symbols, but no ambiguity should arise.) Each course has capacity 1. The preferences of the applicants are:

$$\begin{aligned} P(a_i): &\quad T_i, [W], y_i, x_{i_1}, x_{i_2}, x_{i_3} \qquad (1 \le i \le m) \\ P(a_{m+1}): &\quad y_1, [X], [W], y_2, \dots, y_m \end{aligned}$$

where the symbols $[W]$ and $[X]$ denote all the courses in W and X, respectively, in an arbitrary strict order. Recall that $\{x_{i_1}, x_{i_2}, x_{i_3}\} \subseteq X$ $(1 \le i \le m)$. The prerequisites of applicants are:

$$\begin{aligned} a_i: &\quad T_i \to_{a_i} x_{i_1} \to_{a_i} x_{i_2} \to_{a_i} x_{i_3} \qquad\qquad (1 \le i \le m) \\ a_{m+1}: &\quad y_1 \to_{a_{m+1}} y_2 \to_{a_{m+1}} \cdots \to_{a_{m+1}} y_m \end{aligned}$$

Define the following matching:

$$M = \{(a_i, y_i) : 1 \le i \le m\} \cup \{(a_{m+1}, x_j) : 1 \le j \le 3n\} \cup \{(a_{m+1}, w_k) : 1 \le k \le m - n\}.$$

We claim that I admits an exact cover if and only if M is dominated in J.

For, suppose that $\{T_{j_1}, T_{j_2}, \ldots, T_{j_n}\}$ is an exact cover in I. We construct a matching M' in J as follows. For each k $(1 \le k \le n)$, in M', assign a_{j_k} to T_{j_k} and to a_{j_k}'s three prerequisites of T_{j_k} that belong to X. Let $A' = \{a_{j_1}, a_{j_2}, \ldots, a_{j_n}, a_{m+1}\}$ and let $A \setminus A' = \{a_{k_1}, a_{k_2}, \ldots, a_{k_{m-n}}\}$. For each r $(1 \le r \le m-n)$, in M', assign a_{k_r} to w_r. Finally in M', assign a_{m+1} to every course in Y. It is straightforward to verify that M' dominates M in J.

Conversely, suppose there exists a matching M' that dominates M in J. Then, at least one applicant must be better off in M' compared to M.

If a_{m+1} improves, she must obtain y_1 and so, due to her prerequisites, all the courses in Y. This means that each applicant a_i $(1 \le i \le m)$ must obtain a course that she prefers to y_i.

Each such applicant a_i can improve relative to M in two ways. Either she obtains in M' a course in W, or she obtains T_i. In the latter case then she must receive the corresponding courses $x_{i_1}, x_{i_2}, x_{i_3}$ in M'. In either of these two cases, since a_{m+1} cannot be worse off, she must obtain in M' the course y_1 and hence all courses in Y.

This means that in M' all the applicants must strictly improve compared to M. As there are only $n-m$ courses in W, there are exactly n applicants in $A \setminus \{a_{m+1}\}$ – let these applicants be $\{a_{j_1}, a_{j_2}, \ldots, a_{j_n}\}$ – who obtain a course in $\mathcal{T}$ and its three prerequisites in X. As the capacity of each course is 1, it follows that $\{T_{j_k} : 1 \le j \le n\}$ is an exact cover in I. □

We remark that the variant of DOM CAPR for additive preferences is also NP-complete by Theorem 11, since lexicographic preferences can be viewed as a special case of additive preferences by creating utilities that steeply decrease in line with applicants' preferences – see [9] for more details.

Now we turn our attention to the size of POMs. Example 8 shows that an instance of CAPR may admit POMs of different cardinalities, where the *cardinality* of a POM refers to the number of occupied course slots. It is known that finding a POM with minimum cardinality is an NP-hard problem even for HA, the House Allocation problem (i.e., the restriction of CA in which each applicant and each course has capacity 1) [13, Theorem 2]. However, for HA [13, Theorem 1] and CA [15, Theorem 6]), the problem of finding a maximum cardinality POM is solvable in polynomial time. By contrast, the same problem in the CAPR context is NP-hard, as we demonstrate next via two different proofs. Our first proof of this result shows that hardness holds even if the matching is not required to be Pareto optimal.

We firstly define some problems. Let MAX CAPR and MAX POM CAPR denote the problems of finding a maximum cardinality matching and a maximum cardinality POM respectively, given an instance of CAPR. Let MAX CAPR-D denote the decision version of MAX CAPR: given an instance I of CAPR and an integer K, decide whether I admits a matching of cardinality at least K. We obtain MAX POM CAPR-D from MAX POM CAPR similarly.

Theorem 12. MAX CAPR-D *is NP-complete, even if prerequisites are acyclic, each applicant has capacity 4 and each course has capacity 1, and the prerequisites are the same for all applicants.*

Proof. Clearly MAX CAPR-D is in NP. To show NP-hardness, we reduce from IND SET-D in cubic graphs; here IND SET-D is the decision version of IND SET, the problem of finding a maximum independent set in a given graph. IND SET-D is NP-complete in cubic graphs [27,28]. Let $\langle G, K \rangle$ be an instance of IND SET-D in cubic graphs, where $G = (V, E)$ is a cubic graph and K is a positive integer. Assume that $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. For a given vertex $v_i \in V$, let $E_i \subseteq E$ denote the set of edges incident to v_i in G. Clearly $|E_i| = 3$ as G is cubic.

We form an instance I of MAX CAPR-D as follows. Let A be the set of applicants and let $V \cup E$ be the set of courses, where $A = \{a_i : v_i \in V\}$ (we use the same notation for the vertices and edges in G as we do for the courses in I, but no ambiguity should arise.) Let the capacity of each applicant be 4 and the capacity of each course be 1. The preference list of each applicant is as follows:

$$a_i : v_i \;\; [E_i] \qquad (1 \le i \le n)$$

where the symbol $[E_i]$ denotes all members of E_i listed in arbitrary order. For each $v_i \in V$ and for each $e_j \in E_i$, define the prerequisite $v_i \to e_j$ for all applicants. We claim that G has an independent set of size at least K if and only if I has a matching of size at least $m + K$.

For, suppose that S is an independent set in G where $|S| \ge K$. Let $A' = \{a_i \in A : v_i \in S\}$. We form an assignment M in I as follows. For each applicant $a_i \in A'$, assign a_i to v_i plus the prerequisite courses in E_i. Then for each applicant $a_i \notin A'$, assign a_i to any remaining courses in E_i (if any). It is straightforward to verify that M is a matching in I. Also $|M| = m + |S| \ge m + K$, since every applicant $a_i \in A'$ obtains v_i and all prerequisite courses in E_i, and then the applicants in $A \setminus A'$ are collectively assigned to all remaining unmatched courses in E.

Conversely suppose that M is a matching in I such that $|M| \ge m + K$. Let S denote the courses in V that are matched in M, and suppose that $|S| < K$. Then since $|E| = m$ and all courses have capacity 1, $|M| \le |S| + |E| \le m + |S| < m + K$, a contradiction. Hence $|S| \ge K$. We now claim that S is an independent set in G. For, suppose that v_i and v_j are two adjacent vertices in G that are both in S. Clearly $(a_i, v_i) \in M$ and $(a_j, v_j) \in M$. But since v_i and v_j are adjacent in G, it is then impossible for both a_i and a_j to meet their prerequisites on v_i and v_j in I, respectively, a contradiction. □

Corollary 13. MAX POM CAPR-D *is NP-hard, even if prerequisites are acyclic and uniform, each applicant has capacity 4 and each course has capacity 1.*

Proof. We firstly remark that, in view of Theorem 11, it is not known whether MAX POM CAPR-D belongs to NP. To show NP-hardness, notice that in the proof of Theorem 12, the matching M in I constructed from an independent set S in G is in fact Pareto optimal. To see this, let σ be an ordering of the applicants such that every applicant in A' precedes every applicant in $A \setminus A'$. Let M be the result of running Algorithm SM-CAPR relative to the ordering σ. It follows by Theorem 6 that M is a POM in I. The remainder of the proof of Theorem 12 can then be used to show that MAX POM CAPR-D is NP-hard. □

We now give an alternative proof of Corollary 13 for the case that the constructed matching is required to be Pareto optimal. This gives NP-hardness for stronger restrictions on prerequisites than those given by Corollary 13. Our reduction involves a transformation from the following NP-complete problem [29]:

Name: (2,2) -E3 -SAT
Instance: a Boolean formula B, where each clause in B has size three, and each variable occurs exactly twice as an unnegated literal and exactly twice as a negated literal in B.
Question: is B satisfiable?

Theorem 14. MAX POM CAPR-D *is NP-hard, even if prerequisites are acyclic and uniform, each applicant has capacity at most 2, each course has capacity 1 and at most one prerequisite.*

Proof. We reduce from (2,2) -E3 -SAT as defined above. Let B be an instance of (2,2) -E3 -SAT, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of variables and $C = \{c_1, c_2, \dots, c_m\}$ is the set of clauses. We construct an instance I of MAX POM CAPR-D as follows. Let $X \cup Y$ be the set of courses, where $X = \{x_i^1, x_i^2, \bar{x}_i^1, \bar{x}_i^2 : 1 \le i \le n\}$ and $Y = \{y_i^1, y_i^2 : 1 \le i \le n\}$. The courses in X correspond to the first and second occurrences of v_i and $\bar{v}_i$ in B for each i $(1 \le i \le n)$. Let $A \cup G$ be the set of applicants, where $A = \{a_j : 1 \le j \le m\}$

and $G = \{g_i^1, g_i^2 : 1 \le i \le n\}$. Each course has capacity 1. Each applicant in A has capacity 1, whilst each applicant in G has capacity 2. For each i $(1 \le i \le n)$, define the prerequisite $y_i^1 \to y_i^2$, which is the same for all applicants. For each j $(1 \le j \le m)$ and for each s $(1 \le s \le 3)$, let $x(c_j^s)$ denote the X-course corresponding to the literal appearing at position s of clause c_j in B. For example if the second position of clause c_5 contains the second occurrence of literal $\bar{v}_i$, then $x(c_5^2) = \bar{x}_i^2$. The preference lists of the applicants are as follows:

$$\begin{array}{lll} P(a_j): & x(c_j^1), x(c_j^2), x(c_j^3) & (1 \le j \le m) \\ P(g_i^1): & y_i^1, y_i^2, x_i^1, x_i^2 & (1 \le i \le n) \\ P(g_i^2): & y_i^1, y_i^2, \bar{x}_i^1, \bar{x}_i^2 & (1 \le i \le n). \end{array}$$

We claim that B has a satisfying truth assignment if and only if I has a POM of size $m + 4n$.

For, suppose that f is a satisfying truth assignment for B. We form a matching M in I as follows. For each i $(1 \le i \le n)$, if $f(v_i)$=`true` then add the pairs (g_i^1, y_i^1), (g_i^1, y_i^2), $(g_i^2, \bar{x}_i^1)$, $(g_i^2, \bar{x}_i^2)$ to M. On the other hand if $f(v_i)$=`false` then add the pairs (g_i^1, x_i^1), (g_i^1, x_i^2), (g_i^2, y_i^1), (g_i^2, y_i^2) to M. For each j $(1 \le j \le m)$, at least one literal in c_j is true under f. Let s be the minimum integer such that the literal at position s of c_j is true under f. Course $x(c_j^s)$ is still unmatched by construction; add $(a_j, x(c_j^s))$ to M. It may be verified that M is a POM of size $m + 4n$ in I.

Conversely suppose that M is a POM in I of size $m + 4n$. Then the cardinality of M implies that every applicant is full in M. We firstly show that, for each i $(1 \le i \le n)$, either $\{(g_i^1, y_i^1), (g_i^1, y_i^2)\} \subseteq M$ or $\{(g_i^2, y_i^1), (g_i^2, y_i^2)\} \subseteq M$. Suppose this is not the case. As a consequence of the prerequisites, if $(g_i^r, y_i^1) \in M$ for some i $(1 \le i \le n)$ and $r \in \{1, 2\}$, then $(g_i^r, y_i^2) \in M$. Suppose now that $(g_i^r, y_i^2) \in M$ for some $r \in \{1, 2\}$, but $(g_i^r, y_i^1) \notin M$. Let M' be the matching obtained from M by removing any assignee of g_i^r worse than y_i^2 (if such an assignee exists) and adding (g_i^r, y_i^1) to M. Then M' dominates M, a contradiction. Now suppose that y_i^2 is unmatched in M. Let M' be the matching obtained from M by removing any assignee of g_i^1 worse than y_i^2 (if such an assignee exists) and adding (g_i^1, y_i^r) to M $(r \in \{1, 2\})$. Then M' dominates M, a contradiction. Thus the claim is established. It follows that for each i $(1 \le i \le n)$, either g_i^1 is matched in M to two members of X and g_i^2 is matched in M to two members of Y, or vice versa.

Now create a truth assignment f in B as follows. For each i $(1 \le i \le n)$, if $(g_i^1, y_i^1) \in M$, set $f(v_i)$=`true`, otherwise set $f(v_i)$=`false`. We claim that f is a satisfying truth assignment for B. For, let j $(1 \le j \le m)$ be given. Then $(a_j, x(c_j^s)) \in M$ for some s $(1 \le s \le 3)$. If $x(c_j^s) = x_i^r$ for some i $(1 \le i \le n)$ and r $(r \in \{1, 2\})$ then $f(v_i) =$ `true` by construction. Similarly if $x(c_j^s) = \bar{x}_i^r$ for some $(1 \le i \le n)$ and r $(r \in \{1, 2\})$ then $f(v_i) =$ `false` by construction. Hence f satisfies B. □

The next example shows that the difference in cardinalities between POMs may be arbitrary, and that SM-CAPR is not in general a constant-factor approximation algorithm for MAX POM CAPR.

Example 15. Consider a CAPR instance I in which $A = \{a_1, a_2\}$ and $C = \{c_1, c_2, \ldots, c_n\}$ for some $n \ge 1$. Let the preferences of the applicants be

$$P(a_1): c_1, c_2, \ldots, c_n \qquad P(a_2): c_n$$

and let $c_i \to c_{i+1}$ for each applicant $(1 \le i \le n - 1)$. Assume that a_1 has capacity n, whilst the capacity of a_2 and the capacity of every course is 1.

There are two POMs in I: if SM-CAPR is executed relative to a policy in which a_1 is first then we obtain the POM M_1 that assigns all the n courses to a_1 and nothing to a_2; if instead a_2 is first, we obtain the POM M_2 that assigns nothing to a_1 and the single course c_n to a_2. Hence executing SM-CAPR relative to different policies can give rise to POMs with arbitrarily large difference in cardinality. It follows that SM-CAPR is not in general a constant-factor approximation algorithm for MAX POM CAPR. However, notice that in this example the cardinality of the set of courses that are reachable from each course is not bounded by a constant; enforcing such a condition could improve the approximation possibilities. □

Algorithm 2 Modified lists for CACR

```
Require: CACR instance $I$
Ensure: return modified preference lists containing supercourses
1: for each applicant $a_i \in A$ do
2:     for each connected component (supercourse) $C^k_{a_i}$ in $D_{a_i}$ do
3:         available($C^k_{a_i}$):=true;
4:     for each course $c_j$ in $a_i$'s preference list do          ▷ in preference order
5:         if available($C(c_j)$) then
6:             add $C(c_j)$ to the modified preference list of $a_i$;
7:             available($C(c_j)$):=false;
```

6. Symmetric prerequisites — corequisites

In this section we focus on CACR, the extension of CA involving corequisite courses. Let us remind the reader that applicant a_i's prerequisite constraints are in the form of corequisites if the digraph D_{a_i} is *symmetric*, i.e., $(c_j, c_k) \in E_{a_i}$ holds if and only if $(c_k, c_j) \in E_{a_i}$ for any two courses $c_j, c_k \in C$. In this case we can think of D_{a_i} as an undirected graph. The graph D_{a_i} is a collection of connected components that partition the set of courses into equivalence classes $C^1_{a_i}, C^2_{a_i}, \ldots, C^r_{a_i}$ (later to be referred to as *supercourses*), and a_i can subscribe either to all the courses in one equivalence class or to none. The equivalence class containing course c_j will be denoted by $C(c_j)$.

Algorithm SM-CAPR finds a POM when preferences are lexicographic for general prerequisite relations, hence it can also be used when the prerequisites of applicants are in the form of corequisites. However, if this is the case, we can achieve a better computational complexity with a specialised algorithm.

The main idea of Algorithm SM-CACR is to work on supercourses. Each applicant's preference list is replaced by the *modified* preference list containing supercourses. As we assume lexicographic preferences, each supercourse $C^k_{a_i}$ occupies the position of the most preferred course $c_j \in C^k_{a_i}$ in the original preference list of applicant a_i.

In the preprocessing phase, for each applicant a_i we search the graph D_{a_i} to create its connected components (supercourses) and to find their sizes, where $s(C_j)$ denotes the size of supercourse C_j. By standard graph traversal techniques this requires $O(|E_{a_i}|)$ steps for applicant a_i, altogether $O(n_1|E|)$ steps, where $|E| = \max_{a_i \in A}|E_{a_i}|$. If the corequisites are not applicant specific, i.e., we are given an instance of CAUCR, we only need $O(|E|)$ steps.

Then we create the modified preference lists containing supercourses; this is accomplished by Algorithm 2. Each supercourse $C^k_{a_i}$ is initially *available*, indicating that it has not already been processed, which is represented by setting variable available($C^k_{a_i}$) to be `true`. We then scan a_i's preference list in preference order. For each course c_j in a_i's list, if $C(c_j)$ is available we add $C(c_j)$ to a_i's modified preference list and set variable available($C(c_j)$) to be `false`. Algorithm 2 needs $O(n_1 n_2 + L)$ steps altogether.

The sequential mechanism SM-CACR for CACR can be executed by working on the modified preference profile as follows. The mechanism works according to a given policy σ in stages. In one stage, the applicant a_i who next has her turn according to σ chooses the most-preferred supercourse C_j in her modified preference list to which she has not yet applied. Applicant a_i is assigned to C_j (and subsequently to all the courses c_k with $C(c_k) = C_j$) if two conditions are fulfilled: (i) the number of courses assigned to a_i so far plus the cardinality of C_j does not exceed $q(a_i)$, and (ii) each course $c_k \in C_j$ still has a free slot. If this is not possible, at the same stage a_i applies to her next supercourse until she is either assigned some supercourse or her preference list is exhausted. Moving down the preference lists can be performed using pointers similarly as in SM-CAPR. This mechanism is described in detail in Algorithm 3.

Algorithm 3 SM-CACR

```
Require: CACR instance $I$ and a policy $\sigma$
Ensure: return $M$, a POM in $I$
1:  $M := \emptyset$;
2:  for each applicant $a_i \in \sigma$ in turn do
3:      feasible := false;
4:      while $a_i$ has not exhausted her modified preference list and not feasible do
5:          $C_j$ := next supercourse in $a_i$'s modified list;
6:          feasible := true;
7:          if $|M(a_i)| + s(C_j) \leq q(a_i)$ then
8:              for each $c_k \in C_j$ do
9:                  if $c_k$ is full then
10:                     feasible := false;
11:             if feasible then
12:                 for each $c_k \in C_j$ do
13:                     $M := M \cup \{(a_i, c_k)\}$;
14:         else
15:             feasible := false;
16: return $M$;
```

Theorem 16. *Algorithm 3 correctly computes a POM in an instance of* CACR *in $O(N + n_1(n_2 + |E|))$ steps, where N is the total capacity of all courses, n_1 and n_2 are the numbers of applicants and courses respectively, and $|E| = \max_{a_i \in A} |E_{a_i}|$.*

Proof. Feasibility and Pareto optimality of the obtained matching can be proved in a similar fashion to the proof of Theorem 6.

To derive the complexity bound, recall that for the pre-processing we need $O(n_1|E|)$ steps and for Algorithm 2 we need $O(n_1 n_2 + L)$ steps, as already mentioned. Algorithm 3 uses $O(N)$ steps for the **for** loop, the **while** loop has $O(L)$ iterations. As supercourses partition the set of courses into disjoint subsets, each course $c_k \in C$ will be treated at most once for each applicant during the whole algorithm. So the lines 5–14 taken over the entire execution of the algorithm need at most $O(n_1 n_2)$ steps. This gives altogether the complexity bound $O(N + n_1(n_2 + |E|))$. $\square$

In the CAUCR model, where corequisites are uniform, the complexity bound of Algorithm 3 improves to $O(N + |E| + n_1 n_2)$, as all the digraphs D_{a_i} are the same. Moreover, after the modification described prior to Theorem 16 in which courses are merged into common supercourses, CAUCR becomes closely related to the matching problem with sizes considered in [25] and it is equivalent to CAP, the extension of CA with price-budget constraints described in [15]. For an instance I of CAP, it is known that for each POM M in I, there exists a policy σ such that executing SM-CACR relative to σ produces M [15, Theorem 3] and also a characterisation is known that gives rise to a polynomial-time algorithm to determine whether a given matching is Pareto optimal. However, if the corequisites are applicant-specific, neither is true.

Example 17. Here we show that, in case of applicant-specific corequisites, there exist POMs that the mechanism SM-CACR is not capable of reaching under any policy. This example is very similar to Example 9. We have again two applicants a_1, a_2 and three courses c_1, c_2, c_3, each applicant has capacity 2 and each course has capacity 1. Applicants have the following preference lists:

$$\begin{array}{ll} P(a_1): & c_1, c_2, c_3 \\ P(a_2): & c_2, c_1, c_3 \end{array}$$

and their corequisites are

$$c_1 \leftrightarrow_{a_1} c_3 \qquad c_2 \leftrightarrow_{a_2} c_3.$$

The modified preference lists for this instance are

$$\begin{array}{ll} P(a_1): & C_1^{a_1}, C_2^{a_1} \\ P(a_2): & C_2^{a_2}, C_1^{a_2} \end{array}$$

where $C_1^{a_1} = \{c_1, c_3\}, C_2^{a_1} = \{c_2\}$ and $C_2^{a_2} = \{c_2, c_3\}, C_1^{a_2} = \{c_1\}$. SM-CACR returns the matching $M_i = \{(a_i, c_i), (a_i, c_3)\}$ if the first applicant in the policy is a_i ($i \in \{1, 2\}$). However the matching $M = \{(a_1, c_2), (a_2, c_1)\}$ is also Pareto optimal and cannot be obtained by SM-CACR. □

For applicant-specific corequisites we even have the following intractability result.

Theorem 18. DOM CACR *with applicant-specific corequisites is NP-complete even if each course has capacity 1.*

Proof. It is easy to check that it suffices to make the prerequisite constraints defined in the proof of Theorem 11 bi-directional and the rest of the arguments remain valid. □

By contrast, finding a POM with maximum cardinality is hard even for uniform corequisites. Let us mention here that the intractability of this problem is also implied by Theorem 7 of [15], using the connection between CACR and the price-budget case of the many-to-many matchings, but the following theorem is stronger as it provides tighter bounds on the sizes of supercourses and the capacities of applicants.

Theorem 19. MAX CAUCR-D *and* MAX POM CAUCR-D *are NP-complete even if each applicant has capacity at most 2, each course has capacity 1 and each equivalence class has cardinality at most 2.*

Proof. Now it suffices to replace the prerequisite $y_i^1 \to y_i^2$ by $y_i^1 \leftrightarrow y_i^2$ in the proof of Theorem 14. The argument in the penultimate paragraph of the proof of that theorem needed Pareto optimality of the given matching M to ensure that each applicant is either assigned both courses y_i^1 and y_i^2 or none, but this is now ensured by corequisites. □

We now strengthen this result by showing that MAX POM CACR is very difficult to approximate.

Theorem 20. MAX POM CACR *is NP-hard and not approximable in polynomial time within a factor of $N^{1-\varepsilon}$, for any $\varepsilon > 0$, unless P=NP, where N is the total capacity of the applicants, even in the case of uniform corequisites and if each course has capacity 1.*

Proof. Let $\varepsilon > 0$ be given. Let B be an instance of (2,2) -E3 -SAT, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of variables and $C = \{c_1, c_2, \dots, c_m\}$ is the set of clauses. Let $\beta = \lceil \frac{2}{\varepsilon} \rceil$ and let $\alpha = n^\beta$.

We form an instance I of CACR as follows. Let $X \cup Y \cup Z$ be the set of courses, where $X = \{x_i^1, x_i^2, \bar{x}_i^1, \bar{x}_i^2 : 1 \le i \le n\}$, $Y = \{y_i^1, y_i^2 : 1 \le i \le n\}$ and $Z = \{z_1, z_2, \dots, z_D\}$, where $D = 6n(\alpha - 1) + 1$. The courses in X correspond to the first and second occurrences of v_i and $\bar{v}_i$ in B for each i ($1 \le i \le n$). Let $A \cup G \cup \{b, h\}$ be the set of applicants, where $A = \{a_j : 1 \le j \le m\}$ and $G = \{g_i^1, g_i^2 : 1 \le i \le n\}$. Each course has capacity 1. Each applicant in A has capacity 1, each applicant in G has capacity 2, h has capacity $2n - m$ and b has capacity D.

For each i $(1 \le i \le n)$, courses y_i^1 and y_i^2 are corequisites. Also all the courses in Z are corequisites. For each j $(1 \le j \le m)$ and for each s $(1 \le s \le 3)$, $x(c_j^s)$ is as defined in the proof of Theorem 14. The preference lists of the applicants are as follows:

$$\begin{array}{rll} P(a_j): & x(c_j^1), x(c_j^2), x(c_j^3) & (1 \le j \le m) \\ P(g_i^1): & y_i^1, y_i^2, x_i^1, x_i^2 & (1 \le i \le n) \\ P(g_i^2): & y_i^1, y_i^2, \bar{x}_i^1, \bar{x}_i^2 & (1 \le i \le n) \\ P(h): & [X] & \\ P(b): & [X], [Z] & \end{array}$$

In the preference lists of h and b, the symbols $[X]$ and $[Z]$ denote all members of X and Z listed in arbitrary strict order, respectively. In I the total capacity of the applicants, denoted by N, satisfies $N = D + 6n$. We claim that if B has a satisfying truth assignment then I has a POM of size $D + 6n$, whilst if B does not have a satisfying truth assignment then any POM in I has size at most $6n$.

For, suppose that f is a satisfying truth assignment for B. We form a matching M in I as follows. For each i $(1 \le i \le n)$, if $f(v_i)$=`true` then add the pairs (g_i^1, y_i^1), (g_i^1, y_i^2), $(g_i^2, \bar{x}_i^1)$, $(g_i^2, \bar{x}_i^2)$ to M. On the other hand if $f(v_i)$=`false` then add the pairs (g_i^1, x_i^1), (g_i^1, x_i^2), (g_i^2, y_i^1), (g_i^2, y_i^2) to M. For each j $(1 \le j \le m)$, at least one literal in c_j is true under f. Let s be the minimum integer such that the literal at position s of c_j is true under f. Course $x(c_j^s)$ is still unmatched by construction; add $(a_j, x(c_j^s))$ to M. There remain $2n - m$ courses in X that are as yet unmatched in M; assign all these courses to h. Finally assign all courses in Z to b in M. It may be verified that M is a POM of size $D + 6n$ in I.

Now suppose that f admits no satisfying truth assignment. Let M be any POM in I. We will show that $|M| \le 6n$. Suppose not. Then $|M| > 6n$ and the only way this is possible is if at least one course in Z is matched in M. But only b can be assigned members of Z in M, and since all pairs of courses in Z are corequisites, it follows that $M(b) = Z$.

We next show that, for each i $(1 \le i \le n)$, either $\{(g_i^1, y_i^1), (g_i^1, y_i^2)\} \subseteq M$ or $\{(g_i^2, y_i^1), (g_i^2, y_i^2)\} \subseteq M$. Suppose this is not the case for some i $(1 \le i \le n)$. As a consequence of the corequisite restrictions on courses in Y, y_i^1 and y_i^2 are unmatched in M. Let M' be the matching obtained from M by deleting any assignee of g_i^1 worse than y_i^2 (if such an assignee exists) and by adding (g_i^1, y_i^1) and (g_i^1, y_i^2) to M. Then M' dominates M, a contradiction.

We claim that each course in X is matched in M. For, suppose that some course $x \in X$ is unmatched. Then let M' be the matching obtained from M by unassigning b from all courses in Z, and by assigning b to x. Then M' dominates M, a contradiction.

It follows that every course in $X \cup Y$ is matched in M. Since $|X \cup Y| = 6n$ and the applicants in $A \cup G \cup \{h\}$ have total capacity $6n$, every applicant in $A \cup G \cup \{h\}$ is full.

Create a truth assignment f in B as follows. For each i $(1 \le i \le n)$, if $(g_i^1, y_i^1) \in M$, set $f(v_i)$=`true`, otherwise set $f(v_i)$=`false`. We claim that f is a satisfying truth assignment for B. For, let j $(1 \le j \le m)$ be given. Then $(a_j, x(c_j^s)) \in M$ for some s $(1 \le s \le 3)$. If $x(c_j^s) = x_i^r$ for some i $(1 \le i \le n)$ and r $(r \in \{1, 2\})$ then $f(v_i)$=`true` by construction. Similarly if $x(c_j^s) = \bar{x}_i^r$ for some $(1 \le i \le n)$ and r $(r \in \{1, 2\})$ then $f(v_i)$=`false` by construction. Hence f satisfies B, a contradiction.

Thus if B is satisfiable then I admits a POM of size $D + 6n = 6n(\alpha - 1) + 1 + 6n > 6n\alpha$. If B is not satisfiable then any POM in I has size at most $6n$. Hence an α-approximation algorithm for MAX POM CACR implies a polynomial-time algorithm to determine whether B is satisfiable, a contradiction to the NP-completeness of (2,2) -E3 -SAT.

It remains to show that $N^{1-\varepsilon} \le \alpha$. On the one hand, $N = 6n + D = 6n\alpha + 1 \le 7n\alpha = 7n^{\beta+1}$. Hence $n^\beta \ge N^{\frac{\beta}{\beta+1}} 7^{-\frac{\beta}{\beta+1}}$. On the other hand, $N = 6n\alpha + 1 \ge \alpha = n^\beta \ge 7^\beta$ as we may assume, without loss of generality, that $n \ge 7$. It follows that $7^{-\frac{\beta}{\beta+1}} \ge N^{-\frac{1}{\beta+1}}$. Thus

$$\alpha = n^\beta \ge N^{\frac{\beta}{\beta+1}} 7^{-\frac{\beta}{\beta+1}} \ge N^{\frac{\beta}{\beta+1}} N^{-\frac{1}{\beta+1}} = N^{\frac{\beta-1}{\beta+1}} = N^{1-\frac{2}{\beta+1}} \ge N^{1-\varepsilon}. \quad \square$$

7. Alternative prerequisites

In this section we turn our attention to CAAPR, the analogue of CAPR in which prerequisites need not be compulsory but may be presented as alternatives. We will show that, in contrast to the case for CAPR, finding a POM is NP-hard, under either additive or lexicographic preferences.

Recall that as CAPR is a special case of CAAPR, Lemma 2 implies that finding a most-preferred bundle of courses under additive preferences is NP-hard. Now we prove a similar result for lexicographic preferences. Notice that it might be tempting to extend SM-CAPR to the case with alternative prerequisites: one might think that when applicant a_i is considering course c_j it suffices to choose the most-preferred course from the set of its alternative prerequisites. However, this might lead to the necessity to add too many courses to the assignment of a_i, thereby exceeding her capacity. By closely following the construction in the proof of the following lemma one can deduce where the difficulties may arise.

Lemma 21. *The problem of finding the most-preferred feasible bundle of courses of a given applicant with lexicographic preferences in* CAAPR *is NP-hard.*

Proof. We reduce from VC-D, the decision version of VC, which is the problem of finding a vertex cover of minimum size in a given graph. VC-D is NP-complete [30]. Let $\langle G, K\rangle$ be an instance of VC-D, where $G = (V, E)$ is a graph and K is a positive integer. Assume that $V = \{v_1, v_2, \dots, v_n\}$ and $E = \{e_1, e_2, \dots, e_m\}$. We construct an instance I of CAAPR as follows. Let the set of courses be $V \cup E \cup \{b\}$ (again, we use the same notation for vertices and edges in G as we do for courses in I, but no ambiguity should arise.) There is a single applicant a_1 with capacity $m + K + 1$ whose preference list is as follows:

$$P(a_1) : b, e_1, e_2, \dots, e_m, v_1, v_2, \dots, v_n.$$

Course b has a single compulsory prerequisite course e_1, and each course e_j $(2 \le j \le m - 1)$ has a single compulsory prerequisite course e_{j+1}. Moreover, all the E-courses have (alternative) prerequisites; namely, for any j $(1 \le j \le m)$, if course e_j corresponds to the edge $e_j = \{v_i, v_k\}$ then $e_j \mapsto_{a_1} \{v_i, v_k\}$. We claim that G admits a vertex cover of size at most K if and only if I admits a matching in which a_1 is assigned course b.

For, suppose that G admits a vertex cover S where $|S| \le K$. Form a matching M by assigning a_1 to the bundle $B = \{b\} \cup E \cup S$. Then B is a feasible bundle of courses for a_1, and $b \in B$.

Conversely, suppose I admits a matching M in which a_1 is assigned a bundle B containing course b. Then, due to the prerequisites, B must contain all E-courses and for each course in $e_j \in E$, B must contain some course in $v_i \in V$ that corresponds to a vertex incident to e_j. Let $S = B \cap V$. Clearly S is a vertex cover in G, and as $q(a_1) = m + K + 1$, it follows that $|S| \le K$.

A polynomial-time algorithm for finding the most-preferred bundle of courses for a_1 can then be used to decide whether I admits a matching in which a_1 is assigned b, hence the result. □

As noted in Section 3, in the case of just one applicant a_1, a matching M is a POM if and only if a_1 is assigned in M her most-preferred feasible bundle of courses. Hence Lemma 21 implies the following assertion.

Theorem 22. *Given an instance of* CAAPR *the problem of finding a POM is NP-hard even if there is only a single applicant. The result holds under either additive or lexicographic preferences.*

We finally remark that, since CAPR is a special case of CAAPR, Theorem 11 implies that the problem of determining whether a given matching M in an instance of CAAPR is a POM is co-NP-complete for lexicographic preferences (and also for additive preferences by the remark following Theorem 11).

8. Open problems and directions for future research

We would like to conclude with several open problems and directions for future research.

1. **Refining the boundary between efficiently solvable and hard problems.** In the proofs of the NP-hardness and inapproximability results in this paper we had some applicants whose preference lists were not complete, whose capacity was not bounded by a constant, and/or whose prerequisite structure was of unbounded depth. Will the hardness results still be valid if there are no unacceptable courses, all capacities are bounded, and prerequisites are more structured (e.g., by an upper bound on the number of prerequisites per course or in total)? These problems also call for a more detailed multivariate complexity analysis. It might be interesting to determine whether restricting some other parameters, e.g., the lengths of preference lists, may make the problems tractable. Another interesting question to investigate is the approximability of the problem of finding the maximum size of a POM in the case of acyclic prerequisites.
2. **Indifferences in the preference lists.** In this paper, we assumed that all the preferences are strict. If preference lists contain ties, sequential mechanisms have to be carefully modified to ensure Pareto optimality. Polynomial-time algorithms for finding a Pareto optimal matching in the presence of ties have been given in the context of HAT (the extension of HA where preference lists may include ties) by Krysta et al. [31] and in its many-to-many generalisation CAT (the extension of CA where preference lists may include ties) by Cechlárová et al. [32]. However as far as we are aware, it remains open to extend these algorithms to the cases of CAPR and CACR where preference lists may include ties.
3. **Strategic issues.** By a standard argument, one can ensure that the sequential mechanism that lets each applicant on her turn choose her entire most-preferred bundle of courses (i.e., the serial dictatorship mechanism) is strategy-proof even in the case of prerequisites. However, serial dictatorship may be very unfair, as the first dictator may grab all the courses and leave nothing for the rest of the applicants. Let us draw the reader's attention to several economic papers that highlight the special position of serial dictatorship among the mechanisms for allocation of multiple indivisible goods: serial dictatorship is the only allocation rule that is Pareto efficient, strategy-proof and fulfils some additional properties, namely non-bossiness and citizen sovereignty [33], and population monotonicity or consistency [34]. We were not able to obtain a similar characterisation of serial dictatorship for CAPR.

   As far as the general sequential mechanism is concerned, a recent result by Hosseini and Larson [35] shows that no sequential mechanism that allows interleaving policies (i.e., in which an a_i is allowed to pick courses more than once, and between two turns of a_i another applicant has the right to pick a course) is strategy-proof, even in the simpler case without any prerequisites. It immediately follows that SM-CAPR is not strategy-proof. However, it is not known whether a successful manipulation can be computed efficiently. Further, we have shown that in CAPR, not all POMs can be obtained by a sequential mechanism. We leave it as an open question whether a strategy-proof and Pareto optimal mechanism other than serial dictatorship exists in CAPR.

Acknowledgements

The authors would like to convey their gratitude to the reviewers for very detailed comments which have helped to improve the presentation of this paper.

References

[1] C. Boutilier, B. Dorn, N. Maudet, V. Merlin, Computational Social Choice: Theory and Applications (Dagstuhl Seminar 15241), Dagstuhl Reports, vol. 5(6), 2016, pp. 1–27.
[2] M. Balinski, T. Sönmez, A tale of two mechanisms: student placement, J. Econom. Theory 84 (1) (1999) 73–94.
[3] A.S. Kelso Jr., V.P. Crawford, Job matching, coalition formation and gross substitutes, Econometrica 50 (1982) 1483–1504.
[4] D. Monte, N. Tumennasan, Matching with quorums, Econom. Lett. 120 (2013) 14–17.
[5] A. Abdulkadiroğlu, T. Sönmez, Random serial dictatorship and the core from random endowments in house allocation problems, Econometrica 66 (3) (1998) 689–701.
[6] E. Budish, E. Cantillon, The multi-unit assignment problem: Theory and evidence from course allocation at Harvard, Amer. Econ. Rev. 102 (5) (2012) 2237–2271.
[7] D.A. Kohler, R. Chandrasekaran, A class of sequential games, Oper. Res. 19 (2) (1971) 270–277.
[8] S.J. Brams, D.L. King, Efficient fair division: Help the worst off or avoid envy? Rationality Soc. 17 (4) (2005) 387–421.
[9] S. Bouveret, J. Lang, Elicitation-free protocol for allocating indivisible goods, in: Proceedings of IJCAI 2011: The 22nd International Joint Conference on Artificial Intelligence, AAAI Press, 2011, pp. 73–78.
[10] H. Aziz, T. Walsh, L. Xia, Possible and necessary allocations via sequential mechanisms, in: Proceedings of IJCAI 2015: The 24th International Joint Conference on Artificial Intelligence, AAAI Press, 2015, pp. 468–474.
[11] M. Manea, Serial dictatorship and Pareto optimality, Games Econom. Behav. 61 (2007) 316–330.
[12] L.-G. Svensson, Queue allocation of indivisible goods, Soc. Choice Welf. 11 (4) (1994) 323–330.
[13] D.J. Abraham, K. Cechlárová, D.F. Manlove, K. Mehlhorn, Pareto optimality in house allocation problems, in: Proceedings of ISAAC '04: The 15th Annual International Symposium on Algorithms and Computation, in: Lecture Notes in Computer Science, vol. 3341, Springer, 2004, pp. 3–15.
[14] F. Diebold, H. Aziz, M. Bichler, F. Matthes, A. Schneider, Course Allocation via Stable Matching, Business Inform. Syst. Eng. 6 (2) (2014) 97–110.
[15] K. Cechlárová, P. Eirinakis, T. Fleiner, D. Magos, I. Mourtos, E. Potpinková, Pareto optimality in many-to-many matching problems, Discrete Optim. 14 (2014) 160–169.
[16] A.E. Roth, The college admissions problem is not equivalent to the marriage problem, J. Econom. Theory 36 (1985) 277–288.
[17] S. Barberà, W. Bossert, P.K. Pattanaik, in: S. Barberà, P.J. Hammond, C. Seidl (Eds.), Ranking Sets of Objects, in: Handbook of Utility Theory, vol. 2, Kluwer Academic Publishers, 2004, pp. 893–977.
[18] P.C. Fishburn, Axioms for lexicographic preferences, Rev. Econom. Stud. 42 (3) (1975) 415–419.
[19] L.J. Schulman, V.V. Vazirani, (2012) Allocation of divisible goods under lexicographic preferences. Technical Report 1206.4366, Computing Research Repository, Cornell University Library, 2012. Available from arXiv:1206.4366.
[20] D. Saban, J. Sethuraman, A note on object allocation under lexicographic preferences, J. Math. Econom. 50 (2014) 283–289.
[21] T. Todo, H. Sun, M. Yokoo, Strategyproof exchange with multiple private endowments, in: Proceedings of AAAI 2014: The 28th AAAI Conference on Artificial Intelligence, 2014, pp. 805–811.
[22] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints, Oper. Res. 26 (1) (1978) 22–35.
[23] J.T. Guerin, J. Hanna, L. Ferland, N. Mattei, J. Goldsmith, The academic advising planning domain, in: Proceedings of WS-IPC-12: The 3rd Workshop on the International Planning Competition, 2012, pp. 1–5.
[24] T. Dodson, N. Mattei, J.T. Guerin, J. Goldsmith, An English-Language Argumentation Interface for Explanation Generation with Markov Decision Processes in the Domain of Academic Advising, ACM Trans. Interactive Intell. Syst. 3 (3) (2013) 18.
[25] P. Biró, E. McDermid, Matching with sizes (or scheduling with processing set restrictions), Discrete Appl. Math. 164 (1) (2014) 61–67.
[26] R.M. Karp, Reducibility Among Combinatorial Problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85–103.
[27] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.
[28] D. Maier, J.A. Storer, A note on the complexity of the superstring problem, Princeton University, Department of Electrical Engineering and Computer Science, Princeton, NJ, 1977.
[29] P. Berman, M. Karpinski, Alexander D. Scott, Approximation hardness of short symmetric instances of MAX-3SAT, Electronic Colloquium on Computational Complexity Report, number 49, 2003.
[30] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, San Francisco, CA, 1979.
[31] P. Krysta, D.F. Manlove, B. Rastegari, J. Zhang, Size versus truthfulness in the House Allocation problem, in: Proceedings of EC 2014: The 15th ACM Conference on Economics and Computation, ACM, 2014, pp. 453–470.
[32] K. Cechlárová, P. Eirinakis, T. Fleiner, D. Magos, D.F. Manlove, I. Mourtos, E. Ocel'áková, B. Rastegari, Pareto optimal matchings in many-to-many markets with ties, Theory Comput. Syst. 59 (4) (2016) 700–721.
[33] S. Pápai, Strategyproof and nonbossy multiple assignments, J. Public Econ. Theory 3 (3) (2001) 257–271.
[34] B. Klaus, E. Miyagawa, Strategy-proofness, solidarity, and consistency for multiple assignment problems, Internat. J. Game Theory 30 (2001) 421–435.
[35] H. Hosseini, K. Larson, (2015) Strategyproof quota mechanisms for multiple assignment problems. Technical Report 1507.07064, Computing Research Repository, Cornell University Library, 2015. Available from arXiv:1507.07064.