Stable Hypergraph Matching in Unimodular Hypergraphs

Péter Biró[1], Gergely Csáji[1], and Ildikó Schlotter[1]

[1] HUN-REN Centre for Economic and Regional Studies, Budapest, Hungary
{biro.peter, csaji.gergely, schlotter.ildiko}@krtk.hun-ren.hu

Abstract

We study the NP-hard Stable Hypergraph Matching (SHM) problem and its generalization allowing capacities, the Stable Hypergraph b-Matching (SHbM) problem, and investigate their computational properties under various structural constraints. Our study is motivated by the fact that Scarf's Lemma [23] together with a result of Lovász [17] guarantees the existence of a stable matching whenever the underlying hypergraph is normal. Furthermore, if the hypergraph is unimodular (i.e., its incidence matrix is totally unimodular), then even a stable b-matching is guaranteed to exist. However, no polynomial-time algorithm is known for finding a stable matching or b-matching in unimodular hypergraphs.

We identify subclasses of unimodular hypergraphs where SHM and SHbM are tractable such as laminar hypergraphs or so-called subpath hypergraphs with bounded-size hyperedges; for the latter case, even a maximum-weight stable b-matching can be found efficiently. We complement our algorithms by showing that optimizing over stable matchings is NP-hard even in laminar hypergraphs. As a practically important special case of SHbM for unimodular hypergraphs, we investigate a tripartite stable matching problem with students, schools, and companies as agents, called the University Dual Admission problem, which models real-world scenarios in higher education admissions.

Finally, we examine a superclass of subpath hypergraphs that are normal but not necessarily unimodular, namely subtree hypergraphs where hyperedges correspond to subtrees of a tree. We establish that for such hypergraphs, stable matchings can be found in polynomial time but, in the setting with capacities, finding a stable b-matching is NP-hard.

1 Introduction

Stable matchings are fundamental in economics, combinatorial optimization, and mechanism design, playing a crucial role in applications such as college admissions, job markets, and organ exchange programs. Since the seminal work of Gale and Shapley [14], stable matchings have been extensively studied in two-sided markets where agents form pairwise relationships. However, many real-world scenarios require interactions beyond pairwise relationships, leading naturally to hypergraph-based formulations.

The focus of our study is the Stable Hypergraph b-Matching (SHbM) problem, where we are given a hypergraph whose vertices represent agents, with each agent v having preferences over the incident hyperedges and, additionally, a capacity value $b(v)$. The task is to find a stable b-matching, i.e., a set M of hyperedges where each agent v is adjacent to at most $b(v)$ hyperedges in M, and no hyperedge e outside M is "desirable" for all agents in e; see Section 2 for the precise definition of stability.

In the context of hypergraph matchings, stability is often difficult to achieve, as stable solutions may not always exist. On the positive side, Scarf's Lemma [23] guarantees the existence of a fractional stable solution in the SHbM problem. Furthermore, a famous result of Lovász [17] in combination with Scarf's Lemma ensures that in an instance of Stable Hypergraph Matching (SHM)—the restriction of SHbM without capacities—with a *normal hypergraph*, a stable matching always exists. An even stronger result follows for *unimodular hypergraphs*, where the incidence matrix is totally unimodular: Scarf's Lemma guarantees the existence of a stable b-matching. However, these results only provide existential guarantees, leaving computational aspects largely unexplored.

Motivated by these structural insights, we investigate the computational complexity of finding stable (b-)matchings in different hypergraph classes. Furthermore, we consider not only the problem of finding *any* stable (b-)matching but also of computing maximum-weight stable solutions.

	Matching		b-Matching	
	Stable	Max-Weight Stable	Stable	Max-Weight Stable
UDA	**P**[R3.9]	**NPh**, **XP**(n_U)[T3.5,T3.6]	**?**, **XP**(n_U)[T3.6]	**NPh**, **XP**(n_U)[T3.5,T3.6]
Laminar	**P** [8]	**NPh**, **XP**(ℓ_{max}) [T4.3,T4.4]	**P** [T4.1]	**NPh**, **XP**(ℓ_{max}) [T4.3,T4.4]
Subpath	**P** [8]	**NPh**, **XP**(ℓ_{max}) [T4.3,T4.4]	**?**, **XP**(ℓ_{max}) [T4.4]	**W[1]h**, **XP**(ℓ_{max}) [T4.10,T4.4]
Subtree	**P**[T5.1]	**NPh**[T4.3]	**NPh**[T5.2]	**NPh**[T4.3]

Table 1: Computational complexity of finding an arbitrary or a maximum-weight stable (b-)matching in different hypergraph classes. **NPh**, **P**, **XP**, and **W[1]h** refers to NP-hardness, polynomial-time solvability, solvability in XP, and W[1]-hardness (the latter two with the given parameter), respectively. Parameter ℓ_{max} is the size of the largest hyperedge and n_U is the number of universities. Recall the hierarchy of hypergraphs which we may write informally as {laminar} ⊆ {subpath} ⊆ {subtree}.

1.1 Our Contribution

We begin with a class inspired by a real-world problem: the University Dual Admission (UDA) problem which arises in higher education admissions where students apply to universities and internship programs simultaneously [12]. Although we show that a stable solution always exists by reducing UDA to SHbM in a unimodular hypergraph, our proof does not directly lead to an efficient algorithm. though we are unable to settle the complexity of finding a stable solution in a general instance of UDA, we provide an algorithm that finds a maximum-weight stable solution and runs in XP time parameterized by the number of universities. To complement this, we prove that computing a maximum-size stable matching is NP-hard even without capacities, although we show that a stable matching can be found efficiently in this case. Additionally, we introduce a relaxation of stability, termed *half-stability*, and design a polynomial-time algorithm that always finds a half-stable matching.

Next, we investigate *laminar hypergraphs*, a class where hyperedges follow a hierarchical inclusion property. Leveraging their structure, we design a polynomial-time algorithm that finds a stable b-matching whenever the input hypergraph is laminar. By contrast, we also establish that, even in the setting without capacities, finding a maximum-weight stable matching (or even a stable matching containing a fixed edge) is NP-hard.

We extend our study to a superclass of laminar hypergraphs, *subpath hypergraphs*, where hyperedges can be represented as subpaths of a path. Subpath hypergraphs naturally model scenarios where relationships or interactions are constrained along a linear structure, such as transportation networks or supply chains. We develop an XP algorithm parameterized by the largest hyperedge size for finding a maximum-weight stable b-matching in such hypergraphs.

Finally, we consider a superclass of subpath hypergraphs, *subtree hypergraphs*, where hyperedges correspond to subtrees of an underlying tree. While these hypergraphs are not necessarily unimodular, they are still normal, which ensures the existence of a stable matching by the results of Lovász [17] and Scarf [23]. For the matching case without capacities, we provide a polynomial-time algorithm, which contrasts our result that computing a stable b-matching for general capacities is NP-hard.

Possible applications that motivate the study of SHbM for various hypergraph classes include problems related to budgeting in transportation networks, project selection management in companies, or coalition formation in politics; see Appendix A.

Our findings, summarized in Table 1, contribute to the understanding of stable matchings in structured hypergraphs, bridging theoretical guarantees with computational feasibility.

1.2 Related Work

In their seminal paper, Gale and Shapley [14] provided a model and an algorithmic solution for college admission problems. Their solution concept is called *stability*, which means that an application is fairly rejected by a college if the college filled its quota with better applicants. Gale and Shapley showed that their so-called deferred-acceptance (DA) algorithm always finds a stable solution that is optimal for the students if they are proposing in the algorithm.

The work of Gale and Shapley [14] inspired extensive research in mathematics, computer science, operations research, as well as in economics and game theory. For an overview, we recommend the reader to consult the book of Manlove [18] in computer science, the books of Roth and Sotomayor [22] and Haeringer [15] in economics. Besides the theoretical research, the DA algorithm has also been used in many important applications, firstly in the US resident allocation scheme (called NRMP) since 1952 [21] and also in several nationwide college admission and school choice programs.

One of our motivating problems originates in the Hungarian university scheme where it is possible to apply for a so-called *dual program* that consist of a normal university major together with an internship at a company. For a student to get accepted to such a dual program, both the university and the company have to grant admission, and these parties can have different rankings. Note that this feature is not integrated in the current admission system, as the hiring decisions of the companies are not shared with the central coordinators when conducting the matching algorithm, which causes serious coordination problems, see a detailed description of the problem by Fleiner et al. [12].

Rather independently from the topic of matching under preferences, Scarf [23] showed that so-called balanced NTU-games (a family of cooperative games with non-transferable utility) have non-empty core by providing a finite algorithm that computes a core solution in such games. Instead of introducing these game-theoretical notions, we only interpret and use Scarf's Lemma in the context of stable matchings and b-matchings in hypergraphs.

The Stable Hypergraph Matching problem, introduced by Aharoni and Fleiner [2], is essentially equivalent to the problem of finding a core solution in simple NTU-games, also known as *core stability in hedonic coalition formation games* [4, 25, 3]; in the hypergraph formulation of these problems, vertices represent agents and hyperedges represent the possible coalitions. The setting where agents have capacities, giving rise to the SHbM problem, was introduced by Biró and Fleiner [6] who proposed a general framework and proved the existence of fractional stable solutions using Scarf's Lemma for b-matching problems. They also highlighted that the matching polytope has the integer property for normal hypergraphs, therefore the solution determined by Scarf's algorithm for the SHM problem is always integer and thus corresponds to a stable matching.

However, the running time of computing a fractional solution by Scarf's algorithm is not polynomial in general. The PPAD-hardness of computing a fractional stable matching was proved by Kintali et al. [16] and was linked with several other PPAD-hard combinatorial problems. Recently Csáji extended these PPAD-hardness results for the fractional stable matching problem for three-partite hypergraphs and for the stable allocation with couples problem [9]. On the positive side, for certain stable matching problems it was shown that Scarf's algorithm terminates in polynomial time for some appropriately defined pivot rule. Very recently Faenza et al. [10] showed this property for the classical stable marriage problem, and Chandrasekaran et al. [8] for SHM for so-called arborescence hypergraphs (see Section 2 for the definition). Even though arborescence hypergraphs include subpath hypergraphs, the results by Chandrasekaran et al. [8] do not imply any of our results, as they involve neither capacities nor optimization over stable matchings.

Organization. We start by introducing the necessary notation and formally define our studied computational problems in Section 2. We proceed with the University Dual Admission (UDA) problem in Section 3 as a practical example of SHbM. We move on to more abstract unimodular hypergraph classes in Section 4: in Section 4.1 we study the class of laminar hypergraphs, while in Section 4.2 we focus on the more general class of subpath hypergraphs. We examine subtree hypergraphs in Section 5, and finally conclude in Section 6.

Results marked with ($\star$) have their proofs deferred to the appendices.

2 Preliminaries

In Section 2.1 we define the concepts related to graphs and hypergraphs that we need in our paper, and in Section 2.2 we provide the definition of the computational problems we will investigate.

2.1 Notation and Terminology

We use the notation $[\ell] = \{1, 2, \dots, \ell\}$ for each positive integer ℓ. We let $\mathbb{Z}, \mathbb{Z}_+, \mathbb{R}$, and $\mathbb{R}_+$ denote the set of integers, non-negative integers, reals, and non-negative reals, respectively. We use basic graph terminology with standard notation; see Appendix B.1 for all necessary definitions.

Matrices, polyhedra and Scarf's Lemma. For a matrix A, we let A_i denote its ith row. A matrix A is *totally unimodular* or *TU* if each subdeterminant of A has value in $\{-1, 0, 1\}$. Totally unimodular matrices are known for their useful properties. Most importantly, if A is TU and b is integral, then all extreme points of the polyhedron $\{x \in \mathbb{R}^m : Ax \leq b, x \geq 0\}$ are integral. A point z in the polyhedron $\{x \in \mathbb{R}^m : Ax \leq b, x \geq 0\}$ is an *extreme point* if it cannot be written as a strictly convex combination of two different points in the polyhedron. That is, there are no $\lambda \in (0,1)$ and $z_1, z_2 \in \{x \in \mathbb{R}^m : Ax \leq b, x \geq 0\}$ such that $z = \lambda z_1 + (1-\lambda)z_2$.

A matrix A is a *network matrix* if it can be obtained from a directed graph $G = (V, E)$ and an edge set $F \subseteq E$ of G for which (V, F) is a spanning tree of G in the undirected sense, via the following method. With each row of A we associate an edge $f \in F$, and with each column we associate an edge $e \in E \setminus F$. Given edges $f \in F$ and $e \in E \setminus F$, let $C_{e,f}$ denote the unique cycle (in the undirected sense) contained in $(V, F \cup \{e\})$. Then, the entry of A at the intersection of row f and column e has value

- 1, if f is contained in $C_{e,f}$, and the edges e and f have the same orientation along the cycle $C_{e,f}$;
- -1, if f is contained in $C_{e,f}$, and e and f have a different orientation along $C_{e,f}$, and
- 0, if f is not contained in the cycle $C_{e,f}$.

It is well known that network matrices are totally unimodular (see e.g., [24, Theorem 13.20]).

The following is a key lemma of Scarf [23], useful for many stable matching problems.

Lemma 2.1 (Scarf [23])**.** *Let $A \in \mathbb{R}_+^{n \times m}$ be a matrix such that every column of A has a nonzero element, and let $b \in \mathbb{R}_+^n$. Suppose that every row $i \in [n]$ has a strict ordering $\succ_i$ over those columns $j \in [m]$ for which $A_{ij} > 0$. Then there is an extreme point of $\{x \in \mathbb{R}^m : Ax \leq b,\ x \geq 0\}$ that dominates every column in some row, where we say that $x \in \mathbb{R}^m$* dominates *column j in row i if $A_{ij} > 0$, $A_i x = b_i$, and $k \succeq_i j$ for all $k \in [m]$ such that $A_{ik} x_k > 0$. Also, such an extreme point can be found by a finite algorithm.*

Hypergraphs, matchings and b-matchings. A *hypergraph* $\mathcal{H} = (V, \mathcal{E})$ contains a set V of vertices and a set $\mathcal{E} \subseteq 2^V$ of *hyperedges*; we may simply refer to hyperedges as *edges* when this causes no confusion. We will say that a hyperedge $e \in \mathcal{E}$ is *incident* to a vertex $v \in V$ if $v \in e$. The *incidence matrix* of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ has $|V|$ rows and $|\mathcal{E}|$ columns, and the entry at the intersection of column $v \in V$ and hyperedge $e \in \mathcal{E}$ is 1 if e is incident to v, and 0 otherwise. For a subset $F \subseteq \mathcal{E}$, let $F(v) := \{e \in F \mid v \in e\}$.

Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$, a *matching* is a subset $M \subseteq \mathcal{E}$ that satisfies that $|M(v)| \leq 1$ for every $v \in V$. A matching M leaves a vertex v *unmatched* if $M(v) = \emptyset$, otherwise it *covers* v. Given *capacities* $b(v) \in \mathbb{Z}_+$ for each $v \in V$, we say that $M \subseteq \mathcal{E}$ is a *b-matching* if $|M(v)| \leq b(v)$ for each $v \in V$. For a b-matching M, we say that v is *unsaturated* in M if $|M(v)| < b(v)$, *saturated* if $|M(v)| = b(v)$, and *oversaturated* if $|M(v)| > b(v)$.

Hypergraph classes. Let us now describe the hypergraph classes relevant to our paper.

Let $\mathcal{H}$ be a hypergraph, and let $\mathcal{H}'$ be any *partial hypergraph* of $\mathcal{H}$, obtained by deleting some of its edges. The *chromatic index* of a hypergraph, denoted $\chi_e(\mathcal{H})$, is the smallest number of colors required to color its edges such that no two edges of the same color share a common vertex. It is clear that the maximum degree $\Delta(\mathcal{H})$—the largest number of edges incident to any single vertex—provides a lower bound for $\chi_e(\mathcal{H})$.

Definition 2.2. A hypergraph $\mathcal{H}$ is *normal* if $\chi_e(\mathcal{H}') = \Delta(\mathcal{H}')$ for every partial hypergraph $\mathcal{H}'$ of $\mathcal{H}$.

Lovász [17] provided an alternative characterization of normal hypergraphs.

Theorem 2.3 (Lovász)**.** *A hypergraph $\mathcal{H}$ is normal if and only if all extreme points of the polyhedron $\{x \in \mathbb{R}^m : Ax \leq 1, x \geq 0\}$ are integral, where A is the incidence matrix of $\mathcal{H}$.*

An important subclass of normal hypergraphs is the class of unimodular hypergraphs.

Definition 2.4. A hypergraph is *unimodular* if its incidence matrix is totally unimodular.

Next, we introduce subclasses of unimodular hypergraphs.

- A *network hypergraph* is a hypergraph whose incidence matrix is a network matrix.
- *Arborescence hypergraphs* are a subclass of network hypergraphs: $\mathcal{H} = (V, \mathcal{E})$ is an *arborescence hypergraph* if there is an arborescence F over V such that each edge of $\mathcal{E}$ forms a directed path in F.
- *Subpath hypergraphs* are the special case of arborescence hypergraphs where the underlying arborescence is required to be a path. Formally, $\mathcal{H} = (V, \mathcal{E})$ is a *subpath hypergraph* if there exists a directed path F over V such that each edge of $\mathcal{E}$ forms a subpath in F.

- A *laminar hypergraph* is a hypergraph $\mathcal{H} = (V, \mathcal{E})$ for which there are no two hyperedges $e_1, e_2 \in \mathcal{E}$ that satisfy $e_1 \setminus e_2 \neq \emptyset$ and $e_2 \setminus e_1 \neq \emptyset$. Laminar hypergraphs are known to be subpath hypergraphs; although this seems to be folklore, we provide a proof in Appendix B.2.

Finally, we define a superclass of arborescence hypergraphs. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is a *subtree hypergraph* if there is a tree $F = (V, E)$ on the ground set V such that each hyperedge induces a subtree of F. While subtree hypergraphs are not necessarily unimodular, they still form a subclass of normal hypergraphs [7, Section 4.4].

Appendix B.2 offers a full, graphical overview of the inclusion relations among these hypergraph classes.

2.2 Stable Hypergraph Matching

We start by defining the Stable Hypergraph Matching and Stable Hypergraph b-Matching problems. Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. For each $v \in V$ we are given a strict preference list $\succ_v$ over the set $\mathcal{E}(v)$ of hyperedges containing v. In the case of Stable Hypergraph b-Matching, we are also given capacities $b(v) \in \mathbb{Z}_+$ for each $v \in V$.

Definition 2.5. A hyperedge $e \in \mathcal{E}$ is *dominated* by a (b-)matching M at some vertex $v \in V$ if v is saturated in M and $f \succ_v e$ for each hyperedge $f \in M(v)$. A hyperedge f *blocks* M if $f \notin M$ and there exists no vertex $v \in f$ at which M dominates f. A (b-)matching is *stable* if no hyperedge blocks it.

The most important question in such instances is to find a stable matching or b-matching. Hence, we get the following computational problem.

Stable Hypergraph b-Matching or SHbM

Input: A hypergraph $\mathcal{H} = (V, \mathcal{E})$ with capacities $b : V \to \mathbb{Z}$ and strict preferences $(\succ_v)_{v \in V}$ over incident edges for each vertex in V.

Output: Find a stable hypergraph b-matching in $\mathcal{H}$.

Stable Hypergraph Matching (or SHM) is defined analogously for matchings instead of b-matchings. It is clear that SHM is a subcase of SHbM where we have $b \equiv 1$.

For $\Pi \in \{$Unimod, Subpath, Subtree$\}$, we denote by Π-SHM and Π-SHbM the restrictions of SHM and SHbM to unimodular; subpath; or subtree hypergraphs, respectively.

As was observed by Aharoni and Fleiner [2], every stable hypergraph matching for an instance $(\mathcal{H}, (\succ_v)_{v \in V})$ of SHM with hypergraph $\mathcal{H} = (V, \mathcal{E})$ can be seen as an integral point in the polyhedron $\{x \in \mathbb{R}^m : Ax \leq 1, x \geq 0\}$ that dominates every column according to the preferences $(\succ_v)_{v \in V}$, where A is the incidence matrix of $\mathcal{H}$. Thus, Scarf's Lemma and Theorem 2.3 imply the existence of a stable matching if the hypergraph is normal. Furthermore, for an instance $(\mathcal{H}, b, (\succ_v)_{v \in V})$ of SHbM, the integral points of the polyhedron $\{x \in \mathbb{R}^m : Ax \leq b, 0 \leq x \leq 1\}$ that dominate every column correspond to the stable hypergraph b-matchings, where b (with a slight abuse of notation) is the vector containing the capacities of the vertices. If the incidence matrix A is TU, then all extreme points of this polyhedron are integral, and thus Scarf's Lemma guarantees the existence of a stable b-matching whenever the underlying hypergraph is unimodular. The guaranteed existence of a stable matching or stable b-matching makes the class of normal hypergraphs and its subclass, unimodular hypergraphs, particularly interesting and useful.

Although the existence of a stable matching or b-matching may be guaranteed in restricted hypergraph classes, it is also of interest to find a maximum-weight stable hypergraph matching. For this, an optimization variant of SHbM (and SHM) can be formulated by associating a *weight* with each hyperedge. Given a weight function $w : \mathcal{E} \to \mathbb{Z}$ over a set $\mathcal{E}$ of hyperedges, we extend it by setting $w(M) = \sum_{e \in M} w(e)$ for each $M \subseteq \mathcal{E}$.

Maximum-Weight SHbM or MaxW-SHbM

Input: A hypergraph $\mathcal{H} = (V, \mathcal{E})$ with capacities $b : V \to \mathbb{Z}$, weights $w : \mathcal{E} \to \mathbb{Z}$, and strict preferences $(\succ_v)_{v \in V}$ over incident edges for each vertex in V.

Output: Find a maximum-weight stable hypergraph b-matching in $\mathcal{H}$.

We will refer to the optimization variant of the matching case, i.e., SHM as Maximum-Weight SHM or MaxW-SHM. For $\Pi \in \{$Unimod, Subpath, Subtree$\}$, we denote by Π-MaxW-SHM and Π-MaxW-SHbM the restrictions of MaxW-SHM and MaxW-SHbM to unimodular, subpath, or subtree hypergraphs, respectively.

We remark that in the special case when the hypergraph $\mathcal{H}$ is a bipartite graph, MaxW-SHbM coincides with the maximum-weight stable b-matching problem (MaxW-SbM) which is solvable in polynomial time [13].

3 The University Dual Admission Problem

In this section, we introduce a practically relevant problem in student allocation which, as we will show, can also be modeled as a special case of Unimod-SHM.

The problem is motivated by practical situations where certain programs in higher education are funded jointly as a result of cooperation between universities and companies, leading to a more complicated dual admission system. In our model, motivated by a real scenario in Hungarian higher education, we assume that such programs, funded by various companies, can be treated independently from each other (even though different programs might be funded by the same company)—hence, we shift the focus from the funding companies to the programs themselves.

Formally, we have a set $U = \{u_1, \dots, u_{n_U}\}$ of *universities*, a set $P_i = \{p_{i1}, \dots, p_{ik_i}\}$ of *programs* for each university $u_i \in U$, and a set $S = \{s_1, \dots, s_{n_S}\}$ of *students*. The set of programs is denoted by $P = \bigcup_{i \in [n_U]} P_i$. We may refer to the set of students, universities, and programs together as the set of *agents*. Each university $u_i \in U$ has a *capacity* $c(u_i)$, and each program $p_{ik} \in P$ has a *quota* $q(p_{ik})$.

The students apply to both a university $u_i \in U$ and a program $p_{ik} \in P_i$ available at university u_i. Since each program $p_{ik} \in P$ uniquely determines the university u_i offering it, this can alternatively be viewed as students applying simply to programs.[1] Thus, for each student $s_j \in S$ we assume a strict preference order $\succ_{s_j}$ over the set of programs acceptable for s_j. Additionally, each university $u_i \in U$ has a strict preference order $\succ_{u_i}$ over the students, and each program $p_{ik} \in P$ has a strict ordering $\succ_{p_{ik}}$ over the students. A triple (s_j, u_i, p_{ik}) is an *acceptable triple* if s_j finds p_{ik} acceptable; note that we do not consider acceptability for universities or programs explicitly, as we implicitly assume that a student finds a program acceptable only if both the program and university offering it finds the student acceptable as well. We may treat acceptable triples as subsets of $S \cup U \cup P$ of size 3.

An *assignment* μ is a function $\mu : S \to P \cup \{\perp\}$ mapping each student $s_j \in S$ either to a program that is acceptable for s_j, or to the special symbol $\perp$ meaning that s_j is left unassigned. For simplicity, we assume that a program is acceptable for s_j if and only if it is preferred by s_j to $\perp$. For a program $p_{ik} \in P$, we let $\mu(p_{ik})$ denote the set of students assigned to it by μ, and similarly, for a university $u_i \in U$ we let $\mu(u_i)$ denote the set of students assigned by μ to some program in P_i. An assignment μ is *feasible* if $|\mu(u_i)| \leq c(u_i)$ for each $u_i \in U$ and $|\mu(p_{ik})| \leq q(p_{ik})$ for each $p_{ik} \in P$; we say that a university or a program is *unsaturated* if the corresponding inequality is strict, otherwise it is *saturated*.

Definition 3.1. A feasible assignment μ for an instance of UDA is *stable* if there is no *blocking* student–university–program triple, that is, a triple (s_j, u_i, p_{ik}) such that the following three conditions hold:

(i) $p_{ik} \succ_{s_j} \mu(s_j)$;
(ii) u_i satisfies one of the followings:
   - u_i is unsaturated,
   - there is a student $s_{j'} \in \mu(u_i)$ such that $s_j \succ_{u_i} s_{j'}$, or
   - $s_j \in \mu(u_i)$;

(iii) the program p_{ik} is unsaturated, or there is a student $s_{j''} \in \mu(p_{ik})$ such that $s_j \succ_{p_{ik}} s_{j''}$.

We are ready to formally define the corresponding computational problem.

University Dual Admission or UDA	
Input:	An instance $I = (S, U, P, c, q, (\succ_a)_{a \in S \cup U \cup P})$ of UDA as described above.
Question:	Find a feasible and stable assignment for I.

Let us remark that the above problem can easily be used to model a situation where students may apply either to university–program pairs or only to universities: to allow for this option, we simply need to create a dummy program p_i^* for each university $u_i \in U$ with quota $c(u_i)$ whose preferences are identical to the preferences of u_i.

[1] Nevertheless, universities play an important role in the admission problem, as will become clear in the definition of a stable assignment (Definition 3.1).

A very natural idea to solve UDA is to use a Gale–Shapley-like proposal–rejection algorithm and iterate it until it outputs a stable matching. We give an example where a student-proposing variant of the Gale–Shapley algorithm goes into an infinite loop when used on an instance of UDA, even if students propose in a lexicographic order. It is also possible to create instances where universities or programs propose but the algorithm still cycles in a similar way. This suggests that the problem should be solved in a different manner.

3.1 Formulating UDA as a Special Case of SHbM

In this section, we first show that UDA can be formulated as a special case of the UNIMOD-SHbM, and as a consequence, a stable assignment always exists for every instance of UDA. By contrast, we also prove that a stable assignment of maximum size is NP-hard to find.

Definition 3.2. Given an instance I of UDA, let $\mathcal{H}_I = (S \cup U \cup P, \mathcal{E}_I)$ denote the *hypergraph associated with* I whose vertices are the agents in I and whose hyperedges are the acceptable triples in I. We say that a hypergraph $\mathcal{H}$ has the UDA *property* if it can arise as $\mathcal{H}_I$ for some instance I of UDA.

We call UDA-SHbM and UDA-MAXW-SHbM the restriction of SHbM and MAXW-SHbM to hypergraphs having the UDA property, respectively.

To create an instance of SHbM that is equivalent with our instance I of UDA, we also need to define a strict ranking over the hyperedges of $\mathcal{H}_I$ incident to each student, university, or program. For a student $s_j \in S$, his preference list $\succ_{s_j}$ in I can be viewed as a preference list over its incident hyperedges: s_j prefers a hyperedge $\{s_j, u_i, p_{ik}\}$ over $\{s_j, u_{i'}, p_{i'k'}\}$ if and only if $p_{ik} \succ_{s_j} p_{i'k'}$. For a program $p_{ik} \in P$, the preference order $\succ_{p_{ik}}$ similarly defines a ranking over the incident hyperedges, as each such hyperedge contains the same university u_i. For a university $u_i \in U$, we extend its preference list to all incident hyperedges the following way. We let u_i prefer $\{s_j, u_i, p_{ik}\}$ to $\{s_{j'}, u_i, p_{ik'}\}$ if and only if either $s_j \succ_{u_i} s_{j'}$ or $s_j = s_{j'}$ and $p_{ik} \succ_{s_j} p_{ik'}$. Let $\succ_a^{\mathcal{H}}$ denote the preferences of agent $a \in S \cup U \cup P$ in I thus defined.

Additionally, we define the capacity function $b^{\mathcal{H}} : S \cup U \cup P$ over the vertices of $\mathcal{H}_I$ as

$$b^{\mathcal{H}}(a) = \begin{cases} 1, & \text{if } a \in S, \\ c(u_i), & \text{if } a = u_i \in U, \\ q(p_{ik}), & \text{if } a = p_{ik} \in P. \end{cases}$$

Theorem 3.3 ($\star$). *The stable assignments of an instance of $I = (S, U, P, c, q, (\succ_a)_{a \in S \cup U \cup P})$ of* UDA *correspond bijectively to the stable $b^{\mathcal{H}}$-matchings of the instance $I' = (\mathcal{H}_I, b^{\mathcal{H}}, (\succ_a^{\mathcal{H}})_{a \in S \cup U \cup P})$ of* SHbM.

Next, we show that the incidence matrix of the hypergraph associated with an arbitrary instance of UDA is a network hypergraph and is thus unimodular.

Theorem 3.4. *For each instance I of* UDA*, the associated hypergraph $\mathcal{H}_I$ is a network hypergraph.*

Proof. We create a directed graph $D = (V, E)$ together with a spanning tree F in the undirected graph $\overline{D} = (V, \overline{E})$ obtained by replacing each arc (u, v) of D by an edge between u and v. We will show that the incidence matrix of $\mathcal{H}_I$ is exactly the network matrix that corresponds to D with the spanning tree F.

We create a vertex in V for each student, university, and program, and we further define an additional vertex x. Formally, we set

$$V = \{v(a) : a \in S \cup U \cup P\} \cup \{x\}.$$

The set E of arcs is defined as follows. First, for each program p_{ik} at some university u_i, we add an arc $e(p_{ik}) = (v(p_{ik}), v(u_i))$. Second, for each university u_i, we add an arc $e(u_i) = (v(u_i), x)$. Third, for each student s_j, we add an arc $e(s_j) = (x, v(s_j))$. Let $E_F = \{e(a) : a \in S \cup U \cup P\}$ denote the set of arcs defined so far. Finally, for each acceptable triple (s_j, u_i, p_{ik}), i.e., for each hyperedge in $\mathcal{H}_I$, we add an arc $(v(s_j), v(p_{ik}))$. It is straightforward to see that $F := \{\{u, v\} : (u, v) \in E_F\}$ is a spanning tree in $\overline{D}$. Let A be the network matrix associated with D and F. Note that the edges of F are in a one-to-one correspondence with the vertices of the hypergraph $\mathcal{H}_I$, i.e., the set of agents in I. Furthermore, the edges of $E \setminus F$ are in a one-to-one correspondence with the hyperedges of $\mathcal{H}_I$.

Thus, it only remains to show that A is indeed the incidence matrix of $\mathcal{H}_I$, i.e., if a vertex $v \in V$ is contained in a hyperedge $\hat{e} \in \mathcal{E}_I$, then the corresponding entry in A is 1, and 0 otherwise. To see this, consider the unique cycle $C_{\hat{e}}$ defined by an arc $(v(s_j), v(p_{ik}))$ corresponding to some hyperedge $\hat{e} = \{s_j, u_i, p_{ik}\}$, which is the cycle formed by the four edges $(v(s_j), v(p_{il}))$, $e(p_{ik})$, $e(u_i)$, and $e(s_j)$. Observe that $C_{\hat{e}}$ contains exactly three arcs of F, namely the arcs of D corresponding to the three vertices (i.e., agents) incident to $\hat{e}$. Hence, there are exactly three non-zero entries in the column of A corresponding to $\hat{e}$, and these three entries are in the three rows that correspond to the vertices incident to $\hat{e}$. Moreover, since all arcs of $C_{\hat{e}}$ have the same orientation along $C_{\hat{e}}$, each of these three entries of A has value 1, as required. ◻

While Theorems 3.3 and 3.4 guarantee the existence of a stable assignment for every UDA instance, a maximum-size stable assignment is NP-hard to find, where the *size* of an assignment is the number of students it assigns to some program.

Theorem 3.5 (⋆)**.** *Finding a maximum size stable assignment in an instance of* UDA *is* NP*-hard, even if all capacities are one.*

To handle this computational difficulty, we formulate an integer program whose solutions are exactly the stable assignments for an instance $I = (S, U, P, c, q, (\succ_a)_{a \in S \cup U \cup P})$ of UDA; see Appendix C.2.

3.2 An XP Algorithm for UDA-MaxW-SHbM

Let us describe an algorithm for an instance $I = (\mathcal{H}, b, w, (\succ_v)_{v \in V})$ of UDA-MaxW-SHbM, where $\mathcal{H} = (V, \mathcal{E})$ over vertex set $V = S \uplus U \uplus P$ is a hypergraph that satisfies the UDA property. Let $U = \{u_1, \ldots, u_{n_U}\}$. We can assume that $P = \biguplus_{i \in [n_U]} P_i$, where the vertices $p \in P_i$ satisfy that for any $p \in e \in \mathcal{E}$, we have $\{u_i, p\} \subset e$.

The pseudocode of the algorithm is given in Algorithm 1. As a first step, we guess a *strategy* $\sigma : U \to \mathcal{E} \cup \{\bot\}$, that is, for each $u_i \in U$ we guess whether it will be saturated in the solution, and if so, what will be the worst hyperedge adjacent to u_i. Then, we create an instance I'_σ of MaxW-SbM, the maximum-weight bipartite stable b-matching problem. The underlying graph of I'_σ will be the bipartite graph $G_\sigma = (S \cup P, E_\sigma)$ where E_σ contains an edge $\{s, p\}$ if (i) there is a (unique) vertex $u \in U$ such that $e = \{s, u, p\} \in \mathcal{E}$ and (ii) for that vertex u, $e \succeq_u \sigma(u)$, where we assume that $e \succeq_u \bot$ always holds. The preferences of the agents in I'_σ are the projection of their original preferences. This is well defined, as each edge $\{s, p\} \in E_\sigma$ uniquely defines a hyperedge $\{s, u, p\} \in \mathcal{E}$; we further set $b'(s) = b(s)$, $b'(p) = b(p)$, and $w'(\{s, p\}) = w(\{s, u, p\})$ so that capacities and weights remain the same.

Let us say that a b'-matching M' in I'_σ is *valid* if, $M := \{\{s, u, p\} \in \mathcal{E} \mid \{s, p\} \in M'\}$ satisfies that for each $u \in U$, $|M(u)| = b(u)$ whenever $\sigma(u) \neq \bot$, and $|M(u)| < b(u)$ otherwise.

For a given strategy σ, the algorithm proceeds by computing a maximum-weight stable b'-matching M' in I'_σ. If M' is valid, then it adds the b-matching $M := \{\{s, u, p\} \in \mathcal{E} \mid \{s, p\} \in M'\}$ corresponding to M' to a set T. If M' is not valid, then we discard the guessed strategy σ. Finally, we output a maximum-weight stable b-matching from T.

Algorithm 1 UDA-MaxW-SHbM($\mathcal{H} = (V, \mathcal{E}), b, w, (\succ_v)_{v \in V}$) where $\mathcal{H}$ defined over $S \uplus U \uplus P$ has the UDA property

$T := \emptyset$
for all strategies $\sigma : U \to \mathcal{E} \cup \{\bot\}$ **do**
  $E_\sigma := \emptyset$
  for each $e = \{s, u, p\} \in \mathcal{E}$ **do**
    if $e \succeq_u \sigma(u)$ **then** $E_\sigma := E_\sigma \cup \{\{s, p\}\}$
  Set $I'_\sigma := ((S \cup P, E_\sigma), b', w', (\succ'_v)_{v \in S \cup P})$ where b', w', and $\succ'_v$ are obtained by projection from b, w, and $\succ_v$, respectively.
  Compute a maximum-weight stable b'-matching M' in I'_σ.
  $M := \{\{s, u, p\} \in \mathcal{E} \mid \{s, p\} \in M'\}$
  if $|M(u)| = b(u)$ whenever $\sigma(u) \neq \bot$, and $|M(u)| < b(u)$ otherwise for all $u \in U$ **then**
    $T := T \cup \{M\}$
return a maximum-weight b-matching M from T.

Theorem 3.6. UDA-MaxW-SHbM *can be solved in* $\mathcal{O}((\Delta_U + 1)^{|U|}) \cdot \text{poly}(|\mathcal{E}|)$ *time, where* U *correspond to the class of universities and* Δ_U *is the maximum degree of a vertex* $u \in U$.

Proof. We show that in the end of Algorithm 1, T will contain a maximum-weight stable b-matching for I.

Claim 3.7. *If M' is a valid stable b'-matching in I'_σ, then the corresponding b-matching $M = \{\{s,u,p\} \in \mathcal{E} \mid \{s,p\} \in M'\}$ is feasible and stable in I.*

Proof of Claim. Let M' be a valid stable b'-matching in I'_σ and let M be defined as in the statement of the claim. Then, M is a feasible b-matching in I: for $v \in S \cup P$, $|M(v)| \le b(v)$ follows from $b'(v) = b(v)$, and the capacities of the agents $u \in U$ are respected due to the validity of M'. It remains to show that M is stable in I.

For a hyperedge $e = \{s,u,p\}$, let $e_{I'} = \{s,p\}$. Suppose on the contrary that a hyperedge $e = \{s,u,p\}$ blocks M. First, if $e_{I'} \in E_\sigma$, then as p (respectively, s) is either unsaturated in both M and M', or $e \succ_p f$ ($e \succ_s f'$) for some $f \in M(p)$ ($f' \in M(s)$) implying $e_{I'} \succ'_p f_{I'} \in M'$ ($e_{I'} \succ'_s f'_{I'} \in M'$); this means that the edge $e_{I'}$ blocks M', a contradiction. Second, if $\{s,p\} \notin E_\sigma$, then $\sigma(u) \succ_u e$. In particular, $\sigma(u) \neq \perp$ and due to the validity of M' we know that u is saturated in M. Moreover, by construction, any edge $\{s,p\} \in M'$ satisfies $\{s,u,p\} \succeq_u \sigma(u)$. Therefore, u is saturated in M with hyperedges that are not worse for u than $\sigma(u)$. Hence e cannot block M, a contradiction. ◁

Claim 3.8. *Suppose that M is a stable b-matching in I. Define $\sigma : U \to \mathcal{E} \cup \{\perp\}$ such that $\sigma(u)$ is the worst hyperedge in $M(u)$ according to $\succ_u$ if u is saturated in M, and $\perp$ otherwise. Then every stable b'-matching M' in I'_σ is valid.*

Proof of Claim. Let M' be the b'-matching in I'_σ defined as $M' = \{\{s,p\} : \{s,u,p\} \in M\}$. By the definition of σ, it holds that $M' \subseteq E_\sigma$, and moreover, M' is valid by construction. We claim that M' is stable in I'_σ.

To show that M' is stable in I'_σ, assume for the sake of contradiction that some edge $\{s,p\} \in E_\sigma$ blocks M'. Then, both s and p are either unsaturated or prefer $\{s,p\}$ to some edge in M'. Since M is stable, this means that the hyperedge $e = \{s,u,p\}$ must be dominated at u. Therefore, we get that u is saturated in M and $\sigma(u) \succ_u e$. However, then the edge $\{s,p\}$ cannot be contained in E_σ, a contradiction.

Due to the many-to-many extension of the well-known Rural Hospitals Theorem (see e.g., [13]), for any stable b'-matching M'' in I'_σ, we have $|M''(v)| = |M'(v)|$ for all $v \in S \cup P$. Hence, all stable b'-matchings of I'_σ must be valid, because for any stable b'-matching N' and its corresponding b-matching N in I, we have $|N(u_i)| = \sum_{p \in P_i} |N'(p)|$ for $u_i \in U$ by the UDA property. This concludes the proof. ◁

As we know that a stable b-matching for I exists, Claims 3.8 guarantees that by iterating over all possible strategies, the algorithm is bound to find the strategy σ corresponding to the maximum-weight stable b-matching M. By Claims 3.7 and 3.8, $M' = \{\{s,p\} \mid \{s,u,p\} \in M\}$ will be a valid stable b'-matching for I_σ, and since any valid b'-matching N' gives a stable b-matching N in I with $w(N') = w(N)$, the algorithm indeed puts M or a stable b-matching with the same weight in T.

Since there are at most $(\Delta_U + 1)^{|U|}$ possible strategies to check, and for each one them, the instance I'_σ and a maximum-weight stable b-matching in I'_σ can be computed in poly($|\mathcal{E}|$) time by a result of Fleiner [13], the claimed running time also follows. □

Further results. On the positive side, we define a relaxed notion of stability in UDA, called *half-stability*, which intuitively means that for a triple to block, if the program is saturated, then the university and the program should be able to agree on a student that can be rejected; we show that we can find a half-stable assignment in polynomial time. Due to space constraints, this result is deferred to Appendix C.3.

Remark 3.9. If $c(u) = 1$ for all universities, then a simple deferred acceptance algorithm between the students and universities produces a stable assignment. This statement is explained further and proved formally in Appendix C.4.

4 SHbM for Unimodular Hypergraphs

We proceed to study UNIMOD-SHbM in further classes of unimodular hypergraphs.

4.1 Laminar Hypergraphs

We construct a polynomial-time algorithm that finds a stable b-matching if the hypergraph $\mathcal{H} = (V, \mathcal{E})$ is laminar. As laminar hypergraphs are unimodular, the existence of a stable b-matching is guaranteed. We present our algorithm for this problem in Algorithm 2.

Algorithm 2 Laminar-SHbM($\mathcal{H}, b, (\succ_v)_{v \in V}$) where $\mathcal{H} = (V, \mathcal{E})$ is laminar

1: $M := \emptyset$ ▷ M is the current b-matching
2: $\mathcal{C} := \emptyset$ ▷ $\mathcal{C}$ denotes the checked hyperedges
3: **while** $\mathcal{E} \setminus \mathcal{C} \neq \emptyset$ **do**
4: Pick an inclusion-wise minimal hyperedge f in $\mathcal{E} \setminus \mathcal{C}$.
5: $\mathcal{C} := \mathcal{C} \cup \{f\}$
6: **if** f blocks M **then**
7: Set $M := M \cup \{f\}$ and initialize $X_f := \emptyset$.
8: **for all** $v \in V$ with $|M(v)| > b(v)$ **do**
9: Add to X_f the hyperedge in $M(v)$ that is worst for v.
10: Set $Y_f := \{e : e \text{ is inclusion-wise maximal in } X_f\}$ and $M := M \setminus Y_f$.
11: **return** M

Theorem 4.1. Laminar-SHbM *is solved in polynomial time by Algorithm 2.*

Proof. First, let us show that the set returned by Algorithm 2 is a b-matching. Observe that whenever some hyperedge f is added to M on line 7, the set X_f constructed on line 9 covers every vertex v that became oversaturated as a consequence of adding f. Therefore, Y_f also covers these vertices, so after the removal of all hyperedges of Y_f from the matching, the resulting set again satisfies all capacity constraints. Therefore, the algorithm indeed returns a b-matching M.

It remains to show that M is stable. Consider the iteration of lines 4–10 during Algorithm 2 where some hyperedge f is picked on line 4, and let M_f and M'_f denote the current b-matching at the beginning and at the end of this iteration, respectively. We need the following claim.

Claim 4.2. *For any edge f added to the b-matching at line 7, the set Y_f is a family of pairwise disjoint hyperedges whose union is a subset of f, with $f \notin Y_f$.*

Proof of Claim. Since the algorithm always picks an inclusion-wise minimal hyperedge on line 4, the laminarity of $\mathcal{H}$ implies that every hyperedge that had been checked (i.e., put into $\mathcal{C}$) before f is either a subset of f or is disjoint from it. In particular, since the algorithm only puts hyperedges of the current b-matching into X_f, we get that all hyperedges in X_f are contained in f (recall that f shares at least one oversaturated vertex with each hyperedge in X_f). Since Y_f contains only the inclusion-wise maximal sets from X_f, the laminarity of $X_f \subseteq \mathcal{E}$ implies that the hyperedges in Y_f are pairwise disjoint.

To see $f \notin Y_f$, it suffices to observe the following. Since f blocks the current b-matching M_f, for each $v \in V$ that is oversaturated in $M_f \cup \{f\}$ we know that f cannot be the worst hyperedge incident to v in $M_f(v) \cup \{f\}$. ◁

Consider now the iteration during which f is picked on line 4. If the algorithm adds f to the current b-matching M_f on line 4, then $M'_f = M_f \cup \{f\} \setminus Y_f$, otherwise $M'_f = M_f$. Hence, the above claim implies $|M'_f(v)| \geq |M_f(v)|$ for each $v \in V$, since v can only be incident to at most one hyperedge in Y_f. Moreover, if $|M_f(v)| = b(v)$ for some $v \in V$, then the worst hyperedge in $M'_f(v)$ is weakly preferred by v to the worst hyperedge in $M_f(v)$: this is obvious if $v \notin f$, and if $v \in f$, then v prefers f to at least one hyperedge in $M_f(v)$, so v's worst hyperedge cannot be worse than before, as f is the only new hyperedge in $M'_f(v)$ not in $M_f(v)$ already. Hence, whenever some vertex becomes saturated with hyperedges better than some $e \in \mathcal{E}$, then this remains true during the remainder of the algorithm. We refer to this observation as the *saturation-monotonicity* of the algorithm.

To prove the stability of M, consider an arbitrary edge $f \in \mathcal{E} \setminus M$ and the iteration during which f is examined at line 4. If f is not added to M_f on line 7, then some vertex v incident to f must be saturated in M_f with hyperedges that are all preferred to f by v. By the saturation-monotonicity of the algorithm, this remains true for the returned b-matching M, so f cannot block M. By contrast, if f is added to M_f on line 7, then f must have been removed at some point on line 10 during a later iteration as a consequence of adding some hyperedge e to the b-matching M_e on line 7 for which $f \in Y_e$. By our construction of Y_e, we know that f is the worst hyperedge for some vertex $u \in V$ in $M_e(u)$, so after removing Y_e from the b-matching $M_e \cup \{e\}$, vertex u is saturated with hyperedges that are all preferred by u to f. Again, by the saturation-monotonicity of the algorithm, this remains true for u during the algorithm, so f cannot block M. Thus, M is indeed a stable b-matching for $\mathcal{H}$.

In every iteration, a hyperedge of $\mathcal{E} \setminus \mathcal{C}$ gets added to $\mathcal{C}$, so there are $|\mathcal{E}|$ iterations. As each iteration can be performed in linear time, the running time is polynomial. ☐

We complement Theorem 4.1 by a hardness result for a very restricted weight function.

Theorem 4.3 ($\star$). *The* MaxW-Laminar-SHM *problem is* NP*-complete on laminar hypergraphs, even if all edges have weight* 0 *with the exception of a single edge of weight* 1.

4.2 Subpath Hypergraphs

Moving our focus to instances of SHbM where the underlying hypergraph $\mathcal{H}$ is a subpath hypergraph, we present an algorithm for the MaxW-Subpath-SHbM problem that runs in polynomial time if the maximum size of a hyperedge in $\mathcal{H}$ is bounded by a constant.

Theorem 4.4. MaxW-Subpath-SHbM *can be solved in* $O((\Delta(b_{\max}+1))^{\ell_{\max}}|\mathcal{E}|\log\omega)$ *time for an instance* $(\mathcal{H}=(V,\mathcal{E}),b,w,(\succ_v)_{v\in V})$*, where* $b_{\max}$ *is the maximum capacity of a vertex,* $\ell_{\max}$ *is the maximum size of an edge in* $\mathcal{E}$*,* Δ *is the maximum degree of a vertex, and* ω *is the maximum absolute value of* $w(e)$ *over* $e\in\mathcal{E}$.

A brute-force approach to solve the MaxW-Subpath-SHbM problem would be to consider the edges one-by-one and enumerate all possible b-matchings that do not violate the capacities and that may turn out to be stable in the end. However, such a brute-force method would not yield a polynomial-time algorithm even if $\ell_{\max}$ is constant. Hence, we apply a more sophisticated approach using dynamic programming: we process edges one-by-one, but instead of recording all possible partial solutions, we only maintain a smaller set of b-matchings that, in a certain sense, represents all possible strategies for obtaining a stable b-matching. We start by introducing the necessary concepts.

Strategies. A *strategy* over a given set $W\subseteq V$ of vertices is a function $\sigma: W\to\mathcal{E}\cup\{\bot\}$ such that $\sigma(w)$ is incident to w for each $w\in W$ unless $\sigma(w)=\bot$. The interpretation of a strategy is that $\sigma(w)$ is the planned *threshold* of w, that is, the worst edge that we plan to put into the b-matching among those incident to w; the symbol $\bot$ corresponds to being unsaturated. We extend the notation $\succ_w$ such that $e\succ_w\bot$ holds for each e incident to w.

Definition 4.5. A b-matching M *realizes* a strategy σ over W w.r.t. a subset $\mathcal{E}'\subseteq\mathcal{E}$ of the edges, if

(i) M is *compatible* with σ, meaning that for each $w\in W$:
  - if $\sigma(w)\in\mathcal{E}$, then $e\succeq_w\sigma(w)$ holds for every $e\in M(w)$, and
  - if $\sigma(w)=\bot$, then $|M(w)|<c(w)$;

(ii) $\sigma(w)\in M(w)$ for each $w\in W$ with $\sigma(w)\in\mathcal{E}'$, and

(iii) every edge in $\mathcal{E}'\setminus M$ that blocks M is planned to be dominated by σ where we say that σ *plans to dominate* an edge e if there exists some $w\in W$ such that $\sigma(w)\succ_w e$.

We call a strategy *realizable* for an edge set $\mathcal{E}'\subseteq\mathcal{E}$ if some b-matching realizes it w.r.t. $\mathcal{E}'$.

Observe that if a strategy σ plans to dominate an edge e at some vertex w and M is a b-matching compatible with σ that also satisfies $|M(w)|=b(w)$, then e cannot block M, because e is *dominated* by M at w, meaning that M contains $b(w)$ edges incident to w, and w prefers each of them to e.

Given a strategy σ over W and a b-matching compatible with σ, we say that M is *complete* on a vertex set $W'\subseteq W$ for σ if $|M(v)|=b(v)$ for each $v\in W'$ such that $\sigma(v)\neq\bot$.

The algorithm. Let $(\mathcal{H},b,w,(\succ_v)_{v\in V})$ be our instance of MaxW-Subpath-SHbM with a subpath hypergraph $\mathcal{H}=(V,\mathcal{E})$, and let $v_1,\dots,v_n$ be the ordering of V witnessing the subpath property of $\mathcal{H}$. That is, for each hyperedge $e\in\mathcal{E}$ there exist indices $i\le j$ such that $e=\{v_i,v_{i+1},\dots,v_j\}$.

For a hyperedge $e\in\mathcal{E}$, we write $l(e)=\max\{i: v_i\in e\}$ and we say that $v_{l(e)}$ is the *endpoint* of e. Let $e_1,\dots,e_m$ be an ordering of $\mathcal{E}$ where $l(e_i)<l(e_j)$ implies $i<j$, that is, we order the edges increasingly according to their endpoints. Let $\mathcal{E}_i=\{e_1,\dots,e_i\}$, and let L_i contain the last $\ell_{\max}$ vertices contained in $\bigcup\mathcal{E}_i$, that is, $L_i=\{v_j:\max\{0,l(e_i)-\ell_{\max}\}<j\le l(e_i)\}$.

Our algorithm considers each set $\mathcal{E}_i=\{e_1,\dots,e_i\}$, and computes a set S_i of pairs (σ,M) where σ is a strategy over L_i and M is a b-matching in $(V,\mathcal{E}_i)$ that realizes σ. We will ensure that S_i is *representative* for $\mathcal{E}_i$ in the following sense:

Definition 4.6. Set S_i is w*-representative* for $\mathcal{E}_i$ if the following holds: whenever there is a strategy σ and a b-matching M realizing σ with respect to $\mathcal{E}_i$, then there is a b-matching M' equivalent to M on L_i for which $(\sigma,M')\in S_i$ and $w(M')\ge w(M)$. Here, M is *equivalent* to M' on L_i, if $|M(v)|=|M'(v)|$ for each $v\in L_i$; this is denoted by $M\sim_{L_i}M'$.
The operation of *updating* S_i with (σ,M) means that we put (σ,M) into S_i unless there is some pair $(\sigma,\hat{M})$ already in S_i for which $\hat{M}$ is equivalent on L_i to M and $w(\hat{M})\ge w(M)$.

We initialize S_0 to contain only the pair $(\sigma_\emptyset, \emptyset)$, where $\sigma_\emptyset$ is the function with empty domain. We then iterate over $i = 1$ to m, and build the set S_i as follows; for a formal description, see Algorithm 3.

Adjusting the domain from L_{i-1} to L_i: For each $(\sigma, M) \in S_{i-1}$, check whether M is complete for σ on $L_{i-1} \setminus L_i$. If not, then we will not be able to make σ complete by adding edges from $\mathcal{E} \setminus \mathcal{E}_{i-1}$, so we ignore the pair (σ, M). Otherwise, we change the domain of σ from L_{i-1} to L_i (forgetting all values assigned to vertices of $L_{i-1} \setminus L_i$) as follows: We set $\sigma(v)$ for each $v \in L_i \setminus L_{i-1}$ to all possible values in $\{e \in \mathcal{E} : v \in e\} \cup \{\perp\}$, and for each strategy σ' over L_i obtained this way, we put (σ', M) into a set T_i.

Computing S_i based on e_i: For each $(\sigma, M) \in T_i$ we check whether M still realizes σ with respect to $\mathcal{E}_i$ by checking if (i) σ plans to dominate e_i, and (ii) for each $v \in e_i$, the threshold $\sigma(v)$ does not coincide with e_i. If these conditions hold, then M realizes σ with respect to $\mathcal{E}_i$. Hence, we update S_i with (σ, M). Next, we check whether we can add e_i to M: if adding e_i does not violate the capacities and yields a b-matching compatible with σ, then we update S_i with $(\sigma, M \cup \{e_i\})$.

The algorithm finishes by checking whether S_m contains a pair (σ, M) such that M is complete for σ on L_m, and if so, outputs among all such b-matchings M one that maximizes $w(M)$.

Algorithm 3 MaxW-Subpath-SHbM$(\mathcal{H} = (V, \mathcal{E}), b, w, (\succ_v)_{v \in V})$ where $\mathcal{H}$ is a subpath hypergraph

```
1: Set L_0 := ∅ and σ_∅ as the empty function over ∅.
2: Set S_0 := {(σ_∅, ∅)}.
3: for i = 1 to m do
4:     for all pair (σ, M) ∈ S_{i−1} do
5:         if M is not complete for σ on L_{i−1} \ L_i then
6:             break;                                   ▷ Continue with next pair in S_{i−1}.
7:         for all σ'' : L_i \ L_{i−1} → E ∪ {⊥} do
8:             if for each v ∈ L_i \ L_{i−1}, either σ''(v) ∋ v or σ''(v) = ⊥ then
9:                 Set σ' as the union σ|_{L_{i−1} ∩ L_i} and σ'', with domain L_i.
10:                T_i ← (σ', M)
11:    S_i := ∅
12:    for all pair (σ, M) ∈ T_i do
13:        if σ plans to dominate e_i and e_i ≠ σ(v) for each v ∈ e_i then
14:            if ∄(σ, M̂) ∈ S_i : M ∼_{L_i} M̂ and w(M̂) ≥ w(M) then S_i ← (σ, M)
15:        if e_i ⪰_v σ(v) for each v ∈ e_i and ∀v ∈ e_i : |M(v)| < b(v) then
16:            M' := M ∪ {e_i}
17:            if ∄(σ, M̂) ∈ S_i : M' ∼_{L_i} M̂ and w(M̂) ≥ w(M') then S_i ← (σ, M')
18: Set w* := −∞ and M* := ∅.
19: for all pair (σ, M) ∈ S_m do
20:    if M is complete for σ on L_m and w(M) > w* then
21:        Set w* := w(M) and M* := M.
22: return M*.
```

Lemma 4.7 ($\star$)**.** *If Algorithm 3 puts a pair (σ, M) into S_i for some $i \in [m]$, then M realizes σ with respect to $\mathcal{E}_i$.*

Proof sketch. We use induction on i. The case $i = 0$ is trivial, so assume that the lemma holds for $i - 1$.

First, suppose that (σ, M) was added to S_i on line 14, so $M \subseteq \mathcal{E}_{i-1}$. Let $(\sigma^-, M) \in S_{i-1}$ be the pair selected on line 4 in the cycle during which (σ, M) was put into T_i on line 10. By induction,

(a) M is compatible with σ^-,
(b) $\sigma^-(v) \in M(v)$ for every $v \in L_{i-1}$ where $\sigma^-(v) \in \mathcal{E}_{i-1}$, and
(c) every edge in $\mathcal{E}_{i-1} \setminus M$ that blocks M is planned to be dominated by σ^-.

First, observe that M is compatible with σ: the compatibility conditions for each $v \in L_{i-1} \cap L_i$ follow immediately from $\sigma^-(v) = \sigma(v)$ due to (a), while the compatibility condition for each $v \in L_i \setminus L_{i-1}$ holds irrespective of the value of $\sigma(v)$, because edges of $M \subseteq \mathcal{E}_{i-1}$ have no vertices in $L_i \setminus L_{i-1}$.

Second, by our assumption that Algorithm 3 added (σ, M) to S_i on line 14, we know that $\sigma(v) \neq e_i$ for each v incident to e_i. Hence, if $\sigma(v) \in \mathcal{E}_i$ for some $v \in L_i$, then $\sigma(v) \in \mathcal{E}_{i-1}$, and consequently,

$v \in L_i \cap L_{i-1}$ (because vertices of $L_i \setminus L_{i-1}$ are not contained in any edge of $\mathcal{E}_{i-1}$). Therefore, (b) implies $\sigma(v) = \sigma^-(v) \in M(v)$. Third, it remains to show that every edge $e \in \mathcal{E}_i \setminus M$ that blocks M is planned to be dominated by σ. First note that Algorithm 3 checks that this holds for e_i explicitly on line 13, so we may assume $e \neq e_i$. By (c), we know that every edge in $\mathcal{E}_{i-1} \setminus M$ that blocks M is planned to be dominated by σ^-, so there is some $v \in L_{i-1}$ such that $\sigma^-(v) \succ_v e$. On the one hand, if $v \in L_i \cap L_{i-1}$, then e is planned to be dominated by σ too, because $\sigma(v) = \sigma^-(v)$. On the other hand, if $v \in L_{i-1} \setminus L_i$, then Algorithm 3 must have confirmed on line 10 that M is complete on v (otherwise it would not have put (σ^-, M) into T_i), which implies that M dominates e at v. Thus, e does not block M, proving that every edge in $\mathcal{E}_i \setminus M$ that blocks M is planned to be dominated by σ. This finishes the proof that M realizes σ with respect to $\mathcal{E}_i$.

The case when (σ, M) was put into S_i on line 17 is similar; see Appendix D.2 for the remainder. ⊿

Lemma 4.8. *Set S_i computed by Algorithm 3 is w-representative for $\mathcal{E}_i$ for each $i \in [m]$.*

Proof. We prove the lemma by induction on i. Clearly, the lemma trivially holds for $i = 0$, so let us assume that the statement holds for $i - 1$.

To prove that S_i remains w-representative for $\mathcal{E}_i$, it suffices to show the following: if (σ, M) is such that $M \subseteq \mathcal{E}_i$ realizes σ with respect to $\mathcal{E}_i$, then the algorithm updates S_i with *some* pair (σ, M') either at line 14 or at line 17 for which $M' \sim_{L_i} M$ (note that $\sim_{L_i}$ is transitive) and $w(M') \geq w(M)$.

Given σ and M, let us define the strategy σ^- over L_{i-1} as follows:

$$\sigma^-(v) = \begin{cases} \sigma(v), & \text{if } v \in L_{i-1} \cap L_i; \\ \bot, & \text{if } v \in L_{i-1} \setminus L_i \text{ and } |M(v)| < b(v); \\ \texttt{worst}(M, v), & \text{if } v \in L_{i-1} \setminus L_i \text{ and } |M(v)| = b(v) \end{cases}$$

where $\texttt{worst}(M, v)$ is the worst edge in $M(v)$ according to $\succ_v$. Note that M is complete on vertices of $L_{i-1} \setminus L_i$ for σ^-. We distinguish between two cases.

Case A: $e_i \notin M$. We claim that M realizes σ^- w.r.t. $\mathcal{E}_{i-1}$. First, since M realizes σ w.r.t. $\mathcal{E}_i$, we know that (i) M satisfies the compatibility conditions for σ^- on vertices of $L_i \cap L_{i-1}$, because σ coincides with σ^- on these vertices, and (ii) $\sigma^-(v) = \sigma(v) \in M(v)$ for each $v \in L_i \cap L_{i-1}$ that satisfies $\sigma(v) \in \mathcal{E}_i$. Considering some $v \in L_{i-1} \setminus L_i$, by the definition of σ^-, we get that M is compatible with σ^-, and also that $\sigma(v) = \texttt{worst}(M, v) \in M(v)$ for every $v \in L_{i-1} \setminus L_i$ with $\sigma(v) \neq \bot$. Hence, to see the claim, we only need to show that σ^- plans to dominate every edge $e \in \mathcal{E}_{i-1} \setminus M \subseteq \mathcal{E}_i \setminus M$ that blocks M. Clearly, e is planned to be dominated by σ at some $v \in L_i$ (since M realizes σ w.r.t. $\mathcal{E}_i$). However, as $e \in \mathcal{E}_{i-1}$, it follows that $v \in L_{i-1} \cap L_i$, and thus σ^- indeed plans to dominate e at v by $\sigma^-(v) = \sigma(v)$. This proves the claim that M realizes σ^- with respect to $\mathcal{E}_{i-1}$. Therefore, by induction there exists a b-matching M' with $w(M') \geq w(M)$ such that $(\sigma^-, M') \in S_{i-1}$ and $M' \sim_{L_{i-1}} M$.

Recall that M is complete on $L_{i-1} \setminus L_i$ for σ^-, and thus $M' \sim_{L_{i-1}} M$ implies that M' is also complete on $L_{i-1} \setminus L_i$ for σ^-. Hence, (σ^-, M') passes the test on line 5. As Algorithm 3 tries all possible ways to extend σ^-, it follows that $(\sigma, M') \in T_i$. Now, since M realizes σ w.r.t. $\mathcal{E}_i$ but $e_i \notin M$, we get that $e_i \neq \sigma(v)$ for any $v \in e_i$. Additionally, σ plans to dominate e_i: if e_i blocks M, then this is because M realizes σ w.r.t. $\mathcal{E}_i$ (by our assumptions), and if e_i does not block M, then it is dominated at some vertex $v \in L_i$, which in turn implies that v is saturated by M and $\texttt{worst}(M, v) = \sigma(v) \succ_v e_i$, that is, σ indeed plans to dominate e_i. Therefore, Algorithm 3 updates S_i with (σ, M') at line 14.

Case B: $e_i \in M$. Consider the b-matching $M^- = M \setminus \{e_i\}$; we claim that M^- realizes σ^- w.r.t. $\mathcal{E}_{i-1}$. Again, since M realizes σ w.r.t. $\mathcal{E}_i$, we know that (i) $M^- \subseteq M$ satisfies the compatibility conditions for σ^- on vertices of $L_i \cap L_{i-1}$, because σ coincides with σ^- on these vertices, and (ii) $\sigma^-(v) = \sigma(v) \in M^-(v)$ for each $v \in L_i \cap L_{i-1}$ that satisfies $\sigma(v) \in \mathcal{E}_{i-1} = \mathcal{E}_i \setminus \{e_i\}$. By the definition of σ^- on $L_{i-1} \setminus L_i$, it follows that M^- is compatible with σ^-, and $\sigma(v) \in M^-(v)$ for each $v \in L_{i-1}$ with $\sigma(v) \in \mathcal{E}_{i-1}$. Hence, to see the claim, we only need to show that σ^- plans to dominate every edge $e \in \mathcal{E}_{i-1} \setminus M^- = \mathcal{E} \setminus M$ that blocks M^-. If e blocks M as well, then σ plans to dominate it at some $v \in L_i$ (since M realizes σ w.r.t. $\mathcal{E}_i$). Since $e \in \mathcal{E}_{i-1}$, it follows that $v \in L_{i-1} \cap L_i$, and thus σ^- plans to dominate e as well. If, by contrast, e does not block M, then there must exist some $v \in L_i \cap e_i$ where M dominates e. Then, $|M(v)| = b(v)$ and $\texttt{worst}(M, v) = \sigma(v) \succ_v e$ (since M realizes σ w.r.t. $\mathcal{E}_i$), and so $\sigma^-(v) \succ_v e$ by the definition of σ^-. Hence, σ^- plans to dominate e at v. This proves the claim that M^- realizes σ^- with respect to $\mathcal{E}_{i-1}$. Therefore, by induction there exists a b-matching M' with $w(M') \geq w(M^-)$ such that $(\sigma^-, M') \in S_{i-1}$ and $M' \sim_{L_{i-1}} M^-$.

Recall that M^- is complete on $L_{i-1} \setminus L_i$ for σ^-, and thus $M' \sim_{L_{i-1}} M^-$ implies that M' is also complete on $L_{i-1} \setminus L_i$ for σ^-. Therefore, (σ^-, M') passes the test on line 5. As Algorithm 3 tries all possible ways to extend σ^-, it follows that $(\sigma, M') \in T_i$. Since $e_i \in M$ and M is compatible with σ, we know that $e_i \succeq_v \sigma(v)$ for each $v \in L_i$. By $M' \sim_{L_{i-1}} M^-$ and since $M^- \cup \{e_i\} = M$ is a b-matching, we know $|M'(v)| < b(v)$ for each $v \in e_i \cap L_{i-1}$; since $M' \subseteq \mathcal{E}_{i-1}$, it is also clear that $|M'(v)| = 0 < b(v)$ for each $v \in L_i \setminus L_{i-1}$. Therefore, the pair (σ, M') satisfies the conditions on line 15 and so Algorithm 3 updates S_i with $(\sigma, M' \cup \{e_i\})$ at line 17. By $M' \cup \{e_i\} \sim_{L_i} M^- \cup \{e_i\} = M$ and $w(M' \cup \{e_i\}) \geq w(M^-) + w(e_i) = w(M)$, this finishes the proof of the lemma. □

Proof of Theorem 4.4. We start by proving that Algorithm 3 solves MaxW-Subpath-SHbM correctly on subpath hypergraphs.

First, we show that the output is stable. Observe that if Algorithm 3 returns a b-matching M on line 22, then M realizes σ on $\mathcal{E}$, and thus σ plans to dominate every edge blocking M. However, as M is complete for σ (due to the condition on line 20), M in fact dominates every edge that σ plans to dominate. Thus, no edge may block M.

Assume now that M is a maximum-weight stable b-matching in our instance. Let σ be the strategy over L_m defined as

$$\sigma(v) = \begin{cases} \perp, & \text{if } |M(v)| < b(v); \\ \texttt{worst}(M, v), & \text{if } |M(v)| = b(v). \end{cases}$$

It is clear that M realizes σ over $\mathcal{E}$. Thus, by Lemma 4.8 there exists a b-matching M' satisfying $w(M') \geq w(M)$ for which $(\sigma, M') \in S_m$ and $M \sim_{L_m} M'$. Moreover, as M is complete on L_m for σ and $M \sim_{L_m} M'$, we know that M' is complete on L_m for σ as well. Hence, Algorithm 3 will set $w^\star$ to at least $w(M') \geq w(M)$ on line 21 at latest when it examines the pair (σ, M'), and will thus return a stable b-matching of weight at least $w(M)$ on line 22, which must be a maximum-weight stable b-matching.

To bound the running time of Algorithm 3, note that the number of possible strategies for a given set L_i is at most $\Delta^{|L_i|} \leq \Delta^{\ell_{\max}}$ where Δ is the maximum number of edges incident to any vertex. Given a strategy σ over L_i, we also know $|\{M : (\sigma, M) \in S_i\}| \leq (b_{\max} + 1)^{|L_i|} \leq (b_{\max} + 1)^{\ell_{\max}}$ since this set does not contain two b-matchings that are equivalent on L_i. This yields

$$|S_i| \leq (\Delta(b_{\max} + 1))^{\ell_{\max}}.$$

There are m iterations, and the computation of S_i from S_{i-1} is proportional to $|S_{i-1}| + |S_i|$, as any line of the algorithm can be performed in $O(\ell_{\max} \log \omega)$ time. Thus, we obtain that the total running time is $O((\Delta(b_{\max} + 1))^{\ell_{\max}} m \log \omega)$, as promised. □

Remark 4.9. We remark that Algorithm 3 can be used to decide whether a stable b-matching exists that contains a given set of "forced" edges (or is disjoint from a given set "forbidden" edges) by setting an appropriate weight function. Furthermore, it can also be used to decide the existence of a stable b-matching that saturates (or does not saturate) a given vertex v. Indeed, to decide whether some stable b-matching saturates v, it suffices to add a newly created hyperedge e_v incident only to v, extend the preferences of v such that e_v becomes the least-favorite edge according to $\succ_v$, and decide whether this modified instance admits a stable b-matching that does not contain e_v. Deciding whether some stable b-matching leaves v unsaturated can be done analogously, using the same construction but searching for a stable b-matching that contains e_v.

Theorem 4.10 shows that the running time of the algorithm in Theorem 4.4 cannot be improved to an FPT algorithm with parameter $\ell_{\max}$, not even if we additionally parameterize the problem with the maximum capacity and severely restrict the weight function.

Theorem 4.10 (⋆)**.** *The* MaxW-Subpath-SHbM *problem is* W[1]*-hard when parameterized by* $b_{\max} + \ell_{\max}$*, where* $b_{\max}$ *is the maximum capacity of a vertex and* $\ell_{\max}$ *is the maximum size of an edge, even if all edges have weight* 0 *with the exception of a single edge of weight* 1*.*

5 SHbM for Subtree Hypergraphs

We now turn our focus to the class of subtree hypergraphs. As these are not necessarily unimodular, so an instance of Subtree-SHbM may not admit a stable b-matching. However, as subtree hypergraphs are still normal, Subtree-SHM is guaranteed to be solvable.

We start by presenting a polynomial-time algorithm that solves Subtree-SHM. Before explaining our algorithm, let us introduce some additional notation. Let $\mathcal{H} = (V, \mathcal{E})$ be a subtree hypergraph and let $T = (V, F)$ be the underlying tree, that is, for each hyperedge $e \in \mathcal{E}$, the set of vertices contained in e

Algorithm 4 Subtree-SHM$(\mathcal{H}, (\succ_v)_{v\in V})$ where $\mathcal{H} = (V, \mathcal{E})$ is a subtree hypergraph with underlying tree T rooted at $r \in V$

if $\mathcal{E} = \emptyset$ **then return** $M = \emptyset$.
$f := \text{argmax}_{e\in\mathcal{E}} \text{dist}_{T,r}(e)$
$r_f := \text{top}_{T,r}(f)$
$X_f := \{f' \in \mathcal{E} \mid r_f \in f', f \succ_{r_f} f'\}$
$\mathcal{E}' := \mathcal{E} \setminus (X_f \cup \{f\})$
$M = \text{Subtree-SHM}((V, \mathcal{E}'), (\succ_v)_{v\in V})$
if r_f is not covered by M **then** $M := M \cup \{f\}$.
return M.

induces a subtree of T. Let r be a fixed vertex of V and consider the rooted tree T with root r. For each vertex v, let $\text{dist}_T(r, v)$ be the *distance* of r and v in T, that is the number of edges of the (unique) path of T between r and v. For each hyperedge $e \in \mathcal{E}$, let $\text{top}_{T,r}(e)$ be the (unique) vertex of e that is closest to r in T, and define $\text{dist}_{T,r}(e) := \text{dist}_T(r, \text{top}_{T,r}(e))$.

Given an instance $(\mathcal{H}, (\succ_v)_{v\in V})$ of Subtree-SHM, where $\mathcal{H} = (V, \mathcal{E})$ is a subtree hypergraph with underlying tree T, algorithm Subtree-SHM does the following (see Algorithm 4 for a pseudocode). First, it finds a hyperedge f such that $\text{top}_{T,r}(f)$ is furthest from r in T; for simplicity, let us write r_f for $\text{top}_{T,r}(f)$. Then it deletes f and every hyperedge incident to r_f that are less preferred by r_f than f, and recursively calls itself on the remaining hypergraph. Finally, it builds a matching by adding f to the output M of the recursive call if r_f is not covered by M.

Theorem 5.1. *Algorithm 4 solves* Subtree-SHM *in polynomial time.*

Proof. The algorithm runs in polynomial time, because Subtree-SHM is called at most $|\mathcal{E}|$ times recursively, and in each call, it only needs to compute the farthest hyperedge f and delete those hyperedges incident to $r_f = \text{top}_{T,r}(f)$ that are worse than f for r_f.

We show that the output M is a stable hypergraph matching. We use induction on $|\mathcal{E}|$. If $\mathcal{E} = \emptyset$, then the output $M = \emptyset$ of Algorithm 4 is clearly a stable hypergraph matching.

Hence, assume that $\mathcal{E} \neq \emptyset$. Let f be the hyperedge chosen by the algorithm in the first iteration. By induction, M is a stable hypergraph matching in the subtree hypergraph $(V, \mathcal{E} \setminus (X_f \cup \{f\}))$.

First, suppose that M saturates r_f, so Algorithm 4 outputs M. Then, M is clearly a matching. Also, no hyperedge of $\mathcal{E} \setminus (X_f \cup \{f\})$ blocks M by induction. Finally, no hyperedge of $X_f \cup \{f\}$ blocks M either, because r_f is covered with some hyperedge $e \in M$ such that $e \succ_{r_f} f \succ_{r_f} f'$ for every $f' \in X_f$ by the choice of X_f.

Suppose now that M does not saturate r_f, and the output is therefore $M \cup \{f\}$. The fact that $\mathcal{H}$ is a subtree hypergraph and that f is chosen so that it maximizes $\text{dist}_{T,r}(f)$ implies that any hyperedge that intersects f must also contain r_f. Hence, as M does not cover r_f, $M \cup \{f\}$ is a matching. By induction, no hyperedge of $\mathcal{E} \setminus (X_f \cup \{f\})$ blocks M, and thus $M \cup \{f\}$. Finally, no hyperedge of X_f blocks $M \cup \{f\}$, because $f \succ_{r_f} f'$ for every $f' \in X_f$ and r_f is saturated by $M \cup \{f\}$.

We conclude that the output is a stable matching. □

A natural question to ask whether a stable b-matching can still be found efficiently in the SHbM setting. Sadly, the answer is no, even with very severe restrictions.

Theorem 5.2 (⋆)**.** *Deciding if there exists an stable hypergraph b-matching in an instance of* Subtree-SHbM *is* NP*-hard, even if the underlying tree is a star, only one vertex has capacity greater than 1, and all hyperedge sizes are at most 4.*

Remark 5.3. Using the reduction presented in the proof of Theorem 5.2 and the fact that the problem of finding a stable *fractional* hypergraph matching is PPAD-complete even if each hyperedge has size 3 [9], it is straightforward to verify that the problem of finding a stable fractional hypergraph matching in an instance of Subtree-SHbM satisfying the conditions of Theorem 5.2 is PPAD-complete too.

6 Conclusions

We studied stable hypergraph matchings and b-matchings, focusing on structured hypergraph families. We introduced the University Dual Admission problem and showed it can be regarded as a special case of

UNIMOD-SHbM. We identified multiple classes of hypergraphs where stable (b-)matchings can be found efficiently, and provided contrasting proofs of intractability; see Table 1 for a summary.

Besides some open questions in Table 1, the main problem we leave open for further research is whether it is possible to find a stable matching in an arbitrary instance of UNIMOD-SHM in polynomial time. Our complexity results seem to suggest that the problem might be computationally hard (e.g., PPAD-hard) in general; however, for special cases there is a potential for efficient combinatorial algorithms. In particular, resolving the complexity of this problem for network hypergraphs would be of great interest. More generally, the development of new techniques to attain stable solutions for instances of SHM and SHbM is an important direction for future research that is motivated by various practical application such as the HOSPITALS/RESIDENTS WITH COUPLES problem [18, Chapter 5.3].

Funding

All authors acknowledge the financial support by the Hungarian Academy of Sciences under its Momentum Programme, grant number LP2021-2. Gergely Csáji and Péter Biró acknowledge the financial support by the Hungarian Scientific Research Fund via OTKA grant number K143858. Gergely Csáji is further supported by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation fund, financed under the KDP-2023 funding scheme (grant number C2258525). Ildikó Schlotter is supported by the Hungarian Academy of Sciences under its János Bolyai Research Scholarship.

References

[1] David J. Abraham, Robert W. Irving, and David F. Manlove. Two algorithms for the Student–Project Allocation problem. *Journal of Discrete Algorithms*, 5(1):73–90, 2007. doi:10.1016/j.jda.2006.03.006.

[2] Ron Aharoni and Tamás Fleiner. On a lemma of Scarf. *Journal of Combinatorial Theory, Series B*, 87(1):72–80, 2003. doi:10.1016/S0095-8956(02)00028-X.

[3] Haris Aziz and Florian Brandl. Existence of stability in hedonic coalition formation games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, volume 2, pages 763–770, 2012.

[4] Suryapratim Banerjee, Hideo Konishi, and Tayfun Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18(1):135–153, 2001. doi:10.1007/s003550000067.

[5] Claude Berge. *Hypergraphs.* North-Holland, 1989.

[6] Péter Biró and Tamás Fleiner. Fractional solutions for capacitated NTU-games, with applications to stable matchings. *Discrete Optimization*, 22:241–254, 2016. doi:10.1016/j.disopt.2015.02.002.

[7] Alain Bretto. *Hypergraph Theory: An Introduction.* Mathematical Engineering. Springer Cham, 2013.

[8] Karthekeyan Chandrasekaran, Yuri Faenza, Chengyue He, and Jay Sethuraman. Scarf's algorithm on arborescence hypergraphs. arXiv:2412.03397 [cs.DM], 2024. doi:10.48550/arXiv.2412.03397.

[9] Gergely Csáji. On the complexity of stable hypergraph matching, stable multicommodity flow and related problems. *Theoretical Computer Science*, 931:1–16, 2022. doi:10.1016/j.tcs.2022.07.025.

[10] Yuri Faenza, Chengyue He, and Jay Sethuraman. Scarf's algorithm and stable marriages. arXiv:2303.00791 [math.CO], 2023. doi:10.48550/arXiv.2303.00791.

[11] Micheal R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

[12] Rita Fleiner, András Ferkai, and Péter Biró. College admission problem for university dual education. In *Proceedings of the IEEE 17th World Symposium on Applied Machine Intelligence and Informatics (SAMI 2019)*, pages 31–36. IEEE, 2019. doi:10.1109/SAMI.2019.8782783.

[13] Tamás Fleiner. On the stable b-matching polytope. *Mathematical Social Sciences*, 46(2):149–158, 2003. doi:10.1016/S0165-4896(03)00074-X.

[14] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. doi:10.2307/2312726.

[15] Guillaume Haeringer. *Market Design: Auctions and Matching*. MIT Press, 2018.

[16] Shiva Kintali, Laura J. Poplawski, Rajmohan Rajaraman, Ravi Sundaram, and Shang-Hua Teng. Reducibility among fractional stability problems. *SIAM Journal on Computing*, 42(6):2063–2113, 2013. doi:10.1137/120874655.

[17] László Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 2(3):253–267, 1972. doi:10.1016/0012-365X(72)90006-4.

[18] David F. Manlove. *Algorithmics of Matching Under Preferences*, volume 2 of *Series on Theoretical Computer Science*. World Scientific, 2013. doi:10.1142/8591.

[19] David F. Manlove, Robert W. Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1–2):261–279, 2002. doi:10.1016/S0304-3975(01)00206-7.

[20] Cheng Ng and Daniel S. Hirschberg. Three-dimensional stable matching problems. *SIAM Journal on Discrete Mathematics*, 4(2):245–252, 1991. doi:10.1137/0404023.

[21] Alvin E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92(6):991–1016, 1984. doi:10.1086/261272.

[22] Alvin E. Roth and Marilda A. O. Sotomayor. *Two-sided Matching: a Study in Game-theoretic Modeling and Analysis*. Econometric Society Monographs, Cambridge, 1990.

[23] Herbert E. Scarf. The core of an N person game. *Econometrica*, 35(1):50–69, 1967. doi:10.2307/1909383.

[24] Alexander Schrijver. *Combinatorial Optimization*. Springer-Verlag Berlin Heidelberg, 2003.

[25] Gerhard J. Woeginger. Core stability in hedonic coalition formation. In *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2013)*, volume 7741 of *Lecture Notes in Computer Science*, pages 33–50. Springer, 2013. doi:10.1007/978-3-642-35843-2_4.

Appendix A Motivation by Applications

Instances of SHM and SHbM on special hypergraph classes arise when there is some underlying structure over participating agents. Such structure may be due to the physical positions of agents, organizational constraints, or some other feature of the model. Besides the real-life application of dual admission in Hungarian universities already mentioned [12], we briefly describe three additional examples:

Budgeting in transportation networks: Suppose that a set of cities aims to collaborate in order to renovate certain roads (or other infrastructure) running between the cities. Assume that there are several possible contracts to choose from, each involving a subset of the cities (and the roads pertaining to these cities), with each city having preferences over the contracts it is involved in. Such a scenario can be formulated as an instance of SHM with cities as agents and contracts as hyperedges. In particular, if cities are positioned along a single road or they are connected in a treelike fashion, then the underlying hypergraph may be a subpath or a subtree hypergraph, respectively.

Project selection management: Consider a firm whose organizational structure is a rooted tree T. Suppose that there is a set of possible projects, each of them to be undertaken by different working groups, and an optimal subset of the projects needs to be selected. Naturally, the firm's employees may have different preferences over the projects, and employees may have an upper bound on the number of projects they can participate in. To incentivize employees, notions of stability may become important, and hence such a situation can be modeled as an instance of SHbM where agents are employees and hyperedges represent projects. If each working group responsible for a project forms a subtree of T, the underlying hypergraph is a subtree hypergraph. In the special case when working groups always include all subordinates of any participant, the underlying hypergraph is laminar.

Coalition formation in politics: Often, political parties can be represented on a left–right axis based on their ideology. To gain power, parties may need to form coalitions, and it is reasonable to assume that possible coalitions form subpaths along this axis. Then coalition formation can be thought of as an instance of SHM over a subtree hypergraph.

Appendix B Additional Material on Section 2

B.1 Graph-Theoretic Basics

Let us provide here some basic definitions and notation related to directed and undirected graphs.

An undirected (directed) graph consists of set V of vertices and a set E of edges (arcs, respectively). The edge *connecting* two vertices u and v of an undirected graph is denoted by $\{u, v\}$, while the arc *leading from u to v* in a directed graph is denoted by (u, v); in both cases, u and v are the called the endpoints of the edge or arc.

Given an undirected or directed graph G, a series $e_1, e_2, \ldots, e_k$ of edges or arcs in G forms a *cycle* if there are vertices $v_1, v_2, \ldots, v_\ell$ in G such that for each $i \in [\ell]$, (a) $e_i = \{v_i v_{i+1}\}$ in the undirected case, and (b) $e_i = (v_i, v_{i+1})$ in the directed case, where in both cases the subscripts are taken modulo ℓ, i.e., $v_{\ell+1} = v_1$.

An undirected graph $G = (V, E)$ is a *tree* if it is *acyclic*, i.e., it does not contain a cycle, and additionally, it is not possible to add an edge to G so that it remains acyclic. A *clique* is a set of vertices in G whose vertices are pairwise adjacent, and an *independent set* in G is a set of vertices that are pairwise non-adjacent. The subgraph of G *induced by* a set U of vertices has U as its vertex set and contains all edges of G with both endpoints in U. An *arborescence* is a directed rooted tree with a unique path leading from the root to each non-root vertex.

B.2 Hypergraph Classes

Here we provide Figure 1 that describes the hierarchy of the various hypergraph classes studied in the paper. We also prove a folklore fact about the relationship between laminar and subpath hypergraphs.

Proposition B.1. *Laminar hypergraphs are subpath hypergraphs.*

Proof. Consider a laminar hypergraph $\mathcal{H} = (V, \mathcal{E})$. We say that a vertex $v \in V$ *belongs* to a given edge $e \in \mathcal{E}$ if $v \in e$ but there is no edge $f \subset e$, $f \in \mathcal{E}$ that contains v. We use a recursive procedure

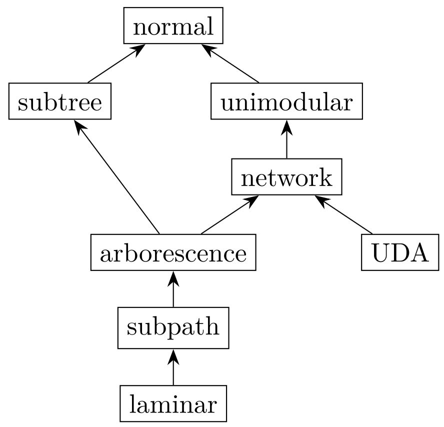

Figure 1: Connections between various classes of hypergraphs. An arrow from $\mathcal{A}$ to $\mathcal{B}$ means that hypergraphs with property $\mathcal{A}$ are a subset of hypergraphs with property $\mathcal{B}$. For the normality of subtree and unimodular hypergraphs, see [7, Section 4.4] and [5, Page 163], resp. In the figure, UDA stands for the class of hypergraphs that arise in our University Dual Admission problem, formally defined in Definition 3.2.

called $\mathsf{List}(\mathcal{H})$ to list the vertices of $\mathcal{H}$ in an ordering that all hyperedges contain consecutive vertices. From that it is easy to see that the hyperedges indeed correspond to some subpaths of a directed path over V.

Step 1. Take a maximal edge $e_{\max}$ in $\mathcal{H}$, and list all vertices belonging to $e_{\max}$.
Step 2. Let $\mathcal{H}'$ be the hypergraph obtained restricting $\mathcal{H}$ to all vertices that are contained in $e_{\max}$ but do not belong to $e_{\max}$.
Perform the recursive call $\mathsf{List}(\mathcal{H}')$.
Step 3. Let $\mathcal{H}''$ be the subhypergraph of $\mathcal{H}$ induced by all vertices that are not in $e_{\max}$.
Perform the recursive call $\mathsf{List}(\mathcal{H}'')$.

Since $\mathcal{H}$ is laminar, we know that $\mathcal{H}$ is the disjoint union of the hypergraph obtained by adding $e_{\max}$ to $\mathcal{H}'$ and hypergraph $\mathcal{H}''$. Hence, the procedure lists each vertex of $\mathcal{H}$ exactly once.

Consider any edge $e \in \mathcal{E}$, and observe that the first moment when a vertex of e is listed must be at the point where e is a maximal edge in Step 1. Then the procedure lists all vertices belonging to e, and then calls $\mathsf{List}(\mathcal{H}'_e)$ where $\mathcal{H}'_e$ contains exactly those edges in $\mathcal{E} \setminus \{e\}$ that are subsets of e. This means that during the call $\mathsf{List}(\mathcal{H}'_e)$, including all of its recursive calls, exactly those vertices are listed that are contained in e but do not belong to e.

Since no subset of e is taken as the maximal edge in Step 1 before e is taken, we know that the vertices listed up to this moment are not in e. This shows that the ordering of V created by $\mathsf{List}(\mathcal{H})$ yields an ordering, where e contains consecutive vertices. As this holds for each edge $e \in \mathcal{E}$, we obtain that $\mathcal{H}$ is an subpath hypergraph. □

Appendix C Additional Material for Section 3

We start off by describing an instance, where a Gale-Shapley type proposal-rejection algorithm cycles.

Example C.1. Consider an instance of UDA with students s_1, s_2, s_3, s_4, universities u_1, u_2 and programs $p_{11}, p_{12}, p_{13}, p_2 = p_{21}$. Let u_1 have capacity 2, u_2 have capacity 1, and let every program have quota 1. The preferences are as follows.

$s_1 : p_{11}$	$p_{11} : s_1 \succ s_3$	$u_1 : s_4 \succ s_3 \succ s_2 \succ s_1$
$s_2 : p_{12}$	$p_{12} : s_2$	$u_2 : s_3 \succ s_4$
$s_3 : p_{11} \succ p_2$	$p_{13} : s_4$	
$s_4 : p_2 \succ p_{13}$	$p_2 : s_3 \succ s_4$	

We consider a variant of the Gale-Shapley algorithm where students propose, and programs as well as universities may accept or reject these proposals. Let the proposing order of the students be s_3, s_4, s_1, s_2. Then in the first run of the Gale-Shapley algorithm, the proposals are $s_3 \to p_{11}$, $s_4 \to p_2$, $s_1 \to p_{11}$

(leading to the rejection of s_3), $s_2 \to p_{12}$, $s_3 \to p_2$ (leading to the rejection of s_4), $s_4 \to p_{13}$ (leading to the rejection of s_1 by u_1). After this first round of proposals, the current assignment is

$$\mu = \{(s_2, u_1, p_{12}), (s_3, u_2, p_2), (s_4, u_1, p_{13})\}$$

which still admits a blocking triple (s_3, u_1, p_{11}).

Thus, let us run the Gale-Shapley algorithm again, or more precisely, let us start another round of proposals where each student starts with μ as her current allocation, but the proposal pointers are reset. Then the proposals of the second round are: $s_3 \to p_{11}$ (leading to the rejection of s_2 by u_1), $s_4 \to p_2$, $s_1 \to p_{11}$ (leading to the rejection of s_3), $s_2 \to p_{12}$, $s_3 \to p_2$ (leading to the rejection of s_4), $s_4 \to p_{13}$ (leading to the rejection of s_1 by u_1). Hence, after the second iteration we arrive at the same (unstable) assignment μ. Hence, this variant of the Gale–Shapley algorithm never terminates on this instance.

Intuitively, the problem that makes UDA difficult is the following. Assume for simplicity that programs have capacity one. While a university u_i is unsaturated, then in case there is a demand for the same program by at least two students, then since we do not know in advance whether the university will be filled at the end, we must let the program decide on whom to reject. However, this can lead to rejecting the worse student for u_i. This can pose a problem, because if u_i gets saturated later and there is a demand for a free program at u_i by a student that u_i prefers, then we have to reject a student from some program p_{ik}. This can lead to some student s_j creating a blocking triple (s_j, u_i, p_{ik}), because as we explained earlier, s_j might have been necessarily rejected by the program. Finally, letting students propose again can easily lead to universities getting unsaturated again and thus to the algorithm cycling.

For the rest of the section, we first present all proofs that were omitted from Section 3 in Section C.1, and then proceed in Section C.2 with an ILP formulation for the UDA problem. Then, we describe the notion of *half-stability* and provide an algorithm to find a half-stable matching in Section C.3. Finally, we conclude in Section C.4 by showing that UDA is polynomial-time solvable if each university has capacity one.

C.1 Proofs omitted from Section 3

Proof of Theorem 3.3

Theorem 3.3 ($\star$). *The stable assignments of an instance of $I = (S, U, P, c, q, (\succ_a)_{a \in S\cup U\cup P})$ of* UDA *correspond bijectively to the stable $b^{\mathcal{H}}$-matchings of the instance $I' = (\mathcal{H}_I, b^{\mathcal{H}}, (\succ_a^{\mathcal{H}})_{a\in S\cup U\cup P})$ of* SHbM.

Proof. Suppose that μ is a stable assignment for I. We define a corresponding hypergraph $b^{\mathcal{H}}$-matching as $M = \{\{s_j, u_i, p_{ik}\} : s_j \in S, p_{ik} = \mu(s_j) \neq \perp\}$. To see that M is indeed a stable $b^{\mathcal{H}}$-matching for $\mathcal{H}_I$, assume for the sake of contradiction that a hyperedge $e = \{s_j, u_i, p_{ik}\}$ blocks M. This means that for each agent a incident to e, either a is unsaturated, or prefers e to some hyperedge $e' \in M$ incident to a. By the definition of $\succ^{\mathcal{H}}$, this implies that (i) s_j prefers p_{ik} to $\mu(s_j)$ (this allows $\mu(s_j) = \perp$), (ii) u_i is unsaturated in μ, or there is a student $s_{j'} \in \mu(u_i)$ such that $s_j \succ_{u_i} s_{j'}$, or $s_j \in \mu(u_i)$ (recall that u_i ranks hyperedges according to its preferences over the students, and ranks two hyperedges containing the same student according to the student's preferences), and (iii) p_{ik} is either unsaturated or there is a student $s_{j''} \in \mu(p_{ik})$ such that $s_j \succ_{p_{ik}} s_{j''}$. Hence, e satisfies the conditions in Definition 3.1 for blocking μ, contradiction.

For the other direction, assume that M is a stable hypergraph $b^{\mathcal{H}}$-matching for $\mathcal{H}_I$. Define the corresponding assignment μ for I by setting $\mu(s_j) = p_{ik}$ if M contains a hyperedge $\{s_j, u_i, p_{ik}\}$, and setting $\mu(s_j) = \perp$ if no hyperedge in M contains s_j; recall that $b^{\mathcal{H}}(s_j) = 1$, so there can be at most one hyperedge in M incident to s_j, so the assignment μ is well-defined. By the feasibility of M, we know that μ is an assignment that respects the capacities of the universities and the quotas of programs. It remains to show that a blocking triple for the assignment μ would also be a blocking hyperedge for M, by our definition of the extended preferences $\succ^{\mathcal{H}}$. So assume that the triple (s_j, u_i, p_{ik}) blocks the assignment μ.

Note first that since it blocks μ, either $\mu(s_j) = \perp$, or the preferences $\succ_{s_j}^{\mathcal{H}}$ are such that s_j prefers $e = \{s_,, u_i, p_{ik}\}$ to the hyperedge $e' = \{s_j, u_{i'}, \mu(s_j)\} \in M$ where $u_{i'}$ is the university offering the program $\mu(s_j)$. Hence, the condition on s_j in Definition 2.5 necessary for e to be a blocking hyperedge for M is satisfied. Second, if the university u_i is unsaturated or $\mu(u_i)$ contains a student $s_{j'}$ for which $s_j \succ_{u_i} s_{j'}$, then the condition on u_i in Definition 2.5 necessary for e to block M is satisfied as well, because $e \succ_{u_i}^{\mathcal{H}} e''$ holds for the hyperedge $e'' = \{s_{j'}, u_i, \mu(s_{j'})\} \in M$. For the case when $s_j \in \mu(u_i)$, recall that we know $p_{ik} \succ_{s_j} \mu(s_j)$, so by construction we get $e \succ_{u_i}^{\mathcal{H}} \{s_j, u_i, \mu(s_j)\} \in M$. Thus, u_i fulfills the necessary condition of Definition 2.5 for e to block M in all three cases. Third, the program p_{ik} is either unsaturated, or $e \succ^{\mathcal{H}}_{p_{ik}} e'''$ for some hyperedge $e''' = \{s_{j''}, u_i, p_{ik}\} \in M$ where $s_{j''}$ is the student whose existence is guaranteed by the third condition of Definition 3.1, using that the triple (s_j, u_i, p_{ik}) blocks μ. Therefore, we can conclude that e as a hyperedge blocks M, a contradiction proving that μ must be stable. ☐

Proof of Theorem 3.5

Theorem 3.5 ($\star$)**.** *Finding a maximum size stable assignment in an instance of* UDA *is* NP*-hard, even if all capacities are one.*

Proof. We reduce from the COM-SMTI problem, the problem of deciding whether an instance of STABLE MATCHING WITH TIES AND INCOMPLETE PREFERENCES admits a complete stable matching. The input instance I of COM-SMTI contains a set $V_w = \{w_1, \ldots, w_n\}$ of women, a set $V_m = \{m_1, \ldots, m_n\}$ of men, and a preference list $\succ_a$ for each person $a \in V_w \cup V_m$, which is a weak ordering over a subset of persons from the opposite sex. We will assume that each man m_i has a strict preference list, and each woman w_j has either a strict preference list, or her preference list is a tie of length two; it is known that COM-SMTI remains NP-hard even in this special case [19]. We may assume w.l.o.g. that the set of women whose preference list is a tie is $\{w_1, \ldots, w_\ell\}$.

Create an instance I' of UDA as follows. For each woman $w_i \in V_w$ we create a university u_i along with a program $p_i = p_{i1}$, where both the capacity of u_i and the quota of p_i are 1. For each man $m_j \in V_m$, we create a student s_j. The preference list of each student s_j is inherited from m_j, by replacing each woman w_i in the preference list of m_j with the program p_j. For some $i \in [\ell]$, there are two acceptable triples containing university u_i and program p_i, namely (s_k, u_i, p_i) and (s_l, u_i, p_i) where m_k and m_l $(k < l)$ are the two acceptable partners of w_i. The preferences of u_i and p_i $(i \in [\ell])$ are such that $s_k \succ_{u_i} s_l$ and $s_l \succ_{p_i} s_k$. For the remaining universities and programs, i.e., for $i \in [n] \setminus [\ell]$, the preference lists of u_i and p_i are the same and are inherited from w_i, by replacing each man m_j in the preference list of w_i with the student s_j. We set each capacity to be 1 and the desired size of a stable assignment for I' as $t = n$.

We claim that there is a complete stable matching in I if and only if there is a stable assignment matching of size $t = n$ in I'.

Let M be a complete stable matching in I. Let μ be the assignment obtained by setting $\mu(s_j) = p_i$ for each $\{w_i, m_j\} \in M$. As M is a matching covering each person exactly once, μ is a feasible assignment of size n (in which no agent is unsaturated). Suppose that there is a blocking triple (s_j, u_i, p_i) for μ. Then s_j prefers p_i to $\mu(s_j) = p_{i'}$, which means that m_j prefers w_i to $w_{i'} = M(s_j)$. Similarly, u_i and p_i both prefer s_j to the unique student assigned to p_i in μ. This implies that u_i and p_i must correspond to a woman w_i whose preference list is strict, as otherwise u_i and p_i would have opposite rankings over the students who apply there and one of them would prefer $\mu(p_i)$ to s_j. Thus, we obtain that w_i prefers m_j to $m_{j'} = M(w_i)$. This proves that $\{w_i, m_j\}$ blocks M, a contradiction to the stability of M. Thus, μ is stable as required.

Suppose now that we have a stable assignment μ of size n in I'. Observe that for each $i \in [n]$. Since all capacities in I' are 1, we can create a matching M in I by adding $\{w_i, m_j\}$ whenever $\mu(s_j) = p_i$. Clearly, M has size n, hence it is complete. Suppose that $\{m_j, w_i\}$ blocks M. Then w_i must have a strict preference list, and thus it is straightforward to check that the triple (s_j, u_i, p_i) blocks μ, a contradiction. Hence, M is stable, as required. ☐

C.2 An Integer Programming Formulation for UDA

Let $I = (S, U, P, c, q, (\succ_a)_{a \in S \cup U \cup P})$ be an instance of UDA. Without loss of generality, we may suppose that each program has quota one, as we can replace each program p with $q(p)$ clones $p^1, \ldots, p^{q(p)}$, each having capacity 1 and retaining the preferences of p; we then replace p with the list $p^1, \ldots, p^{q(p)}$ in the preference lists of students.

For convenience, we rely on the corresponding UNIMOD-SHbM instance $I' = (\mathcal{H}_I, b^{\mathcal{H}}, (\succ^{\mathcal{H}}_a)_{a \in S \cup U \cup P})$. Let $\mathcal{H}_I = (S \cup U \cup P, \mathcal{E}_I)$ be the hypergraph associated with I, and let $|\mathcal{E}_I| = m$.

(IP$_{\text{UDA}}$)
$$\sum_{e \ni s} x_e \le 1 \qquad \forall s \in S$$
$$\sum_{e \ni u} x_e \le c(u) \qquad \forall u \in U$$

$$\sum_{e \ni p} x_e \leq 1 \qquad \forall p \in P$$

$$c(u)\Bigg(\sum_{e' \succ_s^{\mathcal{H}} e} x_{e'} + \sum_{e' \succ_p^{\mathcal{H}} e} x_{e'} + x_e\Bigg) + \sum_{e' \succ_u^{\mathcal{H}} e} x_{e'} \geq c(u) \qquad \forall e = (s,u,p) \in \mathcal{E}_I$$

$$x \in \{0,1\}^m$$

Consider an arbitrary solution x to IP_{UDA}. Clearly, the first three inequalities in IP_{UDA} ensure that x corresponds to a $b^{\mathcal{H}}$-matching M_x in I' whose characteristic vector is x. By the last inequality, for each hyperedge $e = \{s,u,p\}$ it holds that either $e \in M_x$, or there is a hyperedge $e' \in M$ that preferred to e by either s or p, or $\sum_{e' \succ_u e} x_{e'} \geq c(u)$, so u is saturated in M_x with hyperedges that it prefers to e. Hence, M_x is stable.

Using the above observations, it is straightforward to verify that the stable matchings for $\mathcal{H}_I$ are exactly the solutions of IP_{UDA}. Therefore, as we have seen, this IP is guaranteed to have a solution.

By Theorem 3.3, M_x also gives a stable assignment μ_x to the UDA instance I.

C.3 A weaker version of stability for UDA

We can define stability in UDA in a slightly weaker, but still natural way. To avoid confusion with previous notions of weak and strong stability in the literature, we will call this relaxed notion *half-stability.*

Definition C.2. A feasible assignment μ is *half-stable* if there is no (s_j, u_i, p_{ik}) triple, such that

- (s_j, u_i, p_{ik}) blocks μ and
- if p_{ik} is saturated, then there is a student $s_{j'} \in \mu(p_{ik})$, who is hence also in $\mu(u_i)$, such that $s_j \succ_{u_i} s_{j'}$ and $s_j \succ_{p_{ik}} s_{j'}$.

If there is such a triple, then we call it *doubly blocking.*

The motivation behind this definition is that if there is a blocking triple (s_j, u_i, p_{ik}), and p_{ik} is saturated, then in order to accept s_j to p_{ik}, we have to drop a student from p_{ik}, hence also from u_i. However, if there is no student that is worse for both u_i and p_{ik}, either u_i or p_{ik} would have to loose a better student in exchange for s_j, so they may prefer not to participate in the blocking after all.

Of course, a stable assignment μ is also *half-stable*, hence a half-stable assignment always exists. We show that one can also be found in polynomial time.

Theorem C.3. *For an instance of* UDA *a half-stable assignment can be found in $\mathcal{O}(|\mathcal{E}|)$ time where $\mathcal{E}$ is the set of acceptable triples.*

Proof. We reduce this problem to the Student–Project Allocation (or spa) problem which can be solved in linear time due to the results of Abraham, Irving and Manlove [1]. This problem can be thought of as the special case of UDA where the preference ordering of each program p coincides with the preference ordering of the university offering p.

Given an instance I of UDA, we construct an instance I' of spa by simply setting the preferences of each program p_{ik} to be identical to $\succ_{u_i}$. A stable matching M in I', which can be found in linear time [1], is automatically a half-stable assignment μ in I as well.

To see this, assume for the sake of a contradiction that (s_j, u_i, p_{ik}) doubly blocks μ. Then s prefers p_{ik} to $\mu(s_j)$ (allowing the possibility that $\mu(s_j) = \perp$). By the stability of M in I', this means that either the program p_{ik} or the university u_i was saturated by students that were better than s_j according to $\succ_{u_i}$. If u_i is saturated by students better than s_j, then this contradicts our assumption that (s_j, u_i, p_{ik}) (doubly) blocks μ in I. Thus, it must be the case that p_{ik} is saturated with students that u_i prefers to s_j; however, then there is no student in $\mu(p_{ik})$ that is worse than s_j for both p_{ik} and u_i, contradicting our assumption that (s_j, u_i, p_{ik}) doubly blocks μ. □

C.4 Solving UDA with unit capacities

Consider an instance $I = (S, U, P, c, q, (\succ_v)_{v \in S \cup U \cup P})$, where $c(u) = 1$ for all universities $u \in U$. We show that the following simple version of the deferred acceptance algorithm by Gale and Shapley finds a stable matching for I.

Initially, every student is unassigned. While there exists an unassigned student, we do the following. Let s_j be an unassigned student, and let p_{ik} be the best program for s_j that has not rejected s_j yet. We let s_j propose to $p_{ik} \in P_i$. However, the university will be the one who decides whether s_j is rejected or not. That is, if there are no students at u_i, then s_j gets assigned to p_{ik}. If there is a student $s_{j'}$ such that $s_{j'} \succ_{u_i} s_j$, then s_j is rejected. Finally, if u_i has a student $s_{j'}$, but he is worse than s_j, then s_j gets assigned to p_{ik} and $s_{j'}$ gets rejected.

Let the assignment thus obtained be μ; we claim that μ is stable. Suppose for the contrary that a triple (s, u, p) blocks μ. Then, $p \succ_s \mu(p)$, so s has proposed to p, but got rejected. Hence, at that point u had a better student s' assigned. Since no student is rejected for another student that is worse for the university, it follows that u has a better student assigned in μ than s. Finally, as $c(u) = 1$, this implies that u is saturated with better students than s, a contradiction to (s, u, p) blocking μ.

The intuitive reason why the above algorithm works is that for unit capacities stability coincides with half-stability. Indeed, if (s_j, u_i, p_{ik}) is a triple that blocks a matching μ for I and p_{ik} is saturated by μ, then by $c(u_i) = 1$ there must be a unique student $s_{j'}$ assigned to p_{ik} and to u_i, and since (s_j, u_i, p_{ik}) blocks μ, it immediately follows that $s_j \succ_{u_i} s_{j'}$ and $s_j \succ_{p_{ik}} s_{j'}$, i.e., the triple (s_j, u_i, p_{ik}) doubly blocks μ. This means that half-stability implies stability and, hence, Theorem C.3 has the following consequence.

Corollary C.4. *For an instance of* UDA *with unit capacities, a stable assignment can be found in* $O(|\mathcal{E}|)$ *time where* $\mathcal{E}$ *is the set of acceptable triples.*

Appendix D Additional Material for Section 4

We first present all omitted proofs from Sections 4.1 and 4.2 in Sections D.1 and D.2, respectively.

D.1 Omitted proofs from Section 4.1

Proof of Theorem 4.3

Theorem 4.3 ($\star$)**.** *The* MaxW-Laminar-SHM *problem is* NP*-complete on laminar hypergraphs, even if all edges have weight* 0 *with the exception of a single edge of weight* 1*.*

Proof. It is clear that the problem is in NP. To prove its NP-hardness, we present a reductions from the CNF-SAT problem. The input of CNF-SAT is a CNF formula φ defined over variables $x_1, \dots, x_n$ and containing clauses $c_1, \dots, c_m$; the task is to decide whether φ is satisfiable.

Construction. We are going to define a hypergraph $\mathcal{H}$ with vertex set $V = \{z\} \cup \{v_i, \bar{v}_i : i \in [n]\}$ and edge set $\mathcal{E} = \{e_z\} \cup E_V \cup F$ where $E_V = \{e_i, \bar{e}_i : i \in [n]\}$ and $F = \{f_j : j \in [m]\}$. For convenience, we allow our hypergraph $\mathcal{H}$ to contain parallel edges, so we will use an incidence function $\psi : \mathcal{E} \to 2^V$ to define $\mathcal{H}$ as follows:

$\psi(e_z) = \{z\}$;
$\psi(e_i) = \psi(\bar{e}_i) = \{v_i, \bar{v}_i\}$ for each $i \in [n]$;
$\psi(f_j) = V$ for each $j \in [m]$.

We define the edge sets $F_i^+ = \{f_j : c_j \text{ contains the literal } x_i\}$ and $F_i^- = \{f_j : c_j \text{ contains the literal } \bar{x}_i\}$ for each variable x_i. The preferences associated with the vertices of $\mathcal{H}$ are defined below; a set in the preferences denotes any arbitrarily fixed ordering of the set.

$z : F \succ e_z$;
$v_i : F \setminus F_i^+ \succ e_i \succ F_i^+ \succ \bar{e}_i$;
$\bar{v}_i : F \setminus F_i^- \succ \bar{e}_i \succ F_i^- \succ e_i$.

We set the weight function w to be 0 on every hyperedge except for e_z, and we set $w(e_z) = 1$. Moreover, we set our threshold (the weight of the desired stable matching) as $t = 1$. This finishes our instance $I_{\mathcal{H}}$ of MaxW-Laminar-SHM. It is clear that the hypergraph $\mathcal{H}$ is laminar, as promised.

Correctness. Observe that weight of a stable matching reaches the threshold t in $I_{\mathcal{H}}$ if and only if it contains the hyperedge e_z. We claim that $I_{\mathcal{H}}$ admits a stable hypergraph containing e_z if and only if φ can be satisfied.

First assume that a stable matching M in $I_{\mathcal{H}}$ contains e_z. Clearly, $M \cap F = \emptyset$, since every edge in F is incident to $z \in \psi(e_z)$. Moreover, $|M \cap \{e_i, \bar{e}_i\}| = 1$ for each $i \in [n]$, as otherwise both v_i and $\bar{v}_i$ would be unmatched (by $M \cap F = \emptyset$), and thus e_i (and also $\bar{e}_i$) would block M. We construct a truth assignment α_M for φ by setting $\alpha_M(x_i) = \texttt{true}$ if $e_i \in M$, and $\alpha_M(x_i) = \texttt{false}$ if $\bar{e}_i \in M$.

To see that α_M satisfies φ, let us consider any clause c_j. Since z prefers f_j to e_w, but f_j cannot block M, there must be some vertex $v \in \psi(f_j) \setminus \{z\}$ and an edge $e \in M$ incident to v that is preferred by v to f_j. First, if $v = v_i$ for some i, then the edge of M preferred by v_i to f_j must be e_i (since $\bar{e}_i$ is the edge least preferred by v_i). Hence, $\alpha(x_i) = \texttt{true}$. Moreover, since v_i prefers e_i to f_j, it must be that $f_j \in F_i^+$, that is, the clause c_j contains x_i as a positive literal. Thus, α_M makes c_j true. Similarly, if $v = \bar{v}_i$ for some i, then the edge of M preferred by $\bar{v}_i$ to f_j must be $\bar{e}_i$ (since e_i is the edge least preferred by $\bar{v}_i$). Hence, $\alpha(x_i) = \texttt{false}$. Moreover, since $\bar{v}_i$ prefers $\bar{e}_i$ to f_j, it must be that $f_j \in F_i^-$, that is, the clause c_j contains the literal $\bar{x}_i$. Thus, α_M again makes c_j true. Since this holds for each clause c_j of φ, we get that α_M satisfies φ.

For the other direction, assume now that α is a truth assignment that satisfies φ. Let

$$M_\alpha = \{e_z\} \cup \{e_i : \alpha(x_i) = \texttt{true}\} \cup \{\bar{e}_i : \alpha(x_i) = \texttt{false}\}.$$

Clearly, the edges in M_α cover each vertex of V exactly once, so M_α is indeed a matching. To see that M_α is stable as well, first observe that no edge of the form e_i or $\bar{e}_i$ can block M_α, as any such edge is the least-preferred edge by some vertex in V. Therefore we only have to prove that no edge in F blocks M_α. Consider some $f_j \in F$. Since c_j contains a literal that is set to $\texttt{true}$ by α, there must exist either some $e_i \in M_\alpha$ for which $c_j \in F_i^+$, or some $\bar{e}_i \in M_\alpha$ for which $c_j \in F_i^-$. In the former case, f_j does not block M_α because v_i prefers $e_i \in M_\alpha$ to $f_j \in F_i^+$, and in the latter case, f_j does not block M_α because $\bar{v}_i$ prefers $\bar{e}_i \in M_\alpha$ to $f_j \in F_i^-$. This proves that M_α is stable in $I_{\mathcal{H}}$ and our reduction is correct. ☐

D.2 Omitted proofs from Section 4.2

Proof of Lemma 4.7

Lemma 4.7 ($\star$). *If Algorithm 3 puts a pair (σ, M) into S_i for some $i \in [m]$, then M realizes σ with respect to $\mathcal{E}_i$.*

Proof. We use induction on i. The case $i = 0$ is trivial, so assume that the lemma holds for $i - 1$.

First, suppose that (σ, M) was added to S_i on line 14, so $M \subseteq \mathcal{E}_{i-1}$. Let $(\sigma^-, M) \in S_{i-1}$ be the pair selected on line 4 in the cycle during which (σ, M) was put into T_i on line 10. By induction,

(a) M is compatible with σ^-,
(b) $\sigma^-(v) \in M(v)$ for every $v \in L_{i-1}$ where $\sigma^-(v) \in \mathcal{E}_{i-1}$, and
(c) every edge in $\mathcal{E}_{i-1} \setminus M$ that blocks M is planned to be dominated by σ^-.

First, observe that the b-matching M is compatible with σ: the compatibility conditions for each $v \in L_{i-1} \cap L_i$ follow immediately from $\sigma^-(v) = \sigma(v)$ due to (a), while the compatibility condition for each $v \in L_i \setminus L_{i-1}$ holds irrespective of the value of $\sigma(v)$, because edges of $M \subseteq \mathcal{E}_{i-1}$ have no vertices in $L_i \setminus L_{i-1}$.

Second, by our assumption that Algorithm 3 added (σ, M) to S_i on line 14, we know that $\sigma(v) \neq e_i$ for each v incident to e_i. Hence, if $\sigma(v) \in \mathcal{E}_i$ for some $v \in L_i$, then $\sigma(v) \in \mathcal{E}_{i-1}$, and consequently, $v \in L_i \cap L_{i-1}$ (because vertices of $L_i \setminus L_{i-1}$ are not contained in any edge of $\mathcal{E}_{i-1}$). Therefore, (b) implies $\sigma(v) = \sigma^-(v) \in M(v)$. Third, it remains to show that every edge $e \in \mathcal{E}_i \setminus M$ that blocks M is planned to be dominated by σ. First note that Algorithm 3 checks that this holds for e_i explicitly on line 13, so we may assume $e \neq e_i$. By (c), we know that every edge in $\mathcal{E}_{i-1} \setminus M$ that blocks M is planned to be dominated by σ^-, so there is some $v \in L_{i-1}$ such that $\sigma^-(v) \succ_v e$. On the one hand, if $v \in L_i \cap L_{i-1}$, then e is planned to be dominated by σ too, because $\sigma(v) = \sigma^-(v)$. On the other hand, if $v \in L_{i-1} \setminus L_i$, then Algorithm 3 must have confirmed on line 10 that M is complete on v (otherwise it would not have put (σ^-, M) into T_i), which implies that M dominates e at v. Thus, e does not block M, proving that every edge in $\mathcal{E}_i \setminus M$ that blocks M is planned to be dominated by σ. This finishes the proof that M realizes σ with respect to $\mathcal{E}_i$.

Suppose now that (σ, M) was put into S_i on line 17. Let $M^- = M \setminus \{e_i\}$, and let $(\sigma^-, M^-) \in S_{i-1}$ be the pair selected on line 4 in the cycle during which (σ, M^-) was put into T_i on line 10. By induction, conditions (a)–(c) hold with M^- replacing M; let these modified conditions be referred to as (a')–(c'). First, by (a') and the conditions checked on line 15 we get that $M = M^- \cup \{e_i\}$ is a b-matching that is compatible with σ on L_i. Second, if $\sigma(v) \in \mathcal{E}_i$ for some $v \in L_i$, then either $\sigma(v) \in \mathcal{E}_{i-1}$, in which case $\sigma(v) = \sigma^-(v) \in M^-(v) \subseteq M(v)$ by (b'), or $\sigma(v) = e_i \in M(v)$. Hence, it remains to see that every edge $e \in \mathcal{E}_i \setminus M = \mathcal{E}_{i-1} \setminus M^-$ that blocks M is planned to be dominated by σ. First note that e must block also M^-, and thus by (c') must be planned to be dominated by σ^- at some $v \in L_{i-1}$. We can argue exactly as for previous case to prove that either e is planned to be dominated by σ at v (in case $v \in L_i$) or e is dominated by M at v (in case $v \notin L_i$). Thus, again it holds that M realizes σ with respect to $\mathcal{E}_i$. ◻

Proof of Theorem 4.10

Theorem 4.10 ($\star$). *The* MaxW-Subpath-SHbM *problem is* W[1]*-hard when parameterized by* $b_{\max} + \ell_{\max}$*, where* $b_{\max}$ *is the maximum capacity of a vertex and* $\ell_{\max}$ *is the maximum size of an edge, even if all edges have weight* 0 *with the exception of a single edge of weight* 1*.*

Proof. We are going to present a reduction from Multicolored Clique which is known to be W[1]-hard [11]. The input of Multicolored Clique is a graph $G = (V, E)$ and integer k with the vertex set of G is partitioned into k independent sets $V_1, \dots, V_k$; the task is to decide whether G admits a clique of size k.

Construction. We first construct a *selection gadget* which involves vertices $w, z, s_1, \dots, s_k$ and edge set $\{e_w, d_1, \dots, d_k\} \cup \{e^0(v) : v \in V\}$. We will also need *incidence gadgets* which will involve a vertex set $\{a_e, a'_e : e \in \bar{E}\}$ and edge set $\{f_e : e \in \bar{E}\}$ where $\bar{E}$ contains all edges in the complement of G running between different partitions. We fix an ordering over $\bar{E}$, and denote its elements by $e_1, \dots, e_{\bar{m}}$. We will also need *repeater gadgets* which will contain vertices $\{r^h_{(i,e)} : h \in [4], i \in [k], e \in \bar{E}\}$ and edge set $\{e^j(v), \hat{e}^j(v), \widetilde{e}^j(v) : e_j \in \bar{E}, i \in [k], v \in V_i\}$.

The incidence function ψ of the edges is as follows:

$$
\begin{aligned}
\psi(e_w) &= \{w, z\}; && \\
\psi(d_i) &= \{z, s_1, \dots, s_i\} && \text{for } i \in [k]; \\
\psi(e^0(v)) &= \{s_i, \dots, s_k\} \cup \{r^h_{(i',e_1)} : h \in [4], i' \in [i-1]\} && \\
&\quad \cup \{r^1_{(i,e_1)}, r^2_{(i,e_1)}, r^3_{(i,e_1)}\} && \text{for } v \in V_i,\ i \in [k]; \\
\psi(e^j(v)) &= \{r^2_{(i,e_j)}, r^3_{(i,e_j)}, r^4_{(i,e_j)}\} \cup \{r^h_{(i',e_j)} : h \in [4], i' \in [k] \setminus [i]\} && \\
&\quad \cup \{a_{e_j}, a'_{e_j}\} \cup \{r^h_{(i',e_{j+1})} : h \in [4], i' \in [i-1]\} && \\
&\quad \cup \{r^1_{(i,e_{j+1})}, r^2_{(i,e_{j+1})}, r^3_{(i,e_{j+1})}\} && \text{for } v \in V_i,\ i \in [k],\ j \in [\bar{m}-1]; \\
\psi(e^j(v)) &= \{r^2_{(i,e_j)}, r^3_{(i,e_j)}, r^4_{(i,e_j)}\} \cup \{r^h_{(i',e_j)} : h \in [4], i' \in [k] \setminus [i]\} && \\
&\quad \cup \{a_{e_j}, a'_{e_j}\} && \text{for } v \in V_i,\ i \in [k],\ j = \bar{m}; \\
\psi(\hat{e}^j(v)) &= \{r^1_{(i,e_j)}, r^2_{(i,e_j)}\} && \text{for } v \in V,\ i \in [k],\ j \in \bar{m}; \\
\psi(\widetilde{e}^j(v)) &= \{r^3_{(i,e_j)}, r^4_{(i,e_j)}\} && \text{for } v \in V,\ i \in [k],\ j \in \bar{m}; \\
\psi(f_e) &= \{a_e, a'_e\} && \text{for } e \in \bar{E}.
\end{aligned}
$$

It is straightforward to check that the following ordering of the vertices in the resulting hypergraph $\mathcal{H}$ witnesses its subpath property. We start with the vertices of the selection gadget as $w, z, s_1, \dots, s_k$, and then proceed with a *segment* for each e_j, $j = 1, \dots, \bar{m}$, where the segment for e is defined as the list

$$r^1_{(1,e)}, r^2_{(1,e)}, r^3_{(1,e)}, r^4_{(1,e)}, \dots, r^1_{(k,e)}, r^2_{(k,e)}, r^3_{(k,e)}, r^4_{(k,e)}, a_e, a'_e.$$

We set the capacities of the vertices as follows:

$$
\begin{aligned}
& b(w) = b(z) = 1; && \\
& b(s_i) = i && \text{for each } i \in [k]; \\
& b(r^1_{(i,e)}) = b(r^4_{(i,e)}) = c(a_e) = b(a'_e) = k && \text{for each } i \in [k],\ e \in \bar{E}; \\
& b(r^2_{(i,e)}) = b(r^3_{(i,e)}) = k+1 && \text{for each } i \in [k],\ e \in \bar{E}.
\end{aligned}
$$

Next, before defining the preferences of the vertices in $\mathcal{H}$, we need some additional notation. For convenience, for each $j = 0, 1, \dots, \bar{m}$ and $i \in [k]$ we define hyperedge sets $\mathcal{E}^j_i = \{e^j(v) : v \in V_i\}$, $\hat{\mathcal{E}}^j_i = \{\hat{e}^j(v) : v \in V_i\}$, and $\widetilde{\mathcal{E}}^j_i = \{\widetilde{e}^j(v) : v \in V_i\}$. Let us also fix an arbitrary ordering over each V_i, and denote the resulting ordering of V_i and the corresponding ordering of the three sets just defined by $\underrightarrow{V}_i$, $\underrightarrow{\mathcal{E}}^j_i$, $\underrightarrow{\hat{\mathcal{E}}}^j_i$, and $\underrightarrow{\widetilde{\mathcal{E}}}^j_i$, respectively. The reverse ordering of these sets will be denoted by $\underleftarrow{V}_i$, $\underleftarrow{\mathcal{E}}^j_i$, $\underleftarrow{\hat{\mathcal{E}}}^j_i$, and $\underleftarrow{\widetilde{\mathcal{E}}}^j_i$, respectively. Furthermore, we write $\mathcal{E}^j_{<i} = \bigcup_{i' \in [i-1]} \mathcal{E}^j_{i'}$ and $\mathcal{E}^j_{>i} = \bigcup_{i' \in [k] \setminus [i]} \mathcal{E}^j_{i'}$. We let $\mathcal{E}^j = \bigcup_{i \in [k]} \mathcal{E}^j_i$, and we define $\hat{\mathcal{E}}^j$ and $\widetilde{\mathcal{E}}^j$ analogously.

We also define $\bar{E}_{i,i'}$ to contain those edges xy of $\bar{E}$ for which $x \in V_i$ and $y \in V_{i'}$. For two series $A = (a_1, \dots, a_n)$ and $B = (b_1, \dots, b_n)$ of the same length let us denote the merging of A and B as

$A \mathbin{\wedge\!\!\wedge} B = (a_1, b_1, \ldots, a_n, b_n)$. Now we are ready to define the preferences, yielding an instance $I_{\mathcal{H}}$ of SHbM; again, sets in the preferences are ordered arbitrarily.

$$
\begin{array}{rll}
w & : e_w; & \\
z & : \{d_1, \ldots, d_k\} \succ e_w; & \\
s_i & : \{d_{i+1}, \ldots, d_k\} \succ \mathcal{E}^0_{<i} \succ \mathcal{E}^0_i \succ d_i & \text{for } i \in [k]; \\
r^1_{(i,e_j)} & : \mathcal{E}^{j-1}_{<i} \cup \mathcal{E}^{j-1}_{>i} \succ \underrightarrow{\mathcal{E}}^{j-1}_i \mathbin{\wedge\!\!\wedge} \underrightarrow{\hat{\mathcal{E}}}^j_i & \text{for } i \in [k],\ j \in [\bar{m}]; \\
r^2_{(i,e_j)} & : \mathcal{E}^{j-1} \succ \underleftarrow{\hat{\mathcal{E}}}^j_i \mathbin{\wedge\!\!\wedge} \underleftarrow{\mathcal{E}}^j_i & \text{for } i \in [k],\ j \in [\bar{m}]; \\
r^3_{(i,e_j)} & : \mathcal{E}^{j-1}_{<i} \cup \mathcal{E}^{j-1}_{>i} \cup \mathcal{E}^j_i \succ \underleftarrow{\widetilde{\mathcal{E}}}^j_i \mathbin{\wedge\!\!\wedge} \underleftarrow{\mathcal{E}}^{j-1}_i & \text{for } i \in [k],\ j \in [\bar{m}]; \\
r^4_{(i,e_j)} & : \mathcal{E}^{j-1}_{<i} \cup \mathcal{E}^{j-1}_{>i} \succ \underrightarrow{\mathcal{E}}^j_i \mathbin{\wedge\!\!\wedge} \underrightarrow{\widetilde{\mathcal{E}}}^j_i & \text{for } i \in [k],\ j \in [\bar{m}]; \\
a_{e_j} & : \mathcal{E}^j \setminus \{e^j(x)\} \succ f_{e_j} \succ e^j(x) & \text{for } e_j = \{x, y\} \in \bar{E}_{i,i'},\ i < i'; \\
a'_{e_j} & : \mathcal{E}^j \setminus \{e^j(y)\} \succ f_{e_j} \succ e^j(y) & \text{for } e_j = \{x, y\} \in \bar{E}_{i,i'},\ i < i'.
\end{array}
$$

To finish our instance of MaxW-SHbM, we set the weight of e_w as 1, and we set the weight of every other edge as 0; moreover, we set $t = 1$ as the weight of the desired stable b-matching. Clearly, a b-matching has weight at least $t = 1$ if and only if contains the edge e_w.

Notice that the maximum capacity is $b_{\max} = k+1$, and the maximum size of any edge is $\ell_{\max} = 4k+4$, the size of any edge $e^j(v)$ with $1 \leq j < \bar{m}$. It remains to show that our reduction is correct, i.e., e_w is contained in some stable b-matching of $I_{\mathcal{H}}$ if and only if G admits a clique of size k.

Direction "⇒": Suppose first that M is a stable b-matching for $I_{\mathcal{H}}$ with $e_w \in M$. Clearly, any edge not in M must be dominated at some vertex.

Note that since z prefers each of the edges $d_1, \ldots, d_k$ to e_w and $b(z) = 1$, it must be the case for each $i \in [k]$ such that $d_i \notin M$ and d_i is dominated at some vertex other than z. By $\psi(d_1) = \{z, s_1\}$ we know that d_1 must be dominated at s_1; by $b(s_1) = 1$ this implies $|\mathcal{E}^0_1 \cap M| = 1$. More generally, observe that d_i for some $i \in [k]$ can only be dominated at s_i (as the vertices $s_1, \ldots, s_{i-1}$ do not prefer any edge that might be in M to d_i), implying that $|\mathcal{E}^0_i \cap M| = 1$. Let $q_1, \ldots, q_k$ be the vertices in G such that $e^0(q_i)$ is the unique edge in $\mathcal{E}^0_i \cap M$. Our aim is to show that the vertices $q_1, \ldots, q_k$ form a clique in G.

Claim D.1. *For each $j \in [\bar{m}]$ and $i \in [k]$ it holds that (i) $|\mathcal{E}^j_i \cap M| = 1$, and (ii) $\mathcal{E}^j_i \cap M = \{e^j(q_i)\}$.*

Proof of Claim. By the paragraph preceding the claim, both statements hold for $j = 0$. We prove our claim by induction, so assume that (i) and (ii) hold for $j - 1$.

Fix an index $i \in [k]$. Let $\underrightarrow{V}_i = (v_1, \ldots, v_{|V_i|})$, and assume that q_i is the p-th vertex in this order, so by (ii) we have $\mathcal{E}^{j-1}_i \cap M = \{e^{j-1}(v_p)\}$. Observe that none of the edges $\hat{e}^j(v_1), \ldots, \hat{e}^j(v_{p-1})$ is dominated at the vertex $r^1_{(i,e_j)}$. Since $r^1_{(i,e_j)}$ is incident to k edges in $M \cap \mathcal{E}^{j-1}$ by (i) and the capacity of $r^1_{(i,e_j)}$ is k, we know that $\hat{\mathcal{E}}^j_i \cap M = \emptyset$. Therefore, all of the edges $\hat{e}^j(v_1), \ldots, \hat{e}^j(v_{p-1})$ must be dominated at $r^2_{(i,e_j)}$. By the preferences of $r^2_{(i,e_j)}$ (and using $\hat{\mathcal{E}}^j \cap M = \emptyset$), it follows that M must contain an edge in $\mathcal{E}^j_i$ that precedes $\hat{e}^j(v_{p-1})$ in $\underleftarrow{\hat{\mathcal{E}}}^j_i \mathbin{\wedge\!\!\wedge} \underleftarrow{\mathcal{E}}^j_i$. By definition, this means that M contains $e^j(v_{p'})$ for some $p' > p - 1$. By $b(r^2_{(i,e_j)}) = k + 1$, this also yields that $r^2_{(i,e_j)}$ is saturated by M, so $|M \cap \mathcal{E}^j_i| = 1$.

Consider now the vertex $r^3_{(i,e_j)}$ and its capacity of $k+1$. We already know that $|M \cap (\mathcal{E}^{j-1} \cup \mathcal{E}^j_i)| = k+1$, and since every edge in $\mathcal{E}^{j-1} \cup \mathcal{E}^j_i$ is incident to $r^3_{(i,e_j)}$, we know that $M \cap \widetilde{\mathcal{E}}^j_i = \emptyset$. Thus, every edge $\widetilde{e}^j(v_h) \in \widetilde{\mathcal{E}}^j_i$ must be dominated either at $r^3_{(i,e_j)}$ or at $r^4_{(i,e_j)}$. In the former case, $e^{j-1}(v_p)$ precedes $\widetilde{e}^j(v_h)$ in $\underleftarrow{\widetilde{\mathcal{E}}}^j_i \mathbin{\wedge\!\!\wedge} \underleftarrow{\mathcal{E}}^{j-1}_i$, meaning $h < p$. In the latter case, $e^j(v_{p'})$ precedes $\widetilde{e}^j(v_h)$ in $\underrightarrow{\mathcal{E}}^j_i \mathbin{\wedge\!\!\wedge} \underrightarrow{\widetilde{\mathcal{E}}}^j_i$, meaning $p' \leq h$. We can conclude that $h < p$ or $p' \leq h$ holds for *every* $h \in [|V_i|]$, in particular, for $h = p$ this yields $p' \leq p$. Taking into account that we proved $p' \geq p$, we get $p' = p$. Hence, we have proved that $\mathcal{E}^j_i \cap M = \{e^j(v_p)\} = \{e^j(q_i)\}$, finishing our proof of the claim. ◁

Using Claim D.1, we can now show that the vertices $q_1, \ldots, q_k$ form a clique in G. Suppose for the sake of contradiction that q_{i_1} and q_{i_2} are not adjacent in G for some $i_1 < i_2$. Then $\bar{E}$ contains the edge $\{q_{i_1}, q_{i_2}\}$, let $e_j = \{q_{i_1}, q_{i_2}\}$. Consider the vertices a_{e_j} and a'_{e_j}. By our claim, we know that they are incident to exactly k edges in $M \cap \mathcal{E}^j$, and since they have capacity k, it follows that the edge f_{e_j} (incident to both a_{e_j} and a'_{e_j}) cannot be contained in M. Hence, it must be dominated either at a_{e_j} or at a'_{e_j}. The former happens exactly if $e^j(q_{i_1}) \notin (M \cap \mathcal{E}^j)$, while the latter happens exactly if $e^j(q_{i_2}) \notin (M \cap \mathcal{E}^j)$. However, by definition, neither of these holds, a contradiction.

Direction "⇐": Suppose now that there is a clique of size k in G, containing a vertex q_i from each partition V_i. We define the b-mathing $M = \{e_w\} \cup \{e^j(q_i) : i \in [k], j \in \{0, 1, \dots, \bar{m}\}\}$. It is easy to check that M saturates all vertices of $\mathcal{H}$.

It suffices to show that M is stable. To this end, we argue that no edge of $\mathcal{H}$ blocks M:

- Edges in $\{d_1, \dots, d_k\}$: It is clear that any edge d_i is dominated by M at s_i, so cannot block M.
- Edges in $\hat{\mathcal{E}}^j$ for some $j \in [\bar{m}]$: Consider an edge $\hat{e}^j(v) \in \hat{\mathcal{E}}^j$ with $v \in V_i$. On the one hand, if q_i weakly precedes v in $\underrightarrow{V}_i$ (allowing $q_i = v$), then $\hat{e}^j(v)$ is dominated by M at $r^1_{(i,e_j)}$: indeed, $r^1_{(i,e_j)}$ prefers each of the $k-1$ vertices in $M \cap (\mathcal{E}^{j-1}_{<i} \cup \mathcal{E}^{j-1}_{>i})$ to $\hat{e}^j(v)$, and since $e^{j-1}(q_i)$ precedes $\hat{e}^j(v)$ in $\underrightarrow{\mathcal{E}}^{j-1}_i \wedge\!\!\!\wedge \underrightarrow{\hat{\mathcal{E}}}^j_i$, we get that $r^1_{(i,e_j)}$ prefers $e^{j-1}(q_i)$ to $\hat{e}^j(v)$ as well. On the other hand, if q_i follows v in $\underrightarrow{V}_i$, then $\hat{e}^j(v)$ is dominated by M at $r^2_{(i,e_j)}$: again, $r^2_{(i,e_j)}$ prefers each of the k vertices in $M \cap \mathcal{E}^{j-1}$ to $\hat{e}^j(v)$, and since $e^{j-1}(q_i)$ precedes $\hat{e}^j(v)$ in $\underleftarrow{\hat{\mathcal{E}}}^j_i \wedge\!\!\!\wedge \underleftarrow{\mathcal{E}}^j_i$, we get that $r^2_{(i,e_j)}$ prefers $e^{j-1}(q_i)$ to $\hat{e}^j(v)$ as well.
- Edges in $\widetilde{\mathcal{E}}^j$ for some $j \in [\bar{m}]$: Consider an edge $\widetilde{e}^j(v) \in \hat{\mathcal{E}}^j$ and $v \in V_i$. We argue similarly as in the previous case. If q_i weakly precedes v in $\underrightarrow{V}_i$, then $\widetilde{e}^j(v)$ is dominated by M at $r^4_{(i,e_j)}$, otherwise it is dominated by M at $r^3_{(i,e_j)}$.
- Edges in $\{f_{e_j} : e_j \in \bar{E}\}$: Let e_j run between V_{i_1} and V_{i_2} in the complement of G, with $i_1 < i_2$. Since e_j is not an edge of G, at least one of its endpoints does not belong to the clique on $\{q_1, \dots, q_k\}$. If this endpoint is a vertex in $V_{i_1} \setminus \{q_{i_1}\}$, then f_{e_j} is dominated at a_{e_j} by the edges $\{e^j(q_i) : i \in [k]\}$. By contrast, if e_j has an endpoint in $V_{i_2} \setminus \{q_{i_2}\}$, then f_{e_j} is dominated at a'_{e_j}.
- Edges in $\mathcal{E}^j$ for some $j \in \{0, 1, \dots, \bar{m}\}$: Suppose that $v \in V_i$, and consider the edge $e^j(v)$. On the one hand, if v precedes q_i in $\underrightarrow{V}_i$, then $e^j(v)$ is dominated at $r^2_{(i,e_j)}$ (in case $j \in [\bar{m}]$) and also at $r^3_{(i,e_{j+1})}$ (in case $0 \le j < \bar{m}$). On the other hand, if v follows q_i in $\underrightarrow{V}_i$, then $e^j(v)$ is dominated at $r^1_{(i,e_j)}$ (in case $0 \le j < \bar{m}$) and also at $r^4_{(i,e_{j+1})}$ (in case $j \in [\bar{m}]$). ☐

Appendix E Omitted proofs from Section 5

Proof of Theorem 5.2

Theorem 5.2 (⋆)**.** *Deciding if there exists an stable hypergraph b-matching in an instance of* Subtree-SHbM *is* NP*-hard, even if the underlying tree is a star, only one vertex has capacity greater than 1, and all hyperedge sizes are at most 4.*

Proof. We give a reduction from SHbM, which is NP-complete even if each hyperedge has size 3 [20]. Let $I = (\mathcal{H}, b, (\succ_v)_{v \in V})$ be such an instance of SHbM with hypergraph $\mathcal{H} = (V, \mathcal{E})$. We create a new instance $I' = (\mathcal{H}', b', (\succ'_v)_{v \in V})$ of Subtree-SHbM satisfying the conditions in the theorem. Let the vertices of $\mathcal{H}'$ be the vertices of $\mathcal{H}$ with one additional vertex x, and let us extend each hyperedge in $\mathcal{E}$ by adding x. That is, $\mathcal{H}' = (V \cup \{x\}, \mathcal{E}')$ where $\mathcal{E}' = \{e \cup \{x\} : e \in \mathcal{E}\}$. Notice that $\mathcal{H}'$ is indeed a subtree hypergraph whose underlying tree is a star on $V \cup \{x\}$ with x as its center. For each vertex $v \in V$, we retain both its capacity and its preferences, i.e., $b'(v) = b(v)$ and $\succ'_v = \succ_v$. For the vertex x, we set $b'(x) = |\mathcal{E}| + 1$ and fix its preferences arbitrarily. Note that x can never be saturated in any b-matching $M \subseteq \mathcal{E}'$, so x cannot dominate any edge for any b-matching.

It is now straightforward to check that the stable b-matchings in I correspond bijectively to the stable b-matchings in I'. More precisely, for a stable b-matching M' in I', the projection of M' to V yields a stable b-matching in I. Similarly, given a stable b-matching M in I, the b-matching obtained by adding x to each hyperedge in M is a stable b-matching in I'. ☐