Modifying Preferences and Capacities for Stability in Flow Networks: Algorithms and Complexity

Gergely Csáji
ELTE KRTK, and Department of Computer Science and Information Theory, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics
Budapest, Hungary
csaji.gergely@krtk.elte.hu

Kitti Varga
Department of Computer Science and Information Theory, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics
Budapest, Hungary
vkitti@cs.bme.hu

ABSTRACT

We investigate inverse-type stable flow problems in directed networks, with possible applications to financial clearing systems and transportation networks. Here, we are given a directed network, where agents are represented by vertices, while contracts, transportation routes, debt obligations, or other type of asymmetric connections between them are represented by arcs. Furthermore, the agents in the network have some form of preferences over the connections they are involved in.

Specifically, we consider the problem where the objective is to make a given flow stable by making a minimum extent of modifications to the system parameters. In this paper, we explore two settings where modifications are allowed: (i) only in the agents' preferences, and (ii) in both the agents' preferences and the capacities of the network arcs. In the second case, we also pose a restriction that preferences can only be changed on the arcs with a positive value of flow, and these preferences can only be increased. For each setting, we present efficient algorithms that minimize the ℓ_1-norm of the modifications. We also consider a model with strict, ordinal preferences, where instead of minimizing the ℓ_1-norm of the modification in the preferences, we minimize the number of swaps in the preference lists, and show that both settings are NP-hard. Our algorithms rely on methods from combinatorial optimization, such as submodular optimization and minimum-cut algorithms.

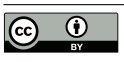
CCS CONCEPTS

• **Theory of computation** → **Network flows**; • **Mathematics of computing** → *Combinatorial optimization.*

KEYWORDS

Inverse optimization; Stable matchings; Network flows; Capacity modification

ACM Reference Format:
Gergely Csáji and Kitti Varga. 2026. Modifying Preferences and Capacities for Stability in Flow Networks: Algorithms and Complexity. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/



Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/

1 INTRODUCTION

Stable matching theory, originating from the seminal paper of Gale and Shapley [10], has long served as a powerful framework for problems like resident allocation and school choice [6, 14, 17]. However, for complex systems, bipartite settings with fixed preferences might be insufficient. This has led to the development of more general frameworks, such as stable flows and circulations, first introduced by Fleiner [9]. Stable flows generalize stable matchings to networks that involve complex chains of transactions between agents. These networks can take various forms, such as supply chain networks [16] or transportation systems [11], where agents represent companies and their locations, arcs correspond to possible transportation routes, and capacities reflect the limits of roads or vehicles. Another example is financial networks [3], where agents are banks, and the arcs represent debt obligations, with capacities indicating the amounts owed. In all cases, the notion of stability ensures that the flow of resources — whether goods or payments — remains efficient and balanced, preventing participants from seeking alternative routes or arrangements that would destabilize the system.

The challenge arises when a central authority — such as a government, regulatory body, or system operator — aims to ensure the stability of the overall system according to certain predefined criteria. In many cases, the preferences of agents (e.g., banks, transport companies, or individuals) are not aligned with the central authority's desired solution. For instance, in a financial network, a regulator may prefer a solution that minimizes systemic risk, while individual banks may prioritize their own liquidity or exposure. Similarly, in transportation networks, the central authority may aim to minimize overall congestion or pollution, which may conflict with individual companies' routing preferences.

In this work, we make the assumption that the central authority can modify the preferences of the agents, and also the capacities of the network arcs by providing compensation proportional to the required changes. For instance, in a financial clearing network, compensation might take the form of monetary incentives to adjust debt repayments, whereas in a transportation network, companies could be compensated for taking less efficient routes through reduced tolls or subsidies. Modifying the capacities of the arcs can mean forgiving debts between banks or improving the already existing infrastructure. In this paper, we present efficient algorithms that enable the central authority to determine some modification of minimum extent to capacities and preferences needed to make a given preferred flow (or circulation) stable. By compensating agents for changes in preferences and capacities, our approach offers a

practical means of aligning individual interests with the central authority's goals, thus ensuring a stable and efficient system.

However, in our second model, where both the agents' preferences and the capacities of the network arcs can be modified, we introduce an additional constraint: the preferences can only be increased on arcs included in the proposed solution. In other words, the central authority can offer compensation or incentives to agents for using specific arcs, but it cannot reduce their preferences for alternative options. This assumption reflects real-world constraints, where providing additional compensation (e.g., financial incentives for using certain transportation routes or prioritizing certain debt repayments) can increase the attractiveness of certain options, but cannot diminish the utility of other potential choices. For instance, in a transportation network, compensation might be offered to encourage the use of less congested routes, but this cannot make more efficient or faster routes less appealing.

1.1 Related Work

Stable matchings were introduced by Gale and Shapley [10] in 1962. They showed that in bipartite graphs, a stable matching always exists and can be found in linear time. Irving [13] proved that if the preferences of the agents are strict, then it is possible to decide if a stable matching in any graph exists in linear time. This result was further strengthened by Tan [18], who showed that a stable matching or a certificate of the nonexistence of stable matchings in any graph can be found in polynomial time.

Baïou and Balinski [1] extended stability from bipartite graphs to the case where both the vertices and the edges have capacities, resulting in what is known as the stable allocation problem. Here, the traditional Gale–Shapley algorithm [10] may have exponential running time, but Baïou and Balinski gave a polynomial-time algorithm [1] to find a stable allocation in such a bipartite instance. The concept of stable allocations and this algorithm was extended to arbitrary graphs by Biró and Fleiner [4]. Later, stability has been extended to different structures, such as hypergraphs and flows. Fleiner [9] proved that stable flows always exist by reducing the stable flow problem to the stable allocation problem.

Boehmer et al. [5] gave a comprehensive study on the manipulability of stable matchings, where manipulation can come from two sources: either from the agents or the central authority. They studied several forms of manipulation (e.g., adding or deleting agents or contracts, and swapping agents in preference lists) and how these actions can be optimally used to ensure that some desirable conditions are satisfied, such as matching two specific agents together.

Bérczi, Csáji, and Király [2] studied similar questions: they considered the problem of determining a modification to the agents' preferences that makes a given matching stable while minimizing the ℓ_1-norm of the modifications. They showed that this problem can be solved in polynomial time for bipartite graphs, but it becomes NP-hard for arbitrary graphs. Eiben et al. [8] studied the case when instead of minimizing the ℓ_1-norm, the goal is to minimize the total number of swaps (i.e. a change of the order of two adjacent entries in a preference list). They showed that this problem can also be solved efficiently for bipartite graphs.

1.2 Our Contributions

In this paper, we consider inverse optimization questions regarding stable flows. Given a network (D, s, t, c) — where D is a directed graph without loops, s, t are designated source and terminal vertices, and $c\colon A(D) \to \mathbb{R}_+$ is a capacity function over the arcs of D — in which the vertices represent agents, and given utility functions p_v for each $v \in V(D)$ representing agents' weak preference lists (i.e., there can be ties in the order) over the arcs incident to v, our objective is to make a given feasible flow (or circulation) x stable by applying minimum extent of modification to the system. Stability in such networks ensures that no agent or participant has an incentive to deviate from the prescribed flow (or circulation), thereby preserving the overall efficiency of the system.

We explore two models based on the types of modifications allowed to achieve stability. In the first model, we assume that only the agents' preferences can be modified, while in the second model, both the agents' preferences and the capacities of the arcs may be altered. We use the ℓ_1-norm to measure the extent of modifications in the capacities. To measure the extent of modification in the agents' preferences, we consider two distinct measures.

In one scenario, the preferences are derived from utility values, and the cost of modifying an agent's preferences is directly proportional to the magnitude of the changes in these utility values. Our goal is to make a given flow stable while minimizing the ℓ_1-norm of the total modifications to all utility functions. In the other scenario, we assume that each agent has strict preferences, and the cost of modifying an agent's preferences is the number of swaps (i.e., the changes of two consecutive agents) in its preference list.

In Section 3, we show that when only preferences can be modified, this problem can be solved in polynomial time under the ℓ_1-norm (see Algorithm 1), but it becomes NP-hard under the swap distance (see Theorem 3.6). In Section 4, both preferences and capacities are allowed to be modified, but we introduce an additional restriction: only the utility values corresponding to nonempty arcs can be modified, and these values can only be increased. Under this restriction, we prove that the problem can again be solved in polynomial time under the ℓ_1-norm (see Algorithm 2) but it is NP-hard if preference modifications are measured by swaps (see Remark 1).

2 PRELIMINARIES

We denote the set of nonnegative real numbers by $\mathbb{R}_+$ and the set of nonnegative integers by $\mathbb{Z}_+$. For a positive integer k, let $[k] := \{1, \dots, k\}$. For an undirected graph $G = (V, E)$ and for a vertex $v \in V$, we denote by $E(v)$ the set of edges incident to v in G. For a directed graph $D = (V, A)$ and for a vertex $v \in V$, we denote by $\rho(v)$ the set of arcs incoming to v and by $\delta(v)$ the set of arcs outgoing from v, and we let $E(v) := \rho(v) \cup \delta(v)$. A vertex cover in an undirected graph is a subset of vertices that intersects each edge.

Let S be a finite ground set. We denote by $\mathcal{P}(S)$ the family of all possible subsets of S. For a set $X \subseteq S$ and for an element $y \in S$, the notation $X \cup \{y\}$ is abbreviated by $X + y$, and $X \setminus \{y\}$ is abbreviated by $X - y$. We say that a set function $h\colon \mathcal{P}(S) \to \mathbb{R}_+ \cup \{\infty\}$ is **monotone increasing** if $h(X) \le h(Y)$ holds for any $X \subseteq Y \subseteq S$. The set function h is said to be **submodular** if $h(X) + h(Y) \ge h(X \cap Y) + h(X \cup Y)$ holds for all $X, Y \subseteq S$. Given a partition $S = U \cup W$, the set function h is called **bipartite-submodular** if $h(X) = h_U(X \cap U) + h_W(X \cap W)$ holds for some submodular functions $h_U\colon \mathcal{P}(U) \to \mathbb{R}_+ \cup \{\infty\}$ and $h_W\colon \mathcal{P}(W) \to \mathbb{R}_+ \cup \{\infty\}$.

We utilize the following theorem of Hochbaum [12].

Theorem 2.1 (Hochbaum [12]). *Given a bipartite graph $G = (U, W; E)$ and a polynomial-time computable, bipartite-submodular cost function $c\colon \mathcal{P}(U \cup W) \to \mathbb{R}_+ \cup \{\infty\}$, a minimum-cost vertex cover can be found in polynomial time.*

2.1 Stable Flows

In the stable matching problem, we are given a bipartite graph $G = (U, W; E)$ such that each vertex has a weak preference list $\succcurlyeq_v$ over its incident edges (i.e., ties are allowed). Given a matching $M \subseteq E$, we say that an edge $e = uw \notin M$ **blocks** M if for both u and w it holds that either it strictly prefers e to the edge it is incident to in M or it is unmatched. That is, we assume that for any vertex v and for any edge $e \in E(v)$, we have $e \succ_v \emptyset$. We say that a matching M is **stable** if there is no blocking edge to M.

A nice generalization of the stable matching problem to directed networks, introduced by Fleiner [9], is called the stable flow problem. In the stable flow problem, we are given a directed graph $D = (V, A)$ with a designated source vertex s with no incoming arcs and a terminal vertex t with no outgoing arcs. Furthermore, for each arc $e \in A$ we have a nonnegative capacity $c(e)$, and for each vertex $v \in V$ we have weak preference lists $\succcurlyeq_v^{\text{in}}$ and $\succcurlyeq_v^{\text{out}}$ over the incoming arcs $\rho(v)$ and the outgoing arcs $\delta(v)$ of v, respectively. Here, we assume these preferences come from utility functions $p_v\colon E(v) \to \mathbb{R}_+$, such that $e \succ_v f$ if and only if $p_v(e) > p_v(f)$, and $e \succcurlyeq_v f$ if and only $p_v(e) \geq p_v(f)$ for any arcs $e, f \in E(v)$.

A function $x\colon A \to \mathbb{R}_+$ is called a **flow** if Kirchoff's current law, that is, $\sum_{e\in\rho(v)} x(e) = \sum_{e\in\delta(v)} x(e)$ holds for all $v \in V \setminus \{s, t\}$. We say that a flow x is **feasible** in the network (D, s, t, c) if $0 \leq x(e) \leq c(e)$ holds for each $e \in A$. Given a feasible flow x, let $\rho_x(v) \coloneqq \{e \in \rho(v) \mid x(e) > 0\}$ and $\delta_x(v) \coloneqq \{e \in \delta(v) \mid x(e) > 0\}$ for any vertex $v \in V$.

Here, the blocking structures for a flow x may not only be edges, but arbitrary directed walks such that both the first and the last agents of the walk can mutually benefit from redirecting some flow along the walk (or just sending or accepting more in the cases of s and t, respectively).

Definition 2.2. Let x be a feasible flow. We say that a directed walk $P = (v_1, e_1, \ldots, e_{k-1}, v_k)$ **blocks** x if the following conditions hold:

(i) $x(e_i) < c(e_i)$ for $i \in [k-1]$,
(ii) either $v_1 = s$ or there exists an arc v_1u such that $x(v_1u) > 0$ and $e_1 \succ_{v_1}^{\text{out}} v_1u$, and
(iii) either $v_k = t$ or there exists an arc uv_k such that $x(uv_k) > 0$ and $e_{k-1} \succ_{v_k}^{\text{in}} uv_k$.

We say that x is **stable** if no walk P blocks x. We remark that this notion is often called **weak stability** in the literature.

If condition (ii) is not satisfied for a walk P, then we say that P is **dominated at** v_1, and if condition (iii) is not satisfied, then we say it is dominated at v_k.

Similarly, we say that an arc $e \in \delta(v)$ is dominated at $v \neq s$ by x if $x(e) = c(e)$ or for each $f \in \delta_x(v)$, we have $f \succcurlyeq_v^{\text{out}} e$. We say that an arc $e \in \delta(s)$ is dominated by x at s if $x(e) = c(e)$. A subset $F \subseteq \delta(v)$ is **dominated by** x at v, if each $e \in F$ is dominated at v. These concepts are defined analogously for incoming arcs. Clearly, if an arc is dominated at its tail or at its head, then it cannot be the first or the last arc of a blocking walk, respectively.

We can also define a stable circulation model for a network (D, c). Here, the definition of a blocking walk and stability is almost the same, with the exception that there are no specified source and terminal vertices s and t in the network. However, such a model trivially reduces to the stable flow model by adding two isolated vertices s and t as the source and the terminal. Hence, we only consider the stable flow model in this paper.

2.2 Measuring the Distance Between Two Preference Lists

We consider two different notions to measure the distance between two preference lists.

If two preference lists $\succcurlyeq$ and $\succcurlyeq'$ over the same set A are induced by some utility functions $p\colon A \to \mathbb{R}_+$ and $p'\colon A \to \mathbb{R}_+$, respectively, then a natural way to compare these two preference lists is to consider the **ℓ_1-norm** of the difference $p - p'$, that is, $\|p - p'\|_1 \coloneqq \sum_{e\in A} |p(e) - p'(e)|$.

Another well-known metric to compare two strict preference lists $\succ$ and $\rhd$ in the literature is the swap distance [7, 8]. A **swap** in a preference list is defined as a change between two adjacent entries. For example, swapping b and c in the preference list $a \succ b \succ c$ yields the preference list $a \rhd c \rhd b$. Given two arbitrary strict preference lists $\succ$ and $\rhd$ over the same set A, the **swap distance** $d_{\text{swap}}(\succ, \rhd)$ of $\succ$ and $\rhd$ is defined to be the minimum number of swaps required to transform $\succ$ into $\rhd$. For example, the swap distance of $a \succ b \succ c$ and $a \rhd c \rhd b$ is 1, while the swap distance of $a \succ b \succ c$ and $c \rhd b \rhd a$ is 3.

3 MODIFYING THE PREFERENCES

In this section, we investigate inverse optimization for our first model, where only the preferences can be modified. Here, we assume that the preferences are induced by utility functions and our objective is to minimize the ℓ_1-norm of the total changes in the utility values to make a given feasible flow stable. With Algorithm 1, we provide a polynomial-time algorithm for this problem.

Formally, we consider the following problem. We are given a feasible flow x in a network (D, s, t, c) with utility functions $p_v : E(v) \to \mathbb{R}_+$ for each $v \in V$, inducing weak preference lists $\succcurlyeq_v^{\text{in}}$ and $\succcurlyeq_v^{\text{out}}$ as described in Section 2.1. For any $v \in V$, let $\alpha_v, \beta_v\colon E(v) \to \mathbb{R}_+ \cup \{\pm\infty\}$ be lower and upper bounds such that $\alpha_v \leq \beta_v$. Our goal is to find a new utility function p_v^* for each $v \in V$ satisfying the following conditions:

- x becomes a stable flow with respect to the preference lists defined by $\{p_v^*\}_{v\in V}$,
- $\alpha_v(e) \leq p_v^*(e) \leq \beta_v(e)$ holds for each $v \in V$ and $e \in E(v)$,
- $\sum_{v\in V} \|p_v - p_v^*\|_1$ is minimized.

3.1 Characterizing Feasibility

Let us start with a simple observation regarding feasibility. Let x be a fixed feasible flow. We say that an arc e is **unsaturated** by x if $x(e) < c(e)$. We say that an s-t path is unsaturated by x if all of its arcs are unsaturated.

Observation 1. *Let x be a feasible flow in the network (D, s, t, c). If there exists an unsaturated directed s-t path, then x is not stable with respect to any possible preference lists.*

Indeed, by Definition 2.2, such an unsaturated directed s-t path always forms a blocking walk to x. Since it is easy to check in polynomial time if there is an unsaturated s-t path (by finding a path between s and t in the subgraph of the unsaturated arcs), from now on we assume that no such path exists.

In addition, assuming that $\max\{\alpha_v(e) \mid e \in E(v)\} \le \min\{\beta_v(e) \mid e \in E(v)\}$ holds for each $v \in V$, the reverse direction of Observation 1 is also true.

Proposition 3.1. *Let x be a feasible flow in the network (D, s, t, c), and assume that $\max\{\alpha_v(e) \mid e \in E(v)\} \le \min\{\beta_v(e) \mid e \in E(v)\}$ holds for each $v \in V$. Then, x can be made stable by modifying the utility values in the preferences if and only if there is no unsaturated s-t path.*

Proof. The forward direction follows from Observation 1. To see the reverse direction, choose a number k_v such that $\max\{\alpha_v(e) \mid e \in E(v)\} \le k_v \le \min\{\beta_v(e) \mid e \in E(v)\}$ for each $v \in V$. Note that if there is no unsaturated s-t path, then by setting $p_v^*(e) = k_v$ for each $v \in V$ and $e \in E(v)$, every vertex ranks all of its incident arcs equally, thus x becomes stable. □

Thus, Proposition 3.1 implies that if we consider the unbounded case (that is, $\alpha_v \equiv -\infty$ and $\beta_v \equiv \infty$ for each $v \in V$), then our problem has a feasible solution if and only if there are no unsaturated directed s-t paths. Although we do not give a compact characterization of feasibility for the general case, our optimization algorithm (cf. Algorithm 1) in the next section determines feasibility in polynomial time by finding an optimal solution if one exists.

3.2 Preparations

Let us introduce some further notation. For a vertex $v \in V$, let $\delta^{\mathrm{B}}(v)$ and $\rho^{\mathrm{B}}(v)$ denote the sets of unsaturated outgoing and incoming arcs, respectively. By definition, every arc of a blocking walk is unsaturated, in particular, the first and the last ones. For a vertex $v \in V \setminus \{s, t\}$, whether an unsaturated arc can actually appear as the first or last arc of a blocking walk also depends on the preferences of v. In contrast, for s and t, their preferences do not affect whether an unsaturated arc can be the first or last arc of a blocking walk, respectively. Therefore, the preference lists of s and t can be disregarded.

To make the input feasible flow x stable, we need to ensure that for each blocking walk, the first arc gets dominated at its tail or the last arc at its head. Therefore, for each vertex, we aim to identify a subset of incident arcs that must be dominated at that vertex by modifying the utility values, while minimizing the total extent of modifications. In the following, we show that the function measuring the ℓ_1-norm of modifying the utility function at a vertex to ensure that no arc of a selected subset F of outgoing arcs can be the first arc of a blocking walk is monotone increasing and submodular in F, and is also polynomial-time computable.

For any $v \in V \setminus \{s, t\}$ and for any $F \subseteq \delta^{\mathrm{B}}(v)$, let

$$g_v^{\mathrm{out}}(F) := \min\left\{ \sum_{e \in \delta(v)} \left|p_v(e) - p_v^*(e)\right| \;\middle|\; \begin{array}{l} F \text{ is dominated by } x \\ \text{at } v \text{ with respect to} \\ \text{the utility function } p_v^*, \\ \text{and } \alpha_v \le p_v^* \le \beta_v \end{array} \right\}.$$

That is, $g_v^{\mathrm{out}}(F)$ represents the minimum possible ℓ_1-norm of the changes in the utility function p_v needed to make each arc in F dominated at v, while satisfying the lower and upper bounds. We analogously define g_v^{in}. In case there is no feasible solution, we define this minimum to be infinity, i.e., we use the convention $\min \emptyset = \infty$.

Next, we show the submodularity of the functions g_v^{out} and g_v^{in}, a property that is heavily utilized in our algorithm. Let us define a linear programming problem for each vertex $v \in V \setminus \{s, t\}$ and for each subset $F \subseteq \delta^{\mathrm{B}}(v)$.

$$\begin{aligned} \min \quad & \sum_{e \in \delta(v)} \left|p_v(e) - z(e)\right| \\ \text{s.t.} \quad & z(f) \le z(e) && \text{for all } f \in F,\ e \in \delta_x(v) \\ & \alpha_v(e) \le z(e) \le \beta_v(e) && \text{for all } e \in \delta(v) \end{aligned} \tag{$\mathrm{LP}_v(F)$}$$

Though the above objective function is not linear, it can be easily linearized by introducing variables $y(e)$ with constraints $y(e) \ge p_v(e) - z(e)$ and $y(e) \ge -p_v(e) + z(e)$ and minimizing $\sum_{e \in \delta(v)} y(e)$ instead. It is not difficult to see that the optimum value of $\mathrm{LP}_v(F)$ represents the minimum cost of modifying the utility function of v so that F becomes dominated by x at v, while the lower and upper bounds are satisfied. We denote the optimum value of $\mathrm{LP}_v(F)$ by $\mathrm{OPT}_v(F)$. Then, we have $g_v^{\mathrm{out}}(F) = \mathrm{OPT}_v(F)$.

The proof of the following technical claim is omitted due to page limits.

Proposition 3.2. *For each $v \in V \setminus \{s, t\}$, the functions g_v^{out} and g_v^{in} are monotone increasing, submodular and polynomial-time computable. Furthermore, if the utility function p_v, the lower bounds α_v, and the upper bounds β_v if p_v, α_v, and β_v take values from $\mathbb{Z} \cup \{\pm\infty\}$, then so do g_v^{out} and g_v^{in}, and the corresponding linear programs have integral optimal solutions if they are feasible.*

In our algorithm, we only use that the functions g_v^{out} and g_v^{in} are monotone increasing, submodular and polynomial-time computable for all $v \in V \setminus \{s, t\}$. Hence, for the sake of generality, we consider an extended framework in which the cost of dominating any subset of $\delta^{\mathrm{B}}(v)$ at a vertex $v \in V$ is given by an arbitrary monotone increasing, submodular and polynomial-time computable function h_v^{out}, and similarly, the cost of dominating the subsets of $\rho^{\mathrm{B}}(v)$ at v is given by a monotone increasing, submodular and polynomial-time computable function h_v^{in}. We allow h_v^{out} and h_v^{in} to have value ∞, which corresponds to the fact that dominating every arc of F at v in a feasible way is impossible. For example, the nonempty subsets of unsaturated arcs cannot be dominated at s or t. For simplicity, we introduce the functions h_u^{out} and h_u^{in} for any $u \in \{s, t\}$, where $h_u^{\mathrm{out}}(\emptyset) = h_u^{\mathrm{in}}(\emptyset) = 0$ and $h_u^{\mathrm{out}}(F) = h_u^{\mathrm{in}}(F) = \infty$ otherwise.

Inv-StabFlow-PrefMod

In: A network (D, s, t, c), a feasible flow x, and monotone increasing, submodular, polynomial-time computable functions $h_v^{\mathrm{out}} \colon \delta^{\mathrm{B}}(v) \to \mathbb{R}_+ \cup \{\infty\}$ and $h_v^{\mathrm{in}} \colon \rho^{\mathrm{B}}(v) \to \mathbb{R}_+ \cup \{\infty\}$ for each $v \in V(D)$, where for any $u \in \{s, t\}$, we have $h_u^{\mathrm{out}}(\emptyset) = h_u^{\mathrm{in}}(\emptyset) = 0$ and $h_u^{\mathrm{out}}(F) = h_u^{\mathrm{in}}(F) = \infty$ otherwise.

Out: Subsets of arcs $F_v^{\mathrm{out}} \subseteq \rho^{\mathrm{B}}(v)$ and $F_v^{\mathrm{in}} \subseteq \delta^{\mathrm{B}}(v)$ for each $v \in V(D)$ such that dominating $F_v^{\mathrm{out}} \cup F_v^{\mathrm{in}}$ at v for each $v \in V$ makes x stable and $\sum_{v \in V} \left(h_v^{\mathrm{out}}(F_v^{\mathrm{out}}) + h_v^{\mathrm{in}}(F_v^{\mathrm{in}})\right)$ is minimum.

3.3 Algorithm 1

Fleiner proposed an algorithm to find a stable flow by reducing the problem to finding a stable allocation [9]. Unfortunately, this reduction seems to be inadequate for our inverse optimization problem, as a stable flow does not necessarily correspond to a unique stable allocation, and the allocations corresponding to the input flow (which we aim to make stable) can vary if the preferences change. Therefore, we now develop a different approach.

In the first step of our algorithm, we introduce an auxiliary bipartite graph whose edges capture the unsaturated walks in the original network. In the second step, we transform this bipartite graph so that its minimum-cost vertex covers with respect to some bipartite-submodular and polynomial-time computable cost function correspond to the optimal solutions of the original problem. In the third step, we construct an optimal solution of the original problem based on the minimum-cost vertex cover found in the second step.

Step 1. Let us create a bipartite graph $G = (V^{\mathrm{out}}, V^{\mathrm{in}}; E)$, where $V^{\mathrm{out}} := \{v^{\mathrm{out}} \mid v \in V \setminus \{t\}\}$ and $V^{\mathrm{in}} := \{v^{\mathrm{in}} \mid v \in V \setminus \{s\}\}$. For each pair of not necessarily distinct vertices $u \in V \setminus \{t\}$ and $v \in V \setminus \{s\}$ and each pair of arcs $uu', v'v \in A$, if there is a walk P of unsaturated arcs with respect to x starting with uu' and ending with $v'v$, we add an edge $u^{\mathrm{out}}v^{\mathrm{in}}$ with label $(uu', v'v)$ to G. Note that we may create parallel edges but only if these parallel edges have distinct labels.

Then in the graph G, every edge corresponds to some unsaturated walk P with respect to x in D, and for each such walk $P = (v_1, e_1, \dots, e_{k-1}, v_k)$ in D there is an edge $v_1^{\mathrm{out}}v_k^{\mathrm{in}}$ with label (e_1, e_{k-1}). Note that different unsaturated walks in D may correspond to the same edge of G. Since we can easily decide in polynomial time whether there exists an unsaturated walk in D with some fixed first and last arcs, G can be built in polynomial time.

Next, for all functions h_v^{in} and h_v^{out}, we define new functions $h_{v^{\mathrm{in}}}$ and $h_{v^{\mathrm{out}}}$ corresponding to the vertices $v^{\mathrm{in}}, v^{\mathrm{out}} \in V^{\mathrm{in}} \cup V^{\mathrm{out}}$ as follows. For $v = s$, we let $h_{s^{\mathrm{out}}}(\emptyset) = 0$ and $h_{s^{\mathrm{out}}}(F) = \infty$ for any $\emptyset \neq F \subseteq E(s^{\mathrm{out}})$, since dominating any arc at s in D is impossible. For $v = t$, we let $h_{t^{\mathrm{in}}}(\emptyset) = 0$ and $h_{t^{\mathrm{in}}}(F) = \infty$ for any $\emptyset \neq F \subseteq E(t^{\mathrm{in}})$. For each $v \in V \setminus \{s, t\}$, we define $h_{v^{\mathrm{out}}}(F)$ for each $F \subseteq E(v^{\mathrm{out}})$ to be $h_v^{\mathrm{out}}(F_{\mathrm{out}}^{-1})$, where $F_{\mathrm{out}}^{-1} := \{vv' \in A \mid \text{there exists } ww' \in A \text{ such that } (vv', w'w) \text{ is the label of some edge in } F\}$. Similarly, let us define $h_{v^{\mathrm{in}}}(F)$ for each $F \subseteq E(v^{\mathrm{in}})$ to be $h_v^{\mathrm{in}}(F_{\mathrm{in}}^{-1})$, where $F_{\mathrm{in}}^{-1} := \{v'v \in A \mid \text{there exists } uu' \in A \text{ such that } (uu', v'v) \text{ is the label of some edge in } F\}$.

The functions $h_{s^{\mathrm{out}}}$ and $h_{t^{\mathrm{in}}}$ are clearly submodular and polynomial-time computable. We claim that $h_{v^{\mathrm{in}}}$ and $h_{v^{\mathrm{out}}}$ are also submodular and polynomial-time computable for each $v \in V \setminus \{s, t\}$. Indeed, we have

$$\begin{aligned} h_{v^{\mathrm{out}}}(X) + h_{v^{\mathrm{out}}}(Y) &= h_v^{\mathrm{out}}\big(X_{\mathrm{out}}^{-1}\big) + h_v^{\mathrm{out}}\big(Y_{\mathrm{out}}^{-1}\big) \\ &\geq h_v^{\mathrm{out}}\big(X_{\mathrm{out}}^{-1} \cup Y_{\mathrm{out}}^{-1}\big) + h_v^{\mathrm{out}}\big(X_{\mathrm{out}}^{-1} \cap Y_{\mathrm{out}}^{-1}\big) \\ &\geq h_v^{\mathrm{out}}\big((X \cup Y)_{\mathrm{out}}^{-1}\big) + h_v^{\mathrm{out}}\big((X \cap Y)_{\mathrm{out}}^{-1}\big) \\ &= h_{v^{\mathrm{out}}}(X \cup Y) + h_{v^{\mathrm{out}}}(X \cap Y), \end{aligned}$$

using the submodularity and the monotonicity of h_v^{out} and the fact that $X_{\mathrm{out}}^{-1} \cup Y_{\mathrm{out}}^{-1} = (X \cup Y)_{\mathrm{out}}^{-1}$ and $(X \cap Y)_{\mathrm{out}}^{-1} \subseteq X_{\mathrm{out}}^{-1} \cap Y_{\mathrm{out}}^{-1}$. Also, as h_v^{out} is polynomial-time computable and F_{out}^{-1} is easy to compute for any $F \subseteq E(v^{\mathrm{out}})$, the function $h_{v^{\mathrm{out}}}$ is also polynomial-time computable. The submodularity and polynomial-time computability of $h_{v^{\mathrm{in}}}$ can be analogously proved.

Step 2. Here, we construct an edge-labeled bipartite graph $\widetilde{G} = (\widetilde{V}^{\mathrm{out}}, \widetilde{V}^{\mathrm{in}}; \widetilde{E})$ as follows. Let us create $|E(v)|$ copies $v_1, \dots, v_{|E(v)|}$ of each vertex v in G. For each edge $u^{\mathrm{out}}w^{\mathrm{in}} \in E$, let us add an edge $u_i^{\mathrm{out}}w_j^{\mathrm{in}}$ to $\widetilde{E}$ such that at the end $\widetilde{E}$ forms a perfect matching in $\widetilde{G}$. Also, let the label of $u_i^{\mathrm{out}}w_j^{\mathrm{in}} \in \widetilde{E}$ be the same as that of $u^{\mathrm{out}}w^{\mathrm{in}} \in E$. Now, there is a one-to-one correspondence between the vertices of $\widetilde{V}^{\mathrm{out}}$ in $\widetilde{G}$ (and the vertices of $\widetilde{V}^{\mathrm{in}}$ in $\widetilde{G}$) and the edges of E in G; for any $\widetilde{X} \subseteq \widetilde{V}^{\mathrm{out}}$, let $E_{\widetilde{X}} \subseteq E$ denote the set of those edges in G which correspond to a vertex in $\widetilde{X}$; and we similarly define $E_{\widetilde{Y}}$ for any $\widetilde{Y} \subseteq \widetilde{V}^{\mathrm{in}}$.

Hence we can define the functions $h_{V^{\mathrm{out}}} \colon \mathcal{P}\big(\widetilde{V}^{\mathrm{out}}\big) \to \mathbb{Z}_+ \cup \{\infty\}$ and $h_{V^{\mathrm{in}}} \colon \mathcal{P}\big(\widetilde{V}^{\mathrm{in}}\big) \to \mathbb{Z}_+ \cup \{\infty\}$ as follows. Let $h_{V^{\mathrm{out}}}(\widetilde{X}) := \sum_{v^{\mathrm{out}} \in V^{\mathrm{out}}} h_{v^{\mathrm{out}}}\big(E_{\widetilde{X}} \cap E_G(v^{\mathrm{out}})\big)$ for any $\widetilde{X} \subseteq \widetilde{V}^{\mathrm{out}}$. Here, $E_G(v^{\mathrm{out}})$ denotes the incident edges to v^{out} in $E = E(G)$. The function $h_{V^{\mathrm{in}}}$ is analogously defined. These functions are polynomial-time computable and submodular, as they are sums of polynomial-time computable submodular functions. Finally, let $h_{\widetilde{G}} \colon \mathcal{P}\big(\widetilde{V}^{\mathrm{out}} \cup \widetilde{V}^{\mathrm{in}}\big) \to \mathbb{R}$ be defined as follows. Let $h_{\widetilde{G}}\big(\widetilde{X}\big) := h_{V^{\mathrm{out}}}\big(\widetilde{X} \cap \widetilde{V}^{\mathrm{out}}\big) + h_{V^{\mathrm{in}}}\big(\widetilde{X} \cap \widetilde{V}^{\mathrm{in}}\big)$ for any $\widetilde{X} \subseteq \widetilde{V}^{\mathrm{out}} \cup \widetilde{V}^{\mathrm{in}}$.

Step 3. Let $\widetilde{C} \subseteq \widetilde{V}^{\mathrm{out}} \cup \widetilde{V}^{\mathrm{in}}$ be a minimum-cost vertex cover in $\widetilde{G}$ with respect to $h_{\widetilde{G}}$. Since $h_{V^{\mathrm{out}}}$ and $h_{V^{\mathrm{in}}}$ are two polynomial-time computable, submodular functions and $h_{\widetilde{G}}$ is a polynomial-time computable, bipartite-submodular cost function, by Theorem 2.1, we can find a minimum-cost vertex cover in $\widetilde{G}$ with respect to $h_{\widetilde{G}}$ in polynomial time using Hochbaum's algorithm [12]. If $h_{\widetilde{G}}\big(\widetilde{C}\big) = \infty$, then let Algorithm 1 output that no solution exists. Otherwise, let $\widetilde{X} := \widetilde{C} \cap \widetilde{V}^{\mathrm{out}}$ and $\widetilde{Y} := \widetilde{C} \cap \widetilde{V}^{\mathrm{in}}$, and let Algorithm 1 output the sets $F_v^{\mathrm{out}} = \big(E_{\widetilde{X}} \cap E_G(v^{\mathrm{out}})\big)^{-1}$ and $F_v^{\mathrm{in}} = \big(E_{\widetilde{Y}} \cap E_G(v^{\mathrm{in}})\big)^{-1}$ for each $v \in V(D)$.

Summary of Algorithm 1. We take an instance $I = \big(D, s, t, c, x, (h_v^{\mathrm{in}})_{v \in V}, (h_v^{\mathrm{out}})_{v \in V}\big)$ of Inv-StabFlow-PrefMod. First, we create the edge-labeled bipartite graph G described in Step 1. Then, we create the graph $\widetilde{G}$ described in Step 2. Then, we find a minimum-cost vertex cover $\widetilde{C}$ in $\widetilde{G}$ with respect to $h_{\widetilde{G}}$. If $h_{\widetilde{G}}\big(\widetilde{C}\big) = \infty$, then Algorithm 1 outputs that there is no feasible solution. Otherwise, it outputs the sets $F_v^{\mathrm{out}} = \big(E_{\widetilde{X}} \cap E_G(v^{\mathrm{out}})\big)^{-1}$ and $F_v^{\mathrm{in}} = \big(E_{\widetilde{Y}} \cap E_G(v^{\mathrm{in}})\big)^{-1}$ for each $v \in V(D)$, where $\widetilde{X} := \widetilde{C} \cap \widetilde{V}^{\mathrm{out}}$ and $\widetilde{Y} := \widetilde{C} \cap \widetilde{V}^{\mathrm{in}}$.

Due to space constraints, the correctness proof of Algorithm 1 is omitted here.

Theorem 3.3. *Algorithm 1 determines an optimal solution to* Inv-StabFlow-PrefMod *in polynomial time, if one exists.*

We remark that by Proposition 3.2, if the functions h_v^{out} and h_v^{in} are in fact the constructed functions g_v^{out} and g_v^{in} from the beginning of the section, then the integrality of p_v, α_v, and β_v implies that dominating the subsets $F_v^{\mathrm{out}} \subseteq \rho^{\mathrm{B}}(v)$ and $F_v^{\mathrm{in}} \subseteq \delta^{\mathrm{B}}(v)$ in an optimal solution p^* for each $v \in V(D)$ can be done so that p^* is also integer-valued.

3.4 Minimizing the Total Number of Swaps in the Preferences

In this section, we consider the inverse optimization problem in our first model using the swap distance. As the swap distance is defined between strict preferences only, in this section, we assume that each agent $v \in V$ has strict preference lists $\succ_v^{\text{in}}$ and $\succ_v^{\text{out}}$ instead of a utility function p_v. We could assume that these preferences are induced by strictly different utility values $p_v(e)$, however, since the swap distance does not depend on these, we refrain from using utility values in this section.

The problem is formalized as follows.

Inv-StabFlow-PrefMod-Swap

In: A network (D, s, t, c) with strict preference lists $\succ_v^{\text{in}}, \succ_v^{\text{out}}$ over the set $E(v)$ of arcs for each $v \in V(D)$ and a feasible flow x.

Out: New preference lists $\rhd_v^{\text{in}}$ and $\rhd_v^{\text{out}}$ for each $v \in V(D)$ such that x becomes a stable flow with respect to them and $\sum_{v \in V(D)} \left(d_{\text{swap}}(\succ_v^{\text{in}}, \rhd_v^{\text{in}}) + d_{\text{swap}}(\succ_v^{\text{out}}, \rhd_v^{\text{out}})\right)$ is minimum.

3.4.1 Characterizing Feasibility.

In contrast to Proposition 3.1, even though we do not have upper and lower bounds in this model, we show in Example 3.4 that a solution may not exist for Inv-StabFlow-PrefMod-Swap, even if x is a maximum flow. The reason for this is that we have to keep the preferences strict, hence only one arc with $0 < x(e) < c(e)$ can be dominated at any vertex. This crucial difference also plays a key role in our hardness proof.

Example 3.4. Let $V = \{s, t, u_1, u_2, u_3, w_1, w_2, w_3\}$ and $A = \{su_i, w_it \mid i \in [3]\} \cup \{u_iw_j \mid i, j \in [3]\}$, and let every arc have capacity 3. Let the flow x be 3 on arcs adjacent to s or t, and 1 elsewhere. Note that between the vertices $V \setminus \{s, t\}$, there are 9 arcs e with $0 < x(e) < c(e)$. Also, for any strict preference lists, at most one such arc is dominated at v, for any $v \in V \setminus \{s, t\}$. Therefore, in total, at most 6 of the 7 arcs can be dominated. So at least one arc is not dominated at any of its end-vertices; such an arc forms a blocking path by itself.

However, on the positive side, we show that we can decide whether an instance of Inv-StabFlow-PrefMod-Swap admits a feasible solution in polynomial time by reducing the problem to 2-SAT. Due to page constraints, the proof of the following theorem is omitted here.

Theorem 3.5. *We can decide in polynomial time whether an instance of* Inv-StabFlow-PrefMod-Swap *has a feasible solution.*

While by Theorem 3.5, we can decide in polynomial time whether Inv-StabFlow-PrefMod-Swap has a feasible solution, the next theorem shows that finding an optimal solution becomes NP-hard in this case.

Theorem 3.6. *The problem* Inv-StabFlow-PrefMod-Swap *is NP-hard.*

Proof. We give a reduction from the NP-hard VertexCover problem [15] to the Inv-StabFlow-PrefMod-Swap problem. Let $G = (V, E)$ be an instance of VertexCover with $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$.

We create a network (D, s, t, c) as follows. For each vertex $v_i \in V$, we add the vertices v_i and v_i' to $V(D)$. For each edge $e_k = v_iv_j$, we add the vertices u_{e_k}, $w_{e_k}^i$, and $w_{e_k}^j$ to $V(D)$. We also add a source s and a terminal t to $V(D)$. Finally, we add dummy vertices $d_1, \dots, d_r$, where $r := n^2m^2$.

We proceed to creating the arcs of D.

- For each $v_i \in V$ and for each $\ell \in [r]$, we add the arcs $sv_i, v_iv_i', v_it, v_id_\ell$ with capacity 1 to D.
- For each edge $e_k = v_iv_j$, we add the arcs $v_i'w_{e_k}^i, v_j'w_{e_k}^j$, $sw_{e_k}^i$, $sw_{e_k}^j$ with capacity 1 and the arcs $w_{e_k}^iu_{e_k}$, $w_{e_k}^ju_{e_k}$, $u_{e_k}t$ with capacity 2 to D.

The flow x is defined as follows. For any arc f of the form sv_i, v_it, $sw_{e_k}^i$, $sw_{e_k}^j$, $w_{e_k}^iu_{e_k}$, $w_{e_k}^ju_{e_k}$, let $x(f) = 1$. For any arc f of the form $u_{e_k}t$, let $x(f) = 2$. For any other arc f, let $x(f) = 0$. It is straightforward to verify that x is a feasible flow.

Finally, we define the preferences. Since there are no parallel edges in the construction, we write the preferences over the in- and outneighbors for simplicity. Let the preference list

- of each $v_i \in V(D)$ over $\delta(v_i)$ be induced by $\succ_{v_i}^{\text{out}}$ be $v_i' \succ d_1 \succ \cdots \succ d_R \succ t$,
- of each $w_{e_k}^i \in V(D)$ over $\rho(w_{e_k}^i)$ be induced by $\succ_{w_{e_k}^i}^{\text{in}}$ be $s \succ v_i'$,
- and of each u_{e_k} with $e_k = (v_i, v_j)$ and $i < j$ over $\rho(u_{e_k})$ be induced by $\succ_{u_{e_k}}^{\text{in}}$ be $w_{e_k}^i \succ w_{e_k}^j$.

The other preference lists of the construction are not given, because either there is only a single in- or out arc from the vertex, or the vertex has no incident arcs with $x(f) > 0$, implying it cannot be a start or end of a blocking walk.

The proof of correctness is omitted here. □

4 MODIFYING THE CAPACITIES AND PREFERENCES

In this section, we consider our second model, in which both the capacities and the preferences can be modified. However, we now impose an additional restriction on the preference modifications: only the utility values corresponding to nonempty arcs can be modified, and these values can only be increased; for a motivation of such a framework, see the introduction.

Formally, as before, we have a directed network (D, s, t, c) together with a feasible flow x and utility functions $p_v \colon E(v) \to \mathbb{R}_+$ for each $v \in V$. Now, our goal is to find both a new capacity function c^* and new utility functions p_v^* for each $v \in V$ such that the following hold:

(a) x becomes stable in the network (D, s, t, c^*) with respect to the preference lists defined by p_v^* for each $v \in V$,
(b) $p_v^* \geq p_v$ and $p_v^*(e) = p_v(e)$ whenever $x(e) = 0$ for each $v \in V$ and $e \in E(v)$,
(c) $\|c - c^*\|_1 + \sum_{v \in V} \|p_v - p_v^*\|_1$ is minimized.

Note that in this model we do not have any lower and upper bounds on the values of the utility functions $\{p_v^*\}_{v \in V}$. Observe that a feasible solution always exists in this case: setting $c'(e) := x(e)$ for all $e \in A$ makes the input flow x stable. Also note that in an optimal solution $c^*(e) = x(e)$ or $c^*(e) = c(e)$ holds for any arc e, since setting $c^*(e)$ to any other value would leave e unsaturated by

x, thus having no effect on the stability of x while unnecessarily increasing the ℓ_1-norm of the total modifications.

For some vertex $v \in V$ and for some subset $F \subseteq \delta(v)$ of its out-going arcs, let $g_v^{\text{out}}(F)$ denote the ℓ_1-norm of the minimum required modification in p_v in order to dominate F at v (note that being dominated and being dominated at v are different conditions), while ensuring that condition (b) holds for p_v^*. Clearly, we have $g_s^{\text{out}}(\emptyset) = 0$, and $g_s^{\text{out}}(F) = \infty$ for any $F \subseteq \delta(v)$. It is easy to see that $g_v^{\text{out}}(F)$ can be computed in polynomial time by solving a linear programming problem similar to $\text{LP}_v(F)$.

We define g_v^{in} for each $v \in V$ analogously, where we have $g_t^{\text{in}}(\emptyset) = 0$, and $g_t^{\text{in}}(F) = \infty$ for any $F \subseteq \rho(t)$.

Let $v \in V$ be arbitrary and assume that $\delta(v) = \{e_1, \dots, e_k\}$ such that $p_v(e_1) \le \dots \le p_v(e_k)$. Now we show that for any $\emptyset \neq F \subseteq \delta(v)$, we have $g_v^{\text{out}}(F) = g_v^{\text{out}}(\{e_\ell\})$, where $\ell \in [k]$ is the largest number such that $e_\ell \in F$. Indeed, as $\max\left\{p_v(e) \;\middle|\; e \in F \cap \rho^{\text{B}}(v)\right\} = p_v(e_\ell)$, the optimum values for dominating F and for dominating $\{e_\ell\}$ are equal (in the case of s, both are ∞): the preference values of all arcs with nonzero flow values have to be increased to $p_v(e_\ell)$, which is not only necessary but clearly sufficient. Hence, g_v^{out} is uniquely determined by the values $g_v^{\text{out}}(\{e_1\}) \le \dots \le g_v^{\text{out}}(\{e_k\})$. Analogous statements hold g_v^{in}.

Hence, for simplicity, from now on, we assume that for each vertex $v \in V$, we are given monotone increasing functions $h_v^{\text{out}}\colon \left\{0, 1, \dots, |\delta(v)|\right\} \to \mathbb{R}_+ \cup \{\infty\}$ and $h_v^{\text{in}}\colon \left\{0, 1, \dots, |\rho(v)|\right\} \to \mathbb{R}_+ \cup \{\infty\}$ such that $h_v^{\text{out}}(j)$ represents the cost of dominating the j least-preferred outgoing arcs of v at v for any $j \in \left\{0, 1, \dots, |\delta(v)|\right\}$, that is, $h_v^{\text{out}}(j) = g_v^{\text{out}}(\{e_j\})$, and similarly, $h_v^{\text{in}}(j)$ represents the cost of dominating the j least-preferred incoming arcs of v at v for any $j \in \left\{0, 1, \dots, |\delta(v)|\right\}$. Here, we have by the above observations on the functions g_s^{out} and g_t^{in} that $h_s^{\text{out}}(j) = \infty$ and $h_t^{\text{in}}(j) = \infty$ hold for any $j \ge 1$. It is also clear that $h_v^{\text{in}}(0) = h_v^{\text{out}}(0) = 0$ for all $v \in V$.

Hence, our new optimization problem can be defined as follows.

Inv-StabFlow-PrefCapMod

In: A network (D, s, t, c) with utility functions $p_v\colon E(v) \to \mathbb{R}_+$ for each $v \in V$, a feasible flow x, and monotone increasing functions $h_v^{\text{out}}\colon \left\{0, 1, \dots, |\delta(v)|\right\} \to \mathbb{R}_+ \cup \{\infty\}$ and $h_v^{\text{in}}\colon \left\{0, 1, 2, \dots, |\rho(v)|\right\} \to \mathbb{R}_+ \cup \{\infty\}$ for each $v \in V$ with $h_s^{\text{out}}(j) = h_t^{\text{in}}(j) = \infty$ for any $j \ge 1$, and $h_v^{\text{out}}(0) = h_v^{\text{in}}(0) = 0$ for all $v \in V$.

Out: A capacity function c^* and numbers k_v^*, ℓ_v^* for each $v \in V$ such that dominating the k_v^* least-preferred outgoing arcs and the ℓ_v^* least-preferred incoming arcs of v at v for each $v \in V$ and modifying the capacity function c to c^* makes x stable, while $\|c - c^*\|_1 + \sum_{v \in V} \left(h_v^{\text{out}}(k_v^*) + h_v^{\text{in}}(\ell_v^*)\right)$ is minimized.

4.1 Algorithm 2

The overview of Algorithm 2 is the following. Given a network (D, s, t, c) with utility functions p_{v_i} for each $v_i \in V = \{v_1, \dots, v_n\}$, a feasible flow x and monotone increasing functions $h_{v_i}^{\text{out}}$ and $h_{v_i}^{\text{in}}$ for each $v_i \in V$, we first create a new network (D', s', t', c'). Then, we find a minimum S-T cut in D' for some specific sets $S, T \subseteq V(D')$ of vertices. Finally, the output is computed based on this cut.

Intuitively, we want to 'cut' all possible blocking walks, either by making modifications in the preferences of one of the endpoints, or by decreasing the capacity of one of its arcs.

Step 1. First, we create a new directed graph $D' = (V', A')$ and a new capacity function $c'\colon A' \to \mathbb{R}_+$ as follows.

For each vertex $v_i \in V$, we create a gadget G_{v_i} as follows.

Let $e_{i1} \coloneqq v_i w_{i1}, \dots, e_{ik} \coloneqq v_i w_{ik}$ be the outgoing arcs from v_i such that $p_{v_i}(e_{i1}) \le \dots \le p_{v_i}(e_{ik})$. Similarly, let $f_{i1} \coloneqq u_{i1} v_i, \dots, f_{i\ell} \coloneqq u_{i\ell} v_i$ be the incoming arcs to v_i such that $p_{v_i}(f_{i1}) \le \dots \le p_{v_i}(f_{i\ell})$.

Let the vertices of G_{v_i} be then $v_i, v_i^{\text{out}}, v_i^{e_{i1}}, \dots, v_i^{e_{ik}}, v_i^{\text{in}}, v_i^{f_{i1}}, \dots, v_i^{f_{i\ell}}$. Let the arcs of G_{v_i} be the following:

- $v_i v_i^{\text{out}}, v_i v_i^{e_{i1}}, v_i^{\text{in}} v_i, v_i^{f_{i1}} v_i$, each with c'-capacity equal to ∞,
- $v_i^{e_{ij}} v_i^{e_{ij'}}$ for all $j, j' \in [k]$ with $j < j'$, each with c'-capacity equal to ∞,
- $v_i^{f_{ij'}} v_i^{f_{ij}}$ for all $j, j' \in [\ell]$ with $j < j'$, each with c'-capacity equal to ∞,
- $v_i^{\text{out}} v_i^{e_{ij}}$ for all $j \in [k]$, with c'-capacity $h_{v_i}^{\text{out}}(j) - h_{v_i}^{\text{out}}(j-1)$, unless $v_i = s$, in which case the c'-capacity of these arcs is ∞.
- $v_i^{f_{ij}} v_i^{\text{in}}$ for all $j \in [\ell]$, with c'-capacity $h_{v_i}^{\text{in}}(j) - h_{v_i}^{\text{in}}(j-1)$, unless $v_i = t$, in which case the c'-capacity of these arcs is ∞.

Finally, we connect these gadgets according to the arcs of D: for each arc $e = v_i v_j \in A$, we add an arc $v_i^e v_j^e$ with capacity $c'(v_i^e v_j^e) = c(e) - x(e)$.

Let $S \coloneqq \{v_i^{\text{out}} \mid v_i \in V\}$ and $T \coloneqq \{v_i^{\text{in}} \mid v_i \in V\}$.

Step 2. The algorithm proceeds by computing a minimum cut F' that separates S and T. Let $F'' \coloneqq F' \cup \{e \in A' \mid c'(e) = 0\}$. For each $v_i \in V$, the algorithm outputs the largest indices $k_{v_i}^*, \ell_{v_i}^* \ge 0$ such that $v_i^{\text{out}} v_i^{e_{ij}} \in F''$ for all $j \in [k_v^*]$ and $v_i^{f_{ij}} v_i^{\text{in}} \in F''$ for all $j \in [\ell_{v_i}^*]$. For example, if $v_i^{\text{out}} v_i^{e_{i1}}, v_i^{\text{out}} v_i^{e_{i3}} \in F''$ but $v_i^{\text{out}} v_i^{e_{i2}} \notin F''$, then $k_{v_i}^* = 1$. Finally, the algorithm also outputs the following capacity function c^*: for any arc $e = uv$, if $u^e v^e \in F''$ then $c^*(e) \coloneqq x(e)$, and $c^*(e) \coloneqq c(e)$ otherwise.

Theorem 4.1. *Algorithm 2 outputs an optimal solution to* Inv-StabFlow-PrefCapMod *in polynomial time.*

Proof. Let $I = \left(D, s, t, c, x, (h_v^{\text{out}}, h_v^{\text{in}})_{v \in V}\right)$ be an instance of Inv-StabFlow-PrefCapMod and let OPT be the objective value of an optimal solution to I. Let $I' = (D', c')$ be the network created in Step 1, and let OPT' denote the c'-capacity of a minimum S-T cut, where the sets S and T were also defined in Step 1. We show that (1) $\text{OPT}' \le \text{OPT}$ and that (2) for the output $\left(c^*, (k_v^*)_{v \in V}, (\ell_v^*)_{v \in V}\right)$ of the algorithm, we have $\|c^* - c\|_1 + \sum_{v \in V} \left(h_v^{\text{out}}(k_v^*) + h_v^{\text{in}}(\ell_v^*)\right) = c'(F') = \text{OPT}'$, where F' is a minimum S-T cut in I' with respect to c'. From these, it follows that the output of Algorithm 2 is optimal.

(1) Assume that $\left(c^*, (k_v^*)_{v \in V}, (\ell_v^*)_{v \in V}\right)$ is an optimal solution to Inv-StabFlow-PrefCapMod. Let $F \coloneqq \left\{e \in A \;\middle|\; c^*(e) = x(e)\right\}$. Now we create an arc set $F' \subseteq A(D')$ and show that it forms an S-T cut of c'-capacity equal to OPT. For each $e = uv \in F$, we add $u^e v^e$ to F'. For each $v_i \in V$, we add $v_i^{\text{out}} v_i^{e_{ij}}$ for any $j \in [k_{v_i}^*]$ and $v_i^{f_{ij}} v_i^{\text{in}}$ for any $j \in [\ell_{v_i}^*]$ to F'.

Note that each arc of the form $u^e v^e$ has capacity $c'(u^e v^e) = c(e) - x(e)$, and for each $v_i \in V$, we have $\sum_{j \in [k_{v_i}^*]} c'\left(v_i^{\text{out}} v_i^{e_{ij}}\right) =$

$h^{\text{out}}_{v_i}(k^*_{v_i})$ and $\sum_{j\in[\ell^*_{v_i}]} c'\left(v_i^{f_{ij}} v_i^{\text{in}}\right) = h^{\text{in}}_{v_i}(\ell^*_{v_i})$. Thus, it is clear that the c'-capacity of F' is exactly OPT. We only need to show that F' is an S-T cut. First, note that for each gadget G_v, we can only reach G_v from a gadget G_u for which $e = uv \in A$, and we can only reach it using the arc $u^e v^e$; and we can only continue to a gadget G_w for which $f = vw \in A$ by using the arc $v^f w^f$. Furthermore, any path through G_v must also travel through $v \in V(G_v)$, so it can visit G_v at most once. So suppose to the contrary that there exist $u^{\text{out}} \in S$ and $v^{\text{in}} \in T$ such that there exists a directed u^{out}-v^{in} path P' not containing any arcs of F'. Then by the previous observation, P' corresponds to a directed u-v path P in D. Let $v_i := v$ and let the last edge of P be $e_{ij} := wv_i$, that is, wv_i is the j-th least-preferred incoming edge for v_i. Then, we have that $\left\{v_i^{e_{ij'}} v_i^{\text{in}} \mid j' \in [j]\right\} \nsubseteq F'$, so e_{ij} is not dominated at v by the definition of F' (otherwise all of these arcs would be contained in F'). Similarly, the first edge of P is not dominated at u. Hence, P gives a blocking walk to x, which is a contradiction. Thus, it follows that $\text{OPT}' \le \text{OPT}$.

(2) Let F' be a minimum S-T cut found by Algorithm 2. As Inv-StabFlow-PrefCapMod always admits a feasible solution, by (1), that is, by $\text{OPT}' \le \text{OPT}$, we can assume that F' has finite c'-capacity. Let $F'' := F' \cup \{e \in A' \mid c'(e) = 0\}$ as said in Step 2.

For each $v \in V$, let k^*_v and ℓ^*_v be the output indices of the algorithm, that is, for each $v \in V$, we dominate the k^*_v least-preferred outgoing arcs of v and the ℓ^*_v least-preferred incoming arcs of v at v. Note that $k^*_s = \ell^*_s = k^*_t = \ell^*_t = 0$ as F'' has finite capacity. Furthermore, for each arc $e = uv$, we have $c^*(e) = x(e)$ if $u^e v^e \in F''$, and $c^*(e) = c(e)$ otherwise. Since each arc of the form $u^e v^e \in A'$ has capacity $c'(u^e v^e) = c(e) - x(e)$, and $\sum_{j\in[k^*_{v_i}]} c'\left(v_i^{\text{out}} v_i^{e_{ij}}\right) = h^{\text{out}}_{v_i}(k^*_{v_i})$ and $\sum_{j\in[\ell^*_{v_i}]} c'\left(v_i^{f_{ij}} v_i^{\text{in}}\right) = h^{\text{in}}_{v_i}(\ell^*_{v_i})$ hold for each $v_i \in V$, the objective value $\|c^* - c\|_1 + \sum_{v\in V}\left(h^{\text{out}}_v(k^*_v) + h^{\text{in}}_v(\ell^*_v)\right)$ of the solution found by Algorithm 2 is at most $c'(F'') = \text{OPT}'$.

We only have to show that x becomes stable after the modifications suggested by the output of the algorithm. Suppose to the contrary that we have a remaining blocking walk $P = (v_1, e_1, v_2, \dots, v_{m-1}, e_{m-1}, v_m)$ to x after the mentioned modifications. Then, P gives a walk from v_1^{out} to v_m^{in} in D', which is $P' = (v_1^{\text{out}}, v_1^{e_1}, v_2^{e_1}, v_2, v_2^{e_2}, v_3^{e_2}, v_3, v_3^{e_3}, \dots, v_m^{e_{m-1}}, v_m^{\text{in}})$, where we omitted the arcs of P' (as there are no parallel arcs in D').

Now we show that $v_1^{\text{out}} \in S$ and $v_m^{\text{in}} \in T$ are not separated by F'', which is a contradiction. Let $i \in [m-2]$. Since P is a blocking walk, we have $x(e_i) < c^*(e_i)$. By the construction of c^* in Step 2, we have that $c^*(e) = x(e)$ holds for any $e \in F''$. Thus, $v_i^{e_i} v_{i+1}^{e_i} \notin F''$. For any $i \in \{2, \dots, m-1\}$, we have $c'\left(v_i v_i^{e_i}\right) = \infty$, and for any $i \in \{1, \dots, m-2\}$, we have $c'\left(v_{i+1}^{e_i} v_{i+1}\right) = \infty$, so these arcs are also not contained in F''. The only arcs that can be contained in F'' are $v_1^{\text{out}} v_1^{e_1}$ and $v_m^{e_{m-1}} v_m^{\text{in}}$. Moreover, if $v_1^{\text{out}} v_1^{e_1} \in F''$, then $v_1 \neq s$, and similarly, if $v_m^{e_{m-1}} v_m^{\text{in}}$, then $v_m \neq t$. However, in the case when these arcs are contained in F'', by the choice of the values k^*_v and ℓ^*_v in the output, we know that there exist arcs $f_1 = v_1 w$ and $f_2 = uv_m$ such that the following hold: the arc f_1 is at least as bad for v_1 as e_1, the arc f_2 is at least as bad for v_m as e_{m-1}, and $v_1^{\text{out}} v_1^{f_1} \notin F''$ and $v_m^{f_2} v_m^{\text{in}} \notin F''$. Indeed, if all arcs leaving v_1 that are at most as good as e_1 for v_1 are contained in F'', then by the choice of $k^*_{v_1}$ in Step 2, we have that the $k^*_{v_1}$ least-preferred incoming arcs that get dominated at v_1 include e_1, and similarly if all arcs incoming to v_m that are at most as good as e_{m-1} for v_m are contained in F'', then by the choice of $\ell^*_{v_m}$, the $\ell^*_{v_m}$ least-preferred outgoing arcs that get dominated at v_m include e_{m-1}. This contradicts the assumption that P is a blocking walk. Thus, using the arc $v_1^{f_1} v_1^{e_1}$ if $f_1 \neq e_1$, and the arc $v_m^{e_{m-1}} v_m^{f_2}$ if $f_2 \neq e_{m-1}$ (these arcs exist since f_1 and f_2 are worse than e_1 and e_{m-1}, respectively) and the arcs $v_1^{\text{out}} v_1^{f_1}$ and $v_m^{f_2} v_m^{\text{in}}$, we can obtain a walk, and thus a path as well, in $A' \setminus F''$ from S to T, which is a contradiction.

As finding a minimum cut separating two disjoint sets S and T with respect to a nonnegative capacity function can be done in polynomial time using a maximum flow algorithm after contracting both S and T, we obtain that Algorithm 2 is polynomial. □

Remark 1. *One may also ask about the complexity of the problem when the part of the objective corresponding to preference changes is measured by swap distance. By making capacity changes irrelevant — simply by increasing every capacity corresponding to an unsaturated arc to infinity (or to a sufficiently large value) — the same reduction used in Theorem 3.6 can be used to show NP-hardness. Indeed, the solutions constructed in the proof of Theorem 3.6 only swap arcs that carry nonzero flow, and after the swaps these arcs get strictly more preferred. Hence the constructed solutions satisfy the required restrictions on preference changes.*

5 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we studied several inverse optimization problems related to stable flows and circulations. We explored two different models, depending on whether only the preferences or also the capacities are allowed to be modified.

While our first algorithm solves the problem of computing new utility functions such that a given flow x becomes stable while the ℓ_1-norm of the modifications is minimized, we imposed stronger restrictions on utility modifications in our second model, that is, we only allowed the utilities to be increased on the nonempty arcs of the flow x. It remains open whether the second model is still solvable in polynomial time if we drop this restriction.

Finally, it is also interesting to pursue min-max characterization theorems, or use different metrics to measure the change, e.g. the Hamming distance.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund – grant STARTING 153093.

REFERENCES

[1] Mourad Baïou and Michel Balinski. 2002. The stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research* 27, 3 (2002), 485–503.
[2] Kristóf Bérczi, Gergely Csáji, and Tamás Király. 2024. Manipulating the outcome of stable matching and roommates problems. , 407–428 pages.
[3] Nils Bertschinger, Martin Hoefer, and Daniel Schmand. 2020. Strategic payments in financial networks. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl – Leibniz Zentrum für Informatik, Dagstuhl Publishing, Dagstuhl, Germany, article no. 46.
[4] Péter Biró and Tamás Fleiner. 2010. The integral stable allocation problem on graphs. *Discrete Optimization* 7, 1–2 (2010), 64–73.

[5] Niclas Boehmer, Robert Bredereck, Klaus Heeger, and Rolf Niedermeier. 2021. Bribery and control in stable marriage. *Journal of Artificial Intelligence Research* 71 (2021), 993–1048.
[6] Canadian Resident Matching Service. 2026. CaRMS (website). https://www.carms.ca. Accessed: 2026-02-15.
[7] Jiehua Chen, Piotr Skowron, and Manuel Sorge. 2021. Matchings under preferences: Strength of stability and tradeoffs. *ACM Transactions on Economics and Computation* 9, 4 (2021), 1–55.
[8] Eduard Eiben, Gregory Gutin, Philip R. Neary, Clément Rambaud, Magnus Wahlström, and Anders Yeo. 2023. Preference swaps for the stable matching problem. *Theoretical Computer Science* 940, A (2023), 222–230.
[9] Tamás Fleiner. 2010. On stable matchings and flows. In *Graph Theoretic Concepts in Computer Science*. Springer, Berlin, Heidelberg, Germany, 51–62.
[10] David Gale and Lloyd S. Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69, 1 (1962), 9–15.
[11] Penny E. Haxell and Gordon T. Wilfong. 2008. A fractional model of the border gateway protocol (BGP). In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 193–199.
[12] Dorit S. Hochbaum. 2018. Complexity and approximations for submodular minimization problems on two variables per inequality constraints. *Discrete Applied Mathematics* 250 (2018), 252–261.
[13] Robert W. Irving. 1994. Stable marriage and indifference. *Discrete Applied Mathematics* 48, 3 (1994), 261–272.
[14] Japan Resident Matching Program. 2026. JRMP (website). https://jrmp.jp. Accessed: 2026-02-15.
[15] Richard M. Karp. 1972. *Reducibility Among Combinatorial Problems*. Plenum Press, New York, 85–103.
[16] Young-San Lin and Thanh Nguyen. 2017. On variants of network flow stability. arXiv preprint, `arXiv:1710.03091 [cs.GT]`.
[17] National Resident Matching Program. 2025. NRMP (website). https://www.nrmp.org. Accessed: 2025-09-21.
[18] Jimmy J. M. Tan. 1991. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms* 12, 1 (1991), 154–178.