New and simple algorithms for stable flow problems*

Ágnes Cseh[1] and Jannik Matuschke[2]

[1] Institute of Economics, Hungarian Academy of Sciences and Corvinus University of Budapest, Budaörsi út 45., 1112 Budapest, Hungary
`cseh.agnes@krtk.mta.hu`
[2] TUM School of Management and Department of Mathematics, Technische Universität München, Arcisstraße 21, 80333 Munich, Germany
`jannik.matuschke@tum.de`

Abstract. Stable flows generalize the well-known concept of stable matchings to markets in which transactions may involve several agents, forwarding flow from one to another. An instance of the problem consists of a capacitated directed network, in which vertices express their preferences over their incident edges. A network flow is stable if there is no group of vertices that all could benefit from rerouting the flow along a walk.
Fleiner [13] established that a stable flow always exists by reducing it to the stable allocation problem. We present an augmenting-path algorithm for computing a stable flow, the first algorithm that achieves polynomial running time for this problem without using stable allocation as a black-box subroutine. We further consider the problem of finding a stable flow such that the flow value on every edge is within a given interval. For this problem, we present an elegant graph transformation and based on this, we devise a simple and fast algorithm, which also can be used to find a solution to the stable marriage problem with forced and forbidden edges.
Finally, we study the highly complex stable multicommodity flow model by Király and Pap [24]. We present several graph-based reductions that show equivalence to a significantly simpler model. We further show that it is NP-complete to decide whether an integral solution exists.

1 Introduction

Stability is a well-known concept used for matching markets where the aim is to reach a certain type of social welfare, instead of profit-maximization [29]. The measurement of optimality is not maximum cardinality or minimum cost, but

* The authors were supported by the Hungarian Academy of Sciences under its Momentum Programme (LP2016-3/2016) and its János Bolyai Research Scholarship, OTKA grant K108383 and COST Action IC1205 on Computational Social Choice, and by the Alexander von Humboldt Foundation with funds of the German Federal Ministry of Education and Research (BMBF).

the certainty that no two agents are willing to selfishly modify the market situation. Stable matchings were first formally defined in the seminal paper of Gale and Shapley [17]. They described an instance of the college admission problem and introduced the terminology based on marriage that since then became widespread. Besides resident allocation, variants of the stable matching problem are widely used in other employer allocation markets [30], university admission decisions [2,4], campus housing assignments [5,28] and bandwidth allocation [16]. A recent honor proves the currentness and importance of results in the topic: in 2012, Lloyd S. Shapley and Alvin E. Roth were awarded the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel for their outstanding results on market design and matching theory.

In the stable marriage problem, we are given a bipartite graph, where the two classes of vertices represent men and women, respectively. Each vertex has a strictly ordered preference list over his or her possible partners. A matching is *stable* if it is not *blocked* by any edge, that is, no man-woman pair exists who are mutually inclined to abandon their partners and marry each other [17].

In practice, the stable matching problem is mostly used in one of its capacitated variants, which are the stable many-to-one matching, many-to-many matching and allocation problems. The *stable flow* problem can be seen as a high-level generalization of all these settings. To the best of our knowledge, it is the most complex graph-theoretical generalization of the stable marriage model, and thus plays a crucial role in the theoretical understanding of the power and limitations of the stable marriage concept. From a practical point of view, stable flows can be used to model markets in which interactions between agents can involve chains of participants, e.g., supply chain networks involving multiple independent companies.

In the stable flow problem, a directed network with preferences models a market situation. Vertices are vendors dealing with some goods, while edges connecting them represent possible deals. Through his preference list, each vendor specifies how desirable a trade would be to him. Sources and sinks model suppliers and end-consumers. A feasible network flow is stable, if there is no set of vendors who mutually agree to modify the flow in the same manner. A blocking walk represents a set of vendors and a set of possible deals so that all of these vendors would benefit from rerouting some flow along the blocking walk.

Literature review. The notion of stability was extended to so-called "vertical networks" by Ostrovsky in 2008 [26]. Even though the author proves the existence of a stable solution and presents an extension of the Gale-Shapley algorithm, his model is restricted to unit-capacity acyclic graphs. Stable flows in the more general setting were defined by Fleiner [13], who reduced the stable flow problem to the stable allocation problem.

The best currently known computation time for finding a stable flow is $\mathcal{O}(|E| \log |V|)$ in a network with vertex set V and edge set E. This bound is due to Fleiner's reduction to the stable allocation problem and its fastest solution described by Dean and Munshi [9]. Since the reduction takes $\mathcal{O}(|V|)$ time and does not change the instance size significantly and the weighted stable allocation problem can be solved in $\mathcal{O}(|E|^2 \log |V|)$ time [9], the same holds for the maximum weight stable flow problem. The Gale-Shapley algorithm can also be extended for stable flows [8], but its straightforward implementation requires exponential running time, just like in the stable allocation problem.

It is sometimes desirable to compute stable solutions using certain forced edges or avoiding a set of forbidden edges. This setting has been an actively researched topic for decades [6,10,14,20,25]. This problem is known to be solvable in polynomial time in the one-to-one matching case, even in non-bipartite graphs [14]. Though Knuth presented a combinatorial method that finds a stable matching in a bipartite graph with a given set of forced edges or reports that none exists [25], all known methods for finding a stable matching with both forced and forbidden edges exploit a somewhat involved machinery, such as rotations [20], LP techniques [11,12,21] or reduction to other advanced problems in stability [10,14].

In many flow-based applications, various goods are exchanged. Such problems are usually modeled by multicommodity flows [22]. A maximum multicommodity flow can be computed in strongly polynomial time [31], but even when capacities are integer, all optimal solutions might be fractional, and finding a maximum integer multicommodity flow is NP-hard [19]. Király and Pap [24] introduced the concept of stable multicommodity flows, in which edges have preferences over which commodities they like to transport and the preference lists at the vertices may depend on the commodity. They show that a stable solution always exists, but it is PPAD-hard to find one.

Our contribution and structure. In this paper we discuss new and simplified algorithms and complexity results for three differently complex variants of the stable flow problem. Section 2 contains preliminaries on stable flows.

- In Section 3 we present a *polynomial algorithm for stable flows.* To derive a fast, elegant, and direct solution method, we combine the well-known pseudo-polynomial Gale-Shapley algorithm and the proposal-refusal pointer machinery known from stable allocations into an augmenting-path algorithm for computing a stable flow.
- Then, in Section 4 *stable flows with restricted intervals* are discussed. We provide a simple combinatorial algorithm to find a flow with flow value within a pre-given interval for each edge. Surprisingly, our algorithm directly translates into a very simple new algorithm for the problem of stable matchings with forced and forbidden edges in the classical stable marriage case. Unlike the previously known methods, our result relies solely on elementary graph transformations.
- Finally, in Section 5 we study *stable multicommodity flows.* First, we provide tools to simplify stable multicommodity flow instances to a great extent by showing that it is without loss of generality to assume that no commodity-specific preferences at the vertices and no commodity-specific capacities on the edges exist. Then, we reduce 3-SAT to the integral stable multicommodity flow problem and show that it is NP-complete to decide whether an integral solution exists even if the network in the input has integral capacities only.

2 Preliminaries

A *network* (D, c) consists of a directed graph $D = (V, E)$ and a capacity function $c : E \to \mathbb{R}_{\geq 0}$ on its edges. The vertex set of D has two distinct elements, also called *terminal vertices*: a source s, which has outgoing edges only and a sink t, which has incoming edges only.

Definition 1 (flow). *Function $f : E \to \mathbb{R}_{\geq 0}$ is a* flow *if it fulfills both of the following requirements:*

1. *capacity constraints: $f(uv) \leq c(uv)$ for every $uv \in E$;*
2. *flow conservation: $\sum_{uv \in E} f(uv) = \sum_{vw \in E} f(vw)$ for all $v \in V \setminus \{s, t\}$.*

A stable flow instance is a triple $\mathcal{I} = (D, c, O)$. It comprises a network (D, c) and O, the preference ordering of vertices on their incident edges. Each non-terminal vertex ranks its incoming and also its outgoing edges strictly and separately. If v prefers edge vw to vz, then we write $\mathrm{r}_v(vw) < \mathrm{r}_v(vz)$. Terminals do not rank their edges, because their preferences are irrelevant with respect to the following definition.

Definition 2 (blocking walk, stable flow). *A* blocking walk *of flow f is a directed walk $W = \langle v_1, v_2, ..., v_k \rangle$ such that all of the following properties hold:*

1. *$f(v_i v_{i+1}) < c(v_i v_{i+1})$, for each edge $v_i v_{i+1}$, $i = 1, ..., k-1$;*
2. *$v_1 = s$ or there is an edge $v_1 u$ such that $f(v_1 u) > 0$ and $\mathrm{r}_{v_1}(v_1 v_2) < \mathrm{r}_{v_1}(v_1 u)$;*
3. *$v_k = t$ or there is an edge $w v_k$ such that $f(w v_k) > 0$ and $\mathrm{r}_{v_k}(v_{k-1} v_k) < \mathrm{r}_{v_k}(w v_k)$.*

A flow is stable*, if there is no blocking walk with respect to it in the graph.*

Unsaturated walks fulfilling point 2 are said to *dominate* f at start, while walks fulfilling point 3 dominate f at the end. We can say that a walk blocks f if it dominates f at both ends.

Throughout the paper, we will assume that the digraph D does not contain loops or parallel edges, the source s only has outgoing edges, the sink t only has incoming edges, and that no isolated vertices exist. All these assumptions are without loss of generality and only for notational convenience.

Problem 1. SF
Input: $\mathcal{I} = (D, c, O)$; a directed network (D, c) and O, the preference ordering of vertices.
Question: Is there a flow f so that no walk blocks f?

Theorem 1 (Fleiner [13]). SF *always has a stable solution and it can be found in polynomial time. Moreover, for a fixed* SF *instance, each edge incident to s or t has the same value in every stable flow.*

This result is based on a reduction to the stable allocation problem. The second half of Theorem 1 can be seen as the flow generalization of the so-called *Rural Hospitals Theorem*, known for stable matching instances in general graphs. Part of this theorem states that if a vertex is unmatched in one stable matching, then all stable solutions leave it unmatched [18].

Algorithm 1: Augmenting path algorithm for stable flows

```
Initialize $\pi, \rho$.
while $\pi[s] \neq \emptyset$ do
    Let $W$ be an $s$-$t$-path or cycle in $H_{\pi,\rho}$.
    Let $\Delta := \min_{e \in W} c_f(e)$.
    Augment $f$ by $\Delta$ along $W$.
    while $\exists\, uv \in E_{H_{\pi,\rho}}$ with $c_f(uv) = 0$ do
        UpdatePointers (u)
```

3 A polynomial-time augmenting path algorithm for stable flows

Using Fleiner's construction [13], it is possible to find a stable flow efficiently by computing a stable allocation instead. Also the popular Gale-Shapley algorithm can be extended to SF. As described in [8], this yields a preflow-push type algorithm, in which vertices forward or reject excessive flow according to their preference lists. While this algorithm has the advantage of operating directly on the network without transforming it to a stable allocation instance, it requires pseudo-polynomial running time.

In the following, we describe a polynomial time algorithm to produce a stable flow that operates directly on the network D. Our method is based on the well-known augmenting path algorithm of Ford and Fulkerson [15], also used by Baïou and Balinski [1] and Dean and Munshi [9] for stability problems. The main idea is to introduce proposal and refusal pointers to keep track of possible Gale-Shapley steps and execute them in bulk. Each such iteration corresponds to augmenting flow along an s-t-path or cycle in a restricted residual network.

In the algorithm, every vertex $v \in V \setminus \{t\}$ is associated with two pointers, the *proposal pointer* $\pi[v]$ and the *refusal pointer* $\rho[v]$. Initially $\pi[v]$ points to the first-choice outgoing edge on v's preference list, whereas $\rho[v]$ is inactive. Throughout the algorithm $\pi[v]$ traverses the outgoing edges of v in order of increasing rank on v's preference list (for the source s, we assume an arbitrary preference order) until it gets advanced beyond the final outgoing edge. Then $\rho[v]$ becomes active and traverses the incoming edges of v in order of decreasing rank on v's preference list.

With any state of the pointers π, ρ, we associate a helper graph $H_{\pi,\rho}$, which contains the edges pointed at by the proposal pointers and the reversals of the edges pointed at by the refusal pointer. A recursive update procedure for advancing the pointers along their lists ensures that throughout the algorithm, $H_{\pi,\rho}$ contains an s-t-path or a cycle, with all edges having positive residual capacity c_f with respect to the current flow. The algorithm then augments the flow along this path or cycle and updates the pointers of saturated edges. Once $\pi[s]$ has traversed all outgoing edges of s, the algorithm has found a stable flow. As in each iteration, at least one pointer is advanced, the running time of the algorithm is polynomial in the size of the graph. See the full version [7] for a listing of the subroutine `UpdatePointers` and a complete analysis.

4 Stable flows with restricted intervals

Various stable matching problems have been tackled under the assumption that restricted edges are present in the graph [10,14]. A restricted edge can be *forced* or *forbidden*, and the aim is to find a stable matching that contains all forced edges, while it avoids all forbidden edges. Such edges correspond to transactions that are particularly desirable or undesirable from a social welfare perspective, but it is undesirable or impossible to push the participating agents directly to use or avoid the edges. We thus look for a stable solution in which the edge restrictions are met voluntarily.

A natural way to generalize the notion of a restricted edge to the stable flow setting is to require the flow value on any given edge to be within a certain interval. To this end, we introduce a *lower capacity* function $\mathfrak{l} : E \to \mathbb{R}_{\geq 0}$ and an *upper capacity* function $\mathfrak{u} : E \to \mathbb{R}_{\geq 0}$.

Problem 1. SF RESTRICTED
Input: $\mathcal{I} = (D, c, O, \mathfrak{l}, \mathfrak{u})$; an SF instance (D, c, O), a lower capacity function $\mathfrak{l} : E \to \mathbb{R}_{\geq 0}$ and an upper capacity function $\mathfrak{u} : E \to \mathbb{R}_{\geq 0}$.
Question: Is there a stable flow f so that $\mathfrak{l}(uv) \leq f(uv) \leq \mathfrak{u}(uv)$ for all $uv \in E$?

Note that in the above definition, the upper bound $\mathfrak{u}$ does not affect blocking walks, i.e., a blocking walk can use edge uv, even if $f(uv) = \mathfrak{u}(uv) < c(uv)$ holds.

SF RESTRICTED generalizes the natural notion of requiring flow to use an edge to its full capacity (by setting $\mathfrak{l}(uv) = c(uv)$) and of requiring flow not to use an edge at all (by setting $\mathfrak{u}(uv) = 0$), which corresponds to the traditional cases of forced and forbidden edges. In fact, it turns out that any given instance of SF RESTRICTED can be transformed to an equivalent instance in which $\mathfrak{l}(uv), \mathfrak{u}(uv) \in \{0, c(uv)\}$ for all $uv \in E$. We describe the corresponding reduction in the full version [7]. Henceforth, we will assume that our instances are of this form and use the notation $Q := \{uv \in E : \mathfrak{l}(uv) = c(uv)\}$ and $P := \{uv \in E : \mathfrak{u}(uv) = 0\}$ for the sets of forced and forbidden edges, respectively.

In the following, we describe a polynomial algorithm that finds a stable flow with restricted intervals or proves its nonexistence. We show that restricted intervals can be handled by small modifications of the network that reduce the problem to the unrestricted version of SF. We show this separately for the case that only forced edges occur, which we call SF FORCED, in Section 4.1 and for the case that only forbidden edges occur, called SF FORBIDDEN, in Section 4.2. It is straightforward to see that both results can be combined to solve the general version of SF RESTRICTED. All missing proofs can be found in the full version [7].

We mention that it is also possible to solve SF RESTRICTED by transforming the instance first into a weighted SF instance, and then into a weighted stable allocation instance, both solvable in $\mathcal{O}(|E|^2 \log |V|)$ time [9]. The advantages of

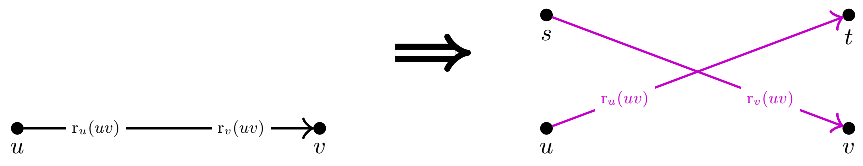

Fig. 1. Substituting forced edge uv by edges sv and ut in D'.

our method are that it can be applied directly to the SF RESTRICTED instance and it also gives us insights to solving the stable roommate problem with restricted edges directly, as pointed out at the end of Sections 4.1 and 4.2. Moreover, our running time is only $\mathcal{O}(|P||E|\log|V|)$, where P is the set of edges with $\mathfrak{u}(uv) < c(uv)$.

4.1 Forced edges

Let us first consider a single forbidden edge uv. We modify graph D to derive a graph D'. The modification consists of deleting the forced edge uv and introducing two new edges sv and ut to substitute it. Both new edges have capacity $c(uv)$ and take over uv's rank on u's and on v's preference lists, respectively, as shown in Fig. 1.

Lemma 1. *Let f be a flow in D with $f(uv) = c(uv)$. Let f' be the flow in D' derived by setting $f'(sv) = f'(ut) = f(uv)$ and $f'(e) = f(e)$ for all $e \in E \setminus \{uv\}$. Then f is stable if and only if f' is stable.*

Proof. We first observe that the set of edges not saturated by f in D is the same as the set of edges not saturated by f' in D'. This is because uv is saturated by f and ut, sv are saturated by f'. Now let $u'v'$ be such an unsaturated edge. Note that there is an edge $u'w'$ with $\mathrm{r}_{u'}(u'w) > \mathrm{r}_{u'}(u'v')$ and $f(u'w) > 0$ if and only if there is an edge $u'w'$ with $\mathrm{r}_{u'}(u'w') > \mathrm{r}_{u'}(u'v')$ and $f'(u'w') > 0$. The same is true for incoming edges at v' dominated by $u'v'$. In other words, the dominance situation at all vertices is the same for f and f'. This implies that any blocking walk for f in D is a blocking walk for f' in D' and vice versa. □

Repeated application of Lemma 1 in conjunction with the Rural Hospital Theorem (Theorem 1) allows us to transform any instance of SF FORCED into a standard stable flow instance.

Theorem 2. SF FORCED *can be solved in time $\mathcal{O}(|E|\log|V|)$.*

Stable matchings with forced edges The technique described above also provides a fairly simple method for the stable matching problem with forced edges even in non-bipartite graphs, because the Rural Hospitals Theorem holds for that case as well. After deleting each forced edge $uw \in Q$ from the graph, we add uw_s and u_tw edges to each of the pairs, where w_s and u_t are newly

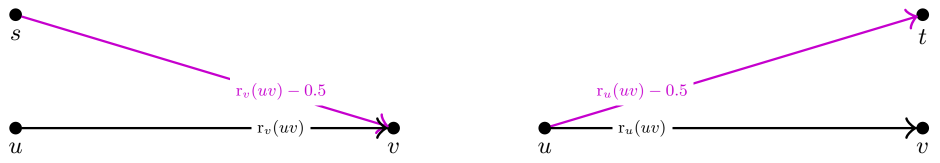

Fig. 2. Adding edges sv in D^+ and ut in D^- to forbidden edge uv.

introduced vertices. These edges take over the rank of uw. Unlike in SF, here we need to introduce two separate dummy vertices to each forced edge, simply due to the matching constraints. There is a stable matching containing all forced edges if and only if an arbitrary stable matching covers all of these new vertices w_s and u_t. The running time of this algorithm is $\mathcal{O}(|E|)$, since it is sufficient to construct a single stable solution in an instance with at most $2|V|$ vertices. More vertices cannot occur, because in a matching problem more than one forced edge incident to a vertex immediately implies infeasibility.

4.2 Forbidden edges

In order to handle SF FORBIDDEN, we present here an argumentation of the same structure as in the previous section. First, we show how to solve the problem of stable flows with a single forbidden edge by solving two instances on two different extended networks. Then we show how these constructions can be used to obtain an algorithm for the general case.

Notation For $e = uv \in P$, we define edges $e^+ = sv$ and $e^- = ut$. We set $c(e^+) = \varepsilon > 0$ and set $r_v(e^+) = r_v(e) - 0.5$, i.e., e^+ occurs on v's preference list exactly before e. Likewise, we set $c(e^-) = \varepsilon$ and $r_u(e^-) = r_u(e) - 0.5$, i.e., e^- occurs on u's preference list exactly before e. For $F \subseteq P$ we define $E^+(F) := \{e^+ : e \in F\}$ and $E^-(F) := \{e^- : e \in F\}$.

A single forbidden edge Assume that $P = \{e_0\}$ for a single edge e_0. First we present two modified instances that will come handy when solving SF FORBIDDEN. The first is the graph D^+, which we obtain from D by adding an edge e_0^+ to E. Similarly, we obtain the graph D^- by adding e_0^- to E. Both graphs are illustrated in Fig. 2.

In the following, we characterize SF FORBIDDEN instances with the help of D^+ and D^-. Our claim is that SF FORBIDDEN in D has a solution if and only if there is a stable flow f^+ in D^+ with $f^+(e^+) = 0$ or there is a stable flow f^- in D^- with $f^-(e^-) = 0$. These existence problems can be solved easily in polynomial time, since all stable flows have the same value on edges incident to terminal vertices by Theorem 1.

We will use the following straightforward observation. It follows from the fact that deletion of an edge that does not carry any flow in a stable flow neither affects flow conversation nor can create blocking walks.

Observation 1. If $f(e) = 0$ for an edge $e \in E$ and stable flow f in D, then f remains stable in $D \setminus e$ as well.

Lemma 2. *Let f be a flow in $D = (V, E)$ with $f(e_0) = 0$. Then f is a stable flow in D if and only if at least one of the following conditions hold:*

Property 1: The flow f^+ with $f^+(e) = f(e)$ for all $e \in E$ and $f^+(e_0^+) = 0$ is stable in $(V, E \cup \{e_0^+\})$.
Property 2: The flow f^- with $f^-(e) = f(e)$ for all $e \in E$ and $f^-(e_0^-) = 0$ is stable in $(V, E \cup \{e_0^-\})$.

Proof. Sufficiency of any of the two properties follows immediately from Observation 1 by deletion of e_0^+ or e_0^-, respectively.

To see necessity, assume that f is a stable flow in D. By contradiction assume that neither f^+ nor f^- is stable. Then there is a blocking walk W^+ for f^+ and a blocking walk W^- for f^-. Since W^+ is not a blocking walk for f in D, it must start with e_0^+. Since W^- is not a blocking walk for f in D, it must end with e_0^-. Let $W'^+ := W^+ \setminus \{e_0^+\}$ and $W'^- := W^- \setminus \{e_0^-\}$. Consider the concatenation $W := W'^- \circ e_0 \circ W'^+$. Note that W is an unsaturated walk in D. If $W'^- \neq \emptyset$, then W starts with the same edge as W^- and thus dominates f at the start. If $W'^- = \emptyset$, then W starts with e_0, which dominates any flow-carrying edge dominated by ut, and hence it dominates f at the start also in this case. By analogous arguments it follows that W also dominates f at the end. Hence W is a blocking walk, contradicting the stability of f. We conclude that at least one of Properties 1 or 2 must be true if f is stable. □

General case The method described above can be used to solve SF FORBIDDEN if $|P| = 1$: We simply compute stable flows f^+ in D^+ and f^- in D^-. If $f^+(e^+) = 0$ or $f^-(e^-) = 0$, we have found a stable flow in f avoiding the forbidden edge e_0. More generally, for $|P| > 1$, Lemma 2 guarantees that we can add either e^+ or e^- for each forbidden edge $e \in P$ without destroying any stable flow avoiding the forbidden edges. However, it is not straightforward to decide for which forbidden edges to add e^+ and for which to add e^-. Simply applying our method greedily for each forbidden edge does not lead to correct results, since the steps can impact each other, as shown in the full version [7]. Now we show how to resolve this issue and obtain a polynomial time algorithm for SF FORBIDDEN.

For any $A, \subseteq E$, let us denote by $D[A|B]$ the network with vertices V and edges $E \cup E^+(A) \cup E^-(B)$. Our algorithm maintains a partition of the forbidden edges in two groups P^+ and P^-. Initially $P^+ = P$ and $P^- = \emptyset$. In every iteration, we compute a stable flow f in $D[P^+|P^-]$. If $f(e^+) > 0$ for some $e \in P^+$, we move e from P^+ to P^- and repeat. If $f(e^+) = 0$ for all $e \in P^+$ but $f(e^-) > 0$ for some $e \in P^-$, we will show that no stable flow avoiding all forbidden edges exists in D. Finally, if we reach a flow f where neither of these two things happens, f is a stable flow in D avoiding all forbidden edges.

For the analysis of Algorithm 2, the following consequence of the augmenting path algorithm presented earlier (Algorithm 1) is helpful. It allows us to prove an important invariant of the Algorithm 2.

Algorithm 2: Stable flow with forbidden edges

```
Initialize $P^+ = P$ and $P^- = \emptyset$.
repeat
    Compute a stable flow $f$ in $D[P^+|P^-]$.
    if $\exists\, e \in P^+$ with $f(e^+) > 0$ then
        $P^+ := P^+ \setminus \{e\}$ and $P^- := P^- \cup \{e\}$
until $f(e^+) = 0$ for all $e \in P^+$;
if $\exists\, e \in P^-$ with $f(e^-) > 0$ then
    return $\emptyset$
else
    return $f$
```

Lemma 3. *Let f be a stable flow in D. Let f' be a stable flow in $D' = D - e'$ for some edge $e' \in \delta^+(s)$. Then $f'(e) \geq f(e)$ for all $e \in \delta^+(s) \setminus \{e'\}$.*

Lemma 4. *Algorithm 2 maintains the following invariant: There is a stable flow in D avoiding P if and only if there is a stable flow in $D[\emptyset|P^-]$ avoiding $P^+ \cup E^-(P^-)$.*

The correctness of Algorithm 2 follows immediately from the above invariant. The running time of this algorithm is bounded by $\mathcal{O}(|P||E|\log|V|)$, as each stable flow f can be computed in $\mathcal{O}(|E|\log|V|)$ time and in each round either $|P^+|$ decreases by one or the algorithm terminates. Note that both forced and forbidden edges in the same instance can be handled by our two algorithms, applying them one after the other. Finally, we can conclude the following result:

Theorem 3. SF RESTRICTED *can be solved in $\mathcal{O}(|P||E|\log|V|)$ time.*

Stable matchings with forbidden edges As before, we finish this part with the direct interpretation of our results in SR and SM instances. To each forbidden edge $uw \in P$ we introduce edges uw_s or u_tw. According to the preference lists, they are slightly better than uw itself. A stable matching with forbidden edges exists, if there is a suitable set of these uw_s and u_tw edges such that all w_s and u_t are unmatched. Our algorithm for several forbidden edges runs in $\mathcal{O}(|P||E|)$ time, because computing stable solutions in each of the $|P|$ or less rounds takes only $\mathcal{O}(|E|)$ time in SM. With this running time, it is somewhat slower than the best known method [10] that requires only $\mathcal{O}(|E|)$ time, but it is a reasonable assumption that the number of forbidden edges is small.

5 Stable multicommodity flows

A *multicommodity network* $(D, c^i, c), 1 \leq i \leq n$ consists of a directed graph $D = (V, E)$, non-negative commodity capacity functions $c^i : E \to \mathbb{R}_{\geq 0}$ for all the n commodities and a non-negative cumulative capacity function $c : E \to \mathbb{R}_{\geq 0}$ on E. For every commodity i, there is a *source* $s^i \in V$ and a *sink* $t^i \in V$, also referred to as the *terminals of commodity* i.

Definition 3 (multicommodity flow). *A set of functions* $f^i : E \to \mathbb{R}_{\geq 0}$, $1 \leq i \leq n$ *is a* multicommodity flow *if it fulfills all of the following requirements:*

1. *capacity constraints for commodities:*
   $f^i(uv) \leq c^i(uv)$ *for all* $uv \in E$ *and commodity* i*;*
2. *cumulative capacity constraints:*
   $f(uv) = \sum_{1 \leq i \leq n} f^i(uv) \leq c(uv)$ *for all* $uv \in E$*;*
3. *flow conservation:*
   $\sum_{uv \in E} f^i(uv) = \sum_{vw \in E} f^i(vw)$ *for all* $v \in V \setminus \{s^i, t^i\}$.

The concept of stability was extended to multicommodity flows by Király and Pap [24]. A stable multicommodity flow instance $\mathcal{I} = (D, c^i, c, O_E, O_V^i), 1 \leq i \leq n$ comprises a network $(D, c^i, c), 1 \leq i \leq n$, *edge preferences* O_E over commodities, and *vertex preferences* $O_V^i, 1 \leq i \leq n$ over incident edges for commodity i. Each edge uv ranks all commodities in a strict order of preference. Separately for every commodity i, each non-terminal vertex ranks its incoming and also its outgoing edges strictly with respect to commodity i. Note that these preference orderings of v can be different for different commodities and they do not depend on the edge preferences O_E over the commodities. If edge uv prefers commodity i to commodity j, then we write $\mathrm{r}_{uv}(i) < \mathrm{r}_{uv}(j)$. Analogously, if vertex v prefers edge vw to vz with respect to commodity i, then we write $\mathrm{r}_v^i(vw) < \mathrm{r}_v^i(vz)$. We denote the flow value with respect to commodity i by $f^i = \sum_{u \in V} f^i(s^i u)$.

Definition 4 (stable multicommodity flow). *A directed walk* $W = \langle v_1, v_2, ..., v_k \rangle$ blocks *flow* f with respect to commodity i *if all of the following properties hold:*

1. $f^i(v_j v_{j+1}) < c^i(v_j v_{j+1})$ *for each edge* $v_j v_{j+1}$, $j = 1, ..., k-1$*;*
2. $v_1 = s^i$ *or there is an edge* $v_1 u$ *such that* $f^i(v_1 u) > 0$ *and* $\mathrm{r}_{v_1}^i(v_1 v_2) < \mathrm{r}_{v_1}^i(v_1 u)$*;*
3. $v_k = t^i$ *or there is an edge* $w v_k$ *such that* $f^i(w v_k) > 0$ *and* $\mathrm{r}_{v_k}^i(v_{k-1} v_k) < \mathrm{r}_{v_k}^i(w v_k)$*;*
4. *if* $f(v_j v_{j+1}) = c(v_j v_{j+1})$*, then there is a commodity* i' *such that* $f^{i'}(v_j v_{j+1}) > 0$ *and* $\mathrm{r}_{v_j v_{j+1}}(i) < \mathrm{r}_{v_j v_{j+1}}(i')$.

A multicommodity flow is stable*, if there is no blocking walk with respect to any commodity.*

Note that due to point 4, this definition allows saturated edges to occur in a blocking walk with respect to commodity i, provided that these edges are inclined to trade in some of their forwarded commodities for more flow of commodity i. On the other hand, the role of edge preferences is limited: blocking walks still must start at vertices who are willing to reroute or send extra flow along the first edge of the walk according to their vertex preferences with respect to commodity i.

Problem 2. SMF
Input: $\mathcal{I} = (D, c^i, c, O_E, O^i_V)$, $1 \le i \le n$; a directed multicommodity network (D, c^i, c), $1 \le i \le n$, edge preferences over commodities O_E and vertex preferences over incident edges $O^i_V, 1 \le i \le n$.
Question: Is there a multicommodity flow f so that no walk blocks f with respect to any commodity?

Theorem 4 (Király, Pap [24]). *A stable multicommodity flow exists for any instance, but it is* PPAD*-hard to find.*

PPAD-hardness is a somewhat weaker evidence of intractability than NP-hardness [27]. Király and Pap use a polyhedral version of Sperner's lemma [23] to prove this existence result. Note that SMF is one of the very few problems in stability [3] where a stable solution exists, but no extension of the Gale-Shapley algorithm is known to solve it – not even a variant with exponential running time.

5.1 Problem simplification

The definition of SMF given above involves many distinct components and constraints. It is natural to investigate how far the model can be simplified without losing any of its generality. It turns out that the majority of the commodity-specific input data can be dropped, as shown by Theorem 5, which we prove in the full version [7]. This result not only simplifies the instance, but it also sheds light to the most important characteristic of the problem, which seems to be the preference ordering of edges over commodities.

Theorem 5. *There is a polynomial-time transformation that, given an instance $\mathcal{I}$ of* SMF*, constructs an instance $\mathcal{I}'$ of* SMF *with the following properties:*

1. *all commodities have the same source and sink,*
2. *at each vertex, the preference lists are identical for all commodities,*
3. *there are no commodity-specific edge capacities,*

and there is a polynomially computable bijection between the stable multicommodity flows of $\mathcal{I}$ and the stable multicommodity flows of $\mathcal{I}'$.

5.2 Integral multicommodity stable flows

Finally, we discuss the problem of integer stable multicommodity flows, introduced in [24]. The proof of our result can be found in the full version [7].

Theorem 6. *It is* NP*-complete to decide whether there is an integer stable multicommodity flow in a given network. This holds even if all commodities share the same set of terminal vertices and all vertices have the same preferences with respect to all commodities (but edges might have different capacities with respect to different commodities).*

Acknowledgment We thank Tamás Fleiner for discussions on Lemma 1.

References

1. M. Baïou and M. Balinski. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics*, 101:1–12, 2000.
2. M. Balinski and T. Sönmez. A tale of two mechanisms: student placement. *Journal of Economic Theory*, 84:73–94, 1999.
3. P. Biró, W. Kern, D. Paulusma, and P. Wojuteczky. The stable fixtures problem with payments. *Games and Economic Behavior*, 2017.
4. S. Braun, N. Dwenger, and D. Kübler. Telling the truth may not pay off: an empirical study of centralized university admissions in Germany. *The B.E. Journal of Economic Analysis and Policy*, 10, article 22, 2010.
5. Y. Chen and T. Sönmez. Improving efficiency of on-campus housing: an experimental study. *American Economic Review*, 92:1669–1686, 2002.
6. Á. Cseh and D. F. Manlove. Stable marriage and roommates problems with restricted edges: Complexity and approximability. *Discrete Optimization*, 20:62 – 89, 2016.
7. Á. Cseh and J. Matuschke. New and simple algorithms for stable flow problems. *CoRR*, abs/1309.3701, 2017.
8. Á. Cseh, J. Matuschke, and M. Skutella. Stable flows over time. *Algorithms*, 6:532–545, 2013.
9. B. C. Dean and S. Munshi. Faster algorithms for stable allocation problems. *Algorithmica*, 58:59–81, 2010.
10. V. M. F. Dias, G. D. da Fonseca, C. M. H. de Figueiredo, and J. L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306:391–405, 2003.
11. T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.
12. T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11:291–319, 1994.
13. T. Fleiner. On stable matchings and flows. *Algorithms*, 7:1–14, 2014.
14. T. Fleiner, R. W. Irving, and D. F. Manlove. Efficient algorithms for generalised stable marriage and roommates problems. *Theoretical Computer Science*, 381:162–176, 2007.
15. L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.
16. A.-T. Gai, D. Lebedev, F. Mathieu, F. de Montgolfier, J. Reynier, and L. Viennot. Acyclic preference systems in P2P networks. In A. Kermarrec, L. Bougé, and T. Priol, editors, *Proceedings of Euro-Par '07 (European Conference on Parallel and Distributed Computing): the 13th International Euro-Par Conference*, volume 4641 of *Lecture Notes in Computer Science*, pages 825–834. Springer, 2007.
17. D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
18. D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.
19. M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, CA., 1979.
20. D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms.* MIT Press, 1989.
21. R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34:532–543, 1987.

22. W. S. Jewell. *Multi-commodity Network Solutions.* Operations Research Center, University of California, 1966.
23. T. Király and J. Pap. A note on kernels and Sperner's Lemma. *Discrete Applied Mathematics*, 157:3327–3331, 2009.
24. T. Király and J. Pap. Stable multicommodity flows. *Algorithms*, 6:161–168, 2013.
25. D. Knuth. *Mariages Stables.* Les Presses de L'Université de Montréal, 1976. English translation in *Stable Marriage and its Relation to Other Combinatorial Problems*, volume 10 of CRM Proceedings and Lecture Notes, American Mathematical Society, 1997.
26. M. Ostrovsky. Stability in supply chain networks. *American Economic Review*, 98:897–923, 2008.
27. C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48:498–532, 1994.
28. N. Perach, J. Polak, and U. G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion. *International Journal of Game Theory*, 36:519–535, 2008.
29. A. E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92:991–1016, 1984.
30. A. E. Roth and M. A. O. Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*, volume 18 of *Econometric Society Monographs*. Cambridge University Press, 1990.
31. É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, pages 250–256, 1986.