# On network design problems: fixed cost flows and the Covering Steiner Problem

Guy Even* Guy Kortsarz† Wolfgang Slany‡

July 8, 2004

**Abstract**

Network design problems, such as generalizations of the Steiner Tree Problem, can be cast as edge-cost-flow problems. An edge-cost flow problem is a min-cost flow problem in which the cost of the flow equals the sum of the costs of the edges carrying positive flow.

We prove a hardness result for the Minimum Edge Cost Flow Problem (MECF). Using the one-round two-prover scenario, we prove that MECF does not admit a $2^{\log^{1-\epsilon} n}$-ratio approximation, for every constant $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylogn}})$ .

A restricted version of MECF, called Infinite Capacity MECF (ICF), is defined. The ICF problem is defined as follows: (i) all edges have infinite capacity, (ii) there are multiple sources and sinks, where flow can be delivered from every source to every sink, (iii) each source and sink has a supply amount and demand amount, respectively, and (iv) the required total flow is given as part of the input. The goal is to find a minimum edge-cost flow that meets the required total flow while obeying the demands of the sinks and the supplies of the sources. This problem naturally arises in practical scheduling applications, and is equivalent to the special case of single source MECF, with all edges not touching the source or the sink having infinite capacity.

The directed ICF generalizes the Covering Steiner Problem in directed and undirected graphs. The undirected version of ICF generalizes several network design problems, such as: Steiner Tree Problem, $k$-MST, Point-to-point Connection Problem, and the generalized Steiner Tree Problem.

An $O(\log x)$-approximation algorithm for undirected ICF is presented. We also present a bi-criteria approximation algorithm for directed ICF. The algorithm for directed ICF finds a flow that delivers half the required flow at a cost that is at most $O(n^{\epsilon}/\epsilon^4)$ times bigger than the cost of an optimal flow. The running time of the algorithm is $O(x^{2/\epsilon} \cdot n^{1+1/\epsilon})$, where $x$ denotes the required total flow.

Randomized approximation algorithms for the Covering Steiner Problem in directed and undirected graphs are presented. The algorithms are based on a randomized reduction to a problem called $\frac{1}{2}$-Group Steiner. In undirected graphs, the approximation ratio matches the

*Dept. of Electrical Engineering-Systems, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail:guy@eng.tau.ac.il.

†Department of computer science, Rutgers University, Camden, NJ, USA. E-mail:guyk@crab.rutgers.edu.

‡Computer Science Dept., Technische Universität Wien, A-1040 Wien, Austria. E-mail:wsi@dbai.tuwien.ac.at.

approximation ratio of Konjevod *et al.* [KRS01]. However, our algorithm is much simpler. In directed graphs, the algorithm is the first non-trivial approximation algorithm for the Covering Steiner Problem. Deterministic algorithms are obtained by derandomization.

# 1 Introduction

## 1.1 Problems

Network design problems deal with finding subgraphs of minimum cost that satisfy certain constraints. The (undirected) Steiner Tree Problem is among the fundamental problems in network design [GJ79]. The input of the Steiner Tree Problem is a graph $G = (V, E)$ with edge prices $p(e)$ and a set of terminals $T \subseteq V$. The objective is to find a min-cost tree $\mathcal{T}$ that spans the terminals in $T$. In the directed version, the objective is to find a min-cost arborescence that spans the terminals in $T$.

There are several generalizations of the Steiner Tree Problem. In the Group Steiner Problem, the input consists of (disjoint) subsets of vertices $\{g_i\}_i$, and the objective is to find a min-cost subgraph that contains at least one vertex from each group (see [GKR00]). In the Covering Steiner Problem, a demand $d_i$ is associated with every group $g_i$. The objective is to find a min-cost subgraph that contains at least $d_i$ vertices from group $g_i$, for every $i$ (see [KR00]).

All these problems are special cases of the Minimum Edge Cost Flow Problem (MECF) (see [GJ79, ND32]). In the fixed cost model, the cost of a flow is the sum of the prices of edges carrying positive flow. In MECF, the objective is to find a max-flow with minimum cost in the fixed cost model. We allow multiple sources and sinks in MECF instances, where flow can be shipped from every source to every sink. In addition, a supply amount and a demand amount is attached to every source and sink, respectively. Observe that an MECF instance with multiple sources and sinks can be easily reduced to an MECF instance with a single source-sink pair.

The Infinite Capacities MECF Problem (ICF) is a restriction of MECF in which there are multiple sources and sinks and all edges have infinite capacity. Observe that in ICF (as opposed to MECF) multiple sources and sinks cannot be reduced to a single source-sink pair. In addition, the ICF problem is equivalent to the *single source* MECF problem with the restriction that only edges touching the source or the sink can have finite capacity. Edges not touching the source and the sink have infinite capacity.

## 1.2 Applications

The MECF Problem is a fundamental flow problem with many applications. A sample of these applications include optimization of synchronous networks (see [LS91]), source-location (see [AIM+00]), transportation (see [EGM+97, HY97, GL94, MW84]), scheduling (for example, trucks or manpower, see [EGM+97, L96]), routing (see [HS89]), and designing networks (for example, communication networks with fixed cost per link used, e.g., leased communication lines, see [CFG00, FA92, GC96, HS89, HY97, KP99, MMW86, MW84]).

It may seem that the ICF is a very restricted variant of MECF, as ICF is equivalent to single source MECF problem with only the edges touching the source or the sink having finite capacity. However, the ICF problem was not invented just for the sake of this paper. Quite the contrary. We have encountered the ICF problem first as one of the problems we needed to solve (from a practical point of view) while working on shift scheduling problems [GGK+03]; This is a project part of us consulted for the Ximes Inc, corporation. At the end it produced a product called OPA marketed since 2001 (see [GGK+03] for more details). The shift problem deals with finding a small collection of shifts that can be used to cover loads that vary over time. In fact, the problem

that we need to solve for the shift scheduling project is even more restricted than ICF. The shift design problem is equivalent to ICF instances with unit edge costs and an acyclic directed graph. In [GGK+03] practical heuristics are used to efficiently solve the ICF problem and the other scheduling applications.

In addition, infinite capacity edges naturally arise in Transshipment problems. Transshipment problems model multiple servers from which commodities can be delivered to clients. The supply models the amount that can be delivered by a warehouse, and the demand models the amount of commodity desired by a client.

But more than that, ICF generalizes classic and well known problems. As we said, the directed ICF generalizes the directed covering Steiner problem. Undirected ICF models several network design problems. A few examples are: The $k$-MST(see [G96]), $k$-Steiner (see [CRW-01]), Generalized Steiner (see [AKR95]), and Point-to-Point Steiner Problems (where groups are pairs). The reason that undirected ICF is so powerful is that the problem reduces to finding a min-cost subset of edges so that in the edge-induced subgraph the following holds: in every connected component, the sum of the demands is not greater than the sum of the supplies. In Appendix A, reductions of various Steiner problems to undirected ICF are presented.

We believe that this uniform formulation of several different problems can help in relating these seemingly very different problems. In fact, our research does not rule out the possibility of a constant approximation algorithm for undirected ICF. This seems interesting, as such an algorithm will give a uniform constant approximation algorithm both for the generalized Steiner problem and for the $k-$MST problem (just to name two examples).

## 1.3 Previous Results

Krumke *et al.* [KNS+98] proved a logarithmic hardness of approximation for MECF. They also presented an $F^*$-approximation, where $F^*$ denotes the maximum total flow.

Konjevod & Ravi [KR00] presented a $O(\log n \cdot \log\log n \cdot \log(\max_i |g_i|) \cdot \log q \cdot \log(\max_i |d_i|))$-approximation for the undirected Covering Steiner Problem, where $q$ denotes the number of subsets. The approximation ratio was improved to $O(\log n \cdot \log\log n \cdot \log(\max_i |g_i|) \cdot \log(q \cdot \max_i |d_i|))$ by Konjevod *et al.* [KRS01]. An improvement of this approximation ratio when $\max_i d_i = 2^{\Omega(\log^2 q)}$ is also presented in [KRS01].

The undirected ICF problem with unit supplies and demands appeared previously as the point-to-point connection problem [LMSL92], where its NP-completeness was proved.

## 1.4 Results

We improve the hardness result of approximating MECF using a reduction from one-round two-prover protocols (see [AL96]). We show that MECF with uniform edge-prices does not admit a $2^{\log^{1-\epsilon} n}$-ratio approximation for any constant $\epsilon > 0$ unless $NP \subseteq DTIME(n^{\text{polylogn}})$ . This hardness holds even if only two edge capacity values are allowed, namely, $c(e) \in \{1, poly(n)\}$, for every $e$.

We present a bi-criteria approximation algorithm for directed ICF. First, we present an algorithm that finds a flow with half the required total flow, the cost of which is $O(n^\epsilon/\epsilon^4)$ times the cost of an optimal flow. The running time of the algorithm is $O(x^{2/\epsilon} \cdot n^{1+1/\epsilon})$, where $x$ denotes the requires total flow. (Scaling is applied when the total flow $x$ is non-polynomial). Observe that this is essentially the best ratio we could expect at the moment, as even the basic *directed Steiner*

currently has only an $n^\epsilon$ approximation for every constant $\epsilon > 0$. In fact, we essentially show that the algorithm of *et al.* [CCC+99] can be generalized with new ideas, to apply to the much more general ICF problem.

Thus, the algorithm is a greedy algorithm and is similar to the algorithms of Charikar *et al.* [CCC+99] and Kortsarz & Peleg [KP99]. A bi-criteria algorithm for directed ICF is presented that increases the flow amount to $(1-\epsilon') \cdot x$ with a multiplicative overhead both in the running time and in the approximation ratio that is exponential in $1/\epsilon'$. For a constant $\epsilon'$, this implies more total flow with the same asymptotic running time and approximation ratio.

We present an $O(\log x)$-approximation algorithm for undirected ICF, where $x$ is the required total flow. Our algorithm is based on a modification of the approximation algorithm of Blum *et al.* [BRV99] for the node-weighted $k$-Steiner Tree Problem.

We present the first non-trivial approximation algorithm for the Directed Covering Steiner Problem. Our algorithm uses a randomized reduction to a new problem called the $1/2-$Group Steiner problem. In the $\frac{1}{2}$-Group Steiner Problem the objective is to find a min-cost subgraph that contains a vertex of $g_i$ for at least half the groups. We solve the $\frac{1}{2}$-Group Steiner Problem in the directed case using a reduction to the directed Steiner Tree Problem approximated by [CCC+99]. The randomized algorithm has an approximation ratio of $O(\frac{n^\epsilon}{\epsilon^2} \cdot \log n)$ and a running time $\tilde{O}(n^{3/\epsilon})$. We present a derandomization procedure using 2-universal hash functions; the increase in the running time is quadratic.

The randomized reduction from the Covering Steiner Problem to the $\frac{1}{2}$-Group Steiner Problem is also used to obtain a randomized algorithm for the Undirected Covering Steiner Problem. Our approximation ratio is $O(\log n \cdot \log\log n \cdot \log(\max_i |g_i|) \cdot \log(\sum_i d_i))$. This ratio matches the best known approximation ratio of Konjevod *et al.* [KRS01]. However, our algorithm is considerably simpler. We approximate the $\frac{1}{2}$-Group Steiner Problem by a modification of the algorithm of Garg *et al.* [GKR00] for the undirected Group Steiner Problem. Again, the derandomization via 2-universal hash functions is applied.

**Organization.** In Section 2, the problems MECF, ICF, and U-ICF are defined. In Section 3, junction trees are defined and two lemmas are presented. In Section 4, we present some simple properties of ICF and U-ICF and a relate ICF to U-ICF. In Section 5, we present a bi-criteria approximation algorithm for U-ICF. In Section 6, we show how to increase the total flow delivered by the approximation algorithm. In Section 7, we present an approximation algorithm for undirected ICF. In Section 8, we present randomized approximation algorithms for the Covering Steiner Problem (directed and undirected graphs). In Section 9, we prove a hardness result for MECF.

# 2 Problem definitions

A network is a 4-tuple $N = (V, E, c, p)$ where $(V, E)$ is a graph, $c(e)$ are edge capacities, and $p(e) \geq 0$ are nonnegative edge prices. Given a source $s$ and the sink $t$, an $st$-flow is a function defined over the edges that satisfies capacity constraints, for every edge, and conservation constraints, for every vertex, except the source and the sink. The net flow that enters the sink $t$ is called the *total flow*, and is denoted by $|f|$.

The *support* of a flow $f$ is the set of edges that either deliver positive flow, namely, the set $\{e \in E : f(e) > 0\}$, or have zero cost. We denote the support of $f$ by $\chi(f)$. The price of a subset of edges $F \subseteq E$ is the sum of the prices of edges in $F$. We denote the price of $F$ by $p(F)$. The price of a flow $f$ in the fixed cost model is $p(\chi(f))$.

The following problem is NP-Complete [GJ79, ND32].

**The Minimum Edge-Cost Flow Problem (MECF).**

**Instance:**

- A network $N = (V, E, c, p)$ consisting of a (directed or undirected) graph $(V, E)$, edge capacities $c(e)$, and edge prices $p(e)$.
- A budget $P$.

**Question:** Is there a maximum $st$-flow $f$ in $N$ such that $p(\chi(f)) \leq P$?

Instead of considering a single source and sink, one may consider a situation where there is a set of sources $S \subseteq V$ and a set of sinks $T \subseteq V$. The set of candidate flow paths consists of the set of paths from a source $s \in S$ to a sink $t \in T$. This version is reducible to an $st$-flow problem.

A *supply amount* $c(s)$ of a source $s \in S$ is an upper bound on the net flow deliverable by $s$. A *demand amount* $c(t)$ of a sink $t \in T$ is an upper bound on the net flow absorbed by $t$.

The Infinite Capacities version of MECF is defined as follows.

**The Infinite Capacities Minimum Edge-Cost Flow Problem (ICF).**

**Instance:**

- A network $N = (V, E, p)$ consisting of a (directed or undirected) graph $(V, E)$, and edge prices $p(e)$. Every edge has an infinite capacity.
- A set of sources $S \subseteq V$ and a set of sinks $T \subseteq V$. Each source $s \in S$ has a positive integral supply amount $c(s)$. Each sink $t \in T$ has a positive integral demand amount $c(t)$.
- A required integral total flow $x$ and a budget $P$.

**Question:** Does there exist a flow $f$ such that $|f| \geq x$ and $p(\chi(f)) \leq P$?

Observe that, by a simple reduction, the in-degree of sources and the out-degree of sinks can be zeroed. This can be achieved by adding dummy nodes that act as sources and sinks.

We refer to a flow $f$ with total flow $x$ as an $x$-flow. We denote by $f^*(N, x)$ an optimal $x$-flow for an ICF instance $(N, x)$. Namely, $f^*(N, x)$ is an $x$-flow with minimum support cost among the set of $x$-flows in $N$. We denote the cost of an optimal $x$-flow in $N$ by $p^*(N, x)$, namely, $p^*(N, x) = p(\chi(f^*(N, x)))$. We reformulate ICF as a search problem of finding an optimal $x$-flow (hence an ICF instance is a pair $(N, x)$ and the budget $P$ is not part of the input).

**The Unit ICF Problem (U-ICF).** The unit demands and supplies version of ICF (U-ICF) is defined for ICF instances in which $c(s) = 1$, for every source $s \in S$, and $c(t) = 1$, for every sink $t \in T$.

# 3 Preliminaries

**Reduced network $N_f$.** Consider an ICF instance $(N, x)$ and an integral flow $f$ in $N$. Suppose that $|f| < x$. Let $N_f$ be the network obtained from $N$ by the following changes:

1. The supply of every source is decreased by the amount of flow it supplies in $f$. Similarly, the demand of every sink is decreased by the amount of flow it receives in $f$.

2. The price of every edge in $\chi(f)$ is set to zero.

Observe that the reduced network $N_f$ does not have reverse edges as defined in residual networks when computing a max-flow. Indeed, there the capacity of a reverse edge is finite (since the capacity of a reverse edge equals the amount of flow along the corresponding edge), but finite capacities are not allowed in ICF.

**Bi-criteria approximation algorithm.** An algorithm $A$ is an $(\alpha, \beta)$ bi-criteria approximation for ICF if, given $(N, x)$, it computes a flow $f$ that satisfies the following two conditions:

$$\begin{aligned} |f| &\geq \alpha \cdot x \\ p(\chi(f)) &\leq \beta \cdot p^*(N, x). \end{aligned}$$

## 3.1 Junction trees

A directed graph is an *arborescence* rooted at $r$ if its underlying graph is a tree, the in-degree of the root $r$ is zero, and there is a directed path from the root $r$ to all the other vertices. A *reverse arborescence* rooted at $r$ is a directed graph such that the directed graph obtained by reversing the directions of all the arcs is an arborescence.

Consider a U-ICF instance $(N, x)$ with a set of sources $S$ and a set of sinks $T$. A *junction tree* rooted at $r$ is an edge induced subgraph $JT$ of $N$ such that: (i) $JT$ is the union of an arborescence $G_1$ and a reverse arborescence $G_2$ both rooted at $r$. (ii) The leaves of $G_1$ are sinks. (iii) The leaves of $G_2$ are sources. (iv) $G_1$ and $G_2$ have an equal number of leaves.

Observe that the total flow that can be shipped using the edges of a junction tree $JT$ equals the number of sources in $JT$.

The problem of finding a low cost junction tree is defined as follows:

**The Minimum Cost Junction Tree Problem (Min-JT).**

**Instance:**

- A network $N = (V, E, p)$ consisting of a directed graph $(V, E)$, and edge prices $p(e)$.
- A set $S$ of sources and a set $T$ of sinks.
- An integer $x$.

**Goal:** Find a min-cost junction tree $JT$ with $x$ sources (and sinks).

We denote the cost of an optimal junction tree with $x$ sources by $JT^*(N, x)$.

In [CCC+99] the $k$-Directed Steiner Problem ($k$-DS) is defined. In this problem the goal is to find a min-cost arborescence rooted at $r$ that spans at least $k$ terminals from a given set of terminals. The best known approximation algorithm for $k$-DS is given in [CCC+99] where the following theorem is proved (see also [Z97] for earlier results).

**Theorem 3.1** *[CCC+99] For any $\epsilon > 0$ there exists a $k^\epsilon/\epsilon^2$-ratio, $O(k^{2/\epsilon} \cdot n^{1/\epsilon})$-time approximation algorithm for $k$-DS.*

Min-JT generalizes $k$-DS, and therefore, it is not easier to approximate than Set-Cover. An approximation algorithm for Min-JT is obtained as follows. Guess the root $r$ of the junction tree. Apply a $k$-DS approximation algorithm on the graph with sinks as terminals and a $k$-DS approximation algorithm on the reversed graph with sources as terminals. We summarize this approximation algorithm in the following corollary.

**Lemma 3.2** *For any $\epsilon > 0$ there exists an $x^\epsilon/\epsilon^2$-ratio, $O(x^{2/\epsilon} \cdot n^{1+1/\epsilon})$-time approximation algorithm for Min-JT.*

## 3.2 The forest lemma

The following lemma shows that we may restrict flows in ICF to forests.

**Lemma 3.3** *For every ICF instance $(N, x)$, there exists an optimal $x$-flow $f^*$ such that the underlying graph of the graph induced by the edges of $\chi(f^*)$ is a forest.*

**Proof:** Let $f^*$ denote an optimal $x$-flow in $N$ with a minimum support size. Assume for the sake of contradiction that there exists a (not necessarily directed) cycle $C$ in $\chi(f^*)$. Let $e_1$ be an edge carrying minimum flow in $C$. We re-route flow along $C$ as follows. Add a circulation of $f^*(e_1)$ along $C$ in the direction opposite to the direction of $e_1$. The effect of the addition of this circulation is as follows: (i) The flow on edges with the same direction as the circulation increases by $f^*(e_1)$. (ii) The flow on edges with a direction opposite to the circulation decreases by $f^*(e_1)$. The minimality of $f^*(e_1)$ along $C$ implies that this decrease does not create negative flows. In particular, the flow on $e_1$ is zeroed.

The new flow is still an optimal $x$-flow and its support is a proper subset of $\chi(f^*)$, contradicting the minimality of $\chi(f^*)$. ∎

The following assumption is based on Lemma 3.3.

**Assumption 3.4** *The underlying graph of the support of an optimal $x$-flow is a tree.*

Assumption 3.4 is justified by the following modification. Add a new node $v$ with zero in-degree and add zero cost edges from $v$ to all the other nodes. A subset of these edges can be used to glue the forest into a tree without increasing the cost of the support. Although these glue edges do not deliver flow, we may regard them as part of the support since their cost is zero.

**Remark:** Observe that the flow tree is *not* an arborescence or even a junction tree. It is simply a tree in the undirected sense, with directions over the edges.

### 3.3 A decomposition lemma

Assumption 3.4 allows us to restrict the search to trees. A subtree $\mathcal{T}'$ of a tree $\mathcal{T}$ is said to *have an articulation vertex* $r$, if every path from a node in $\mathcal{T}'$ to a node in $\mathcal{T} - \mathcal{T}'$ traverses the node $r$. The following decomposition lemma is used for recursing when optimal $x$-flows avoid junction trees.

**Lemma 3.5** *[BRV99, KP99] Let $\mathcal{T}$ be a tree with edge costs $p(e)$. Let $p(\mathcal{T})$ denote the sum of the edge costs of edges in $\mathcal{T}$. Let $S$ be a subset of vertices in $\mathcal{T}$, and $k \leq |S|$. There exists a sub-tree $\mathcal{T}' \subseteq \mathcal{T}$ that has an articulation vertex such that*

$$\begin{array}{rcll} |\mathcal{T}' \cap S| & \in & [k, 3k], & \text{and} \\ \dfrac{p(\mathcal{T}')}{|S \cap \mathcal{T}'|} & \leq & \dfrac{p(\mathcal{T})}{|S|}. & \end{array}$$

# 4 Properties of ICF and U-ICF

In this section we state some easy properties of ICF. Since the proofs are completely standard, and for the sake of keeping the paper in a reasonable length, some proofs are omitted.

**Integrality of $f^*(N, x)$.** The Integrality of supplies and demands implies that without loss of generality $f^*$ is integral and hence may be decomposed into unit flow paths.

**Claim 4.1** *Let $x$ denote an integer. Every ICF instance that allows an $x$-flow has an optimal integral $x$-flow.*

**Proof:** We show how to compute an integral $x$-flow $f$ from a non-integral $x$-flow $g$ such that $p(\chi(f)) \leq p(\chi(g))$. Let $g$ denote a non-integral optimal $x$-flow of $N$. Construct a network $N'$ that contains only edges in the support of $g$. Every $x$-flow in $N'$ is also an $x$-flow in $N$. Compute an integral $x$-flow $f$ in $N'$ (one exists since supplies and demands are all integral). Since $\chi(f) \subseteq \chi(g)$, it follows that $p(\chi(f)) \leq p(\chi(g))$. It follows that $f$ is an integral optimal $x$-flow in $N$, as required. ∎

Observe that in a U-ICF instance the unit supplies and demands imply that every integral $x$-flow induces a matching between sources and sinks.

**$x$-approximation.** Krumke *et al.* prove that MECF admits an $F$-approximation, where $F$ denotes the maximum total flow [KNS+98]. Their algorithm uses a min-cost max-flow algorithm. The flow along every edge in the support of a min-cost max-flow is in the interval $[1, F]$, and therefore, an $F$-approximation follows.

In the case of U-ICF an $x$-approximation can be obtained by a min-weight matching of size $x$ between sources and sinks. The weight of an edge between a source and a sink is the minimum cost of a path between them. The complexity of computing a min-weight matching is $O(n^3)$.

**Monotonicity.** Consider an ICF instance $(N, x)$ and an integral flow $f$ in $N$. We now wish to solve the ICF instance $(N_f, x - |f|)$ since an $(x - |f|)$-flow in $N_f$ plus $f$ constitutes an $x$-flow in $N$. The problem with this approach is that $p^*(N_f, x - |f|)$ might be larger than $p^*(N, x)$. Say that the source $s_1$ can deliver a unit of flow either to $t_1$ or to $t_2$ with a price of 1. The source $s_2$ can deliver a unit of flow to $t_2$ with a price of 1, or deliver a unit to $t_1$ with a price of $p \gg 1$. Suppose $f$ is a flow of one unit from $s_1$ to $t_2$. It follows that $s_2$ must deliver a unit of flow to $t_1$, but that is very costly. Observe that the usage of a residual graph as used in max-flow algorithms (i.e., reverse edges with a capacity equal to the flow along the edge) is not possible in ICF problems because all edge capacities must be infinite in ICF. The development of an incremental non-backtracking algorithm therefore requires that a *monotonicity property* holds.

**Definition 4.2** *An ICF instance* $(N, x)$ *is* monotone *with respect to a flow* $f$ *if*

$$p^*(N_f, x - |f|) \leq p^*(N, x).$$

There are two important cases in which monotonicity always holds: undirected networks and networks with a single source (or a single sink). We refer to an ICF instance with a single source as a *rooted* ICF instance. Observe that if the capacities are polynomial in $n$, the rooted ICF is identical to the $k$-Steiner Problem [RSMRR-96]. It is easily seen that the $k-$Steiner problem is a special case of rooted $ICF$. We can use a solution for $k-$Steiner to solve rooted $ICF$ by adding $c(t)$ edges leaving every sink $t$; $c(t)$ being the sum of capacities in the graph. The union of all the new vertices is declared the set of terminals. Clearly, a Steiner tree spanning $k$ vertices in the new instance translates to a $k-$flow in the $ICF$ instance.

In the following claims, $f_x^*$ denotes an integral optimal $x$-flow in $N$.

**Claim 4.3** *Monotonicity always holds in undirected networks.*

**Proof:** The proof is similar to the reduction of a bi-criteria approximation algorithm to an approximation algorithm. For simplicity consider a U-ICF instance $(N, x)$ with an undirected network $N$. Consider a flow $f$ in $N$. We wish to prove that $p^*(N_f, x - |f|) \leq p^*(N, x)$. Let $M^*$ and $M_f$ denote the matching between sources and sinks induced by $f_x^*$ and $f$, respectively. By Claim B.1 in the appendix, there exists a matching consisting of $x - |f|$ edges between $f$-vacant sources and $f$-vacant sinks. This matching induces $x - |f|$ paths from $f$-vacant sources to $f$-vacant sinks in $\chi(f) \cup \chi(f_x^*)$. These paths are augmenting paths. These paths define a flow of $x - |f|$ units in $N_f$ the cost of which is bounded by $p^*(N, x)$, and the claim follows. ∎

**Claim 4.4** *Monotonicity always holds in rooted ICF.*

**Proof:** Consider a flow path $f_p$ from the source $s$ to a sink $t$. The flow path $f_p$ can only block flow paths of $f_x^*$ that end in $t$. The amount of flow that is blocked by an integral flow $f$ is bounded by $|f|$. Hence, by removing $|f|$ units of flow from $f_x^*$, we obtain an $(x - |f|)$-flow in $N_f$ whose support is contained in the support of $f_x^*$. ∎

Claim 4.4 is not used in the rest of the paper, but seemed important to observe. The following claim proves that monotonicity holds in a weak sense as long as $|f| \leq x/2$.

**Claim 4.5** *[Weak Monotonicity] Consider a U-ICF instance $(N, x)$ and an integral flow $f$ such that $|f| \leq x/2$. Then,*

$$p^*(N_f, x/2 - |f|)) \leq p^*(N_f, x - 2 \cdot |f|)),$$

*and*

$$p^*(N_f, x - 2 \cdot |f|) \leq p^*(N, x),$$

**Proof:** An integral optimal $x$-flow $f_x^*$ induces a matching between sources and sinks. A matched pair $(s, t)$ is said to be *touched* by $f$ if $s$ supplies flow in $f$ or $t$ receives flow in $f$. The number of touched pairs is bounded by $2 \cdot |f|$. Let $g$ denote the flow obtained from $f_x^*$ by removing all the flow paths between matched pairs that are touched by $f$. Therefore, $|g| \geq x - 2|f|$ and $\chi(g) \subseteq \chi(f_x^*)$. In addition we get that $p^*(N_f, x/2 - |f|)) \leq p^*(N_f, x - 2 \cdot |f|))$, and he claim follows. ∎

Hence, weak-monotonicity can be used in designing non-backtracking bi-criteria algorithms that deliver at most $x/2$ flow.

Finally, by standard scaling and rounding methods we get a reduction from ICF to U-ICF.

**Polynomial supplies and demands.** The reduction splits every source $s$ into $c(s)$ sources, each with a unit supply amount. The neighbor set of each source originating from $s$ equals the neighbor set of $s$. We apply the same reduction to every sink. Since the supplies and demands are polynomial, it follows that the size of the obtained network is polynomial in the size of the ICF instance.

**Non-polynomial supplies and demands.** In this case scaling and rounding are applied first to obtain an ICF instance with polynomial supplies and demands.

Let $n = |V|$ and $m = |E|$. Recall that we are looking for an $x$-flow. Fix a parameter $\gamma$ that is polynomial in $n$. Let $\varphi = \frac{x}{\gamma \cdot \max\{n,m\}}$.

We may assume without loss of generality that $c(s) \leq x$, for every source or sink $s$. We round the supplies and demands down to the nearest multiple of $\varphi$ as follows:

$$c'(s) = \varphi \cdot \lfloor \frac{c(s)}{\varphi} \rfloor.$$

Let $N'$ denote the network obtained by rounding the supplies and demands in $N$. Consider the support $\chi(f_x^*)$ of an optimal $x$-flow $f_x^*$ in $N$. The total flow across every cut of this support is at least $x$. The rounding reduces the capacity of every cut by less than $\varphi \cdot m$. Hence, by the min-cut max-flow theorem, it follows that one can deliver more than $x - \varphi \cdot m$ flow in $N'$ using only edges in $\chi(f_x^*)$. The ICF instance corresponding to $N'$ requires a total flow $x' = \varphi \cdot \lfloor \frac{x}{\varphi} \cdot \left(1 - \frac{1}{\gamma}\right) \rfloor$. Observe that $x' < x - \varphi \cdot m$ and that $p^*(N', x') \leq p^*(N, x)$. Moreover, every $x'$-flow in $N'$ is also an $x'$-flow in $N$.

Since all supplies and demands are integral multiples of $\varphi$ in $N'$, an argument similar to Claim 4.1 is applicable. Namely, there exists an optimal $x'$-flow $f_{x'}^*$ in $N'$ such that $f_{x'}^*$ delivers an integral multiple of $\varphi$ along every edge.

We now apply scaling. Let $N''$ denote the network obtained from $N'$ by scaling demands and supplies as follows: $c''(s) = c'(s)/\varphi$. Let $x'' = x'/\varphi$. The total flow required in $N''$ is $x''$. Observe that the network $N''$ has polynomial supplies and demands (i.e., the maximum supply is bounded by $\gamma \cdot \max\{n, m\}$). Every $x''$-flow in $N''$ induces an $x'$-flow in $N$, and $p^*(N'', x'') \leq p^*(N, x)$.

We summarize this reduction in the following claim.

**Claim 4.6** *Let $\gamma$ be polynomial in $n$. If there exists an $(\alpha, \beta)$ bi-criteria poly-time approximation algorithm for U-ICF, then there exists an $(\alpha \cdot (1 - \frac{1}{\gamma} - \frac{1}{\gamma \cdot \max\{n,m\}}), \beta)$ bi-criteria poly-time approximation algorithm for ICF.*

Observe that the total flow that the claim guarantees for an ICF instance almost equals $\alpha \cdot (1 - \frac{1}{\gamma})$; the $o(1)$ term is due to the rounding procedure. Thus, in the sequel we restrict our attention to U-ICF.

# 5 An approximation algorithm for U-ICF: directed networks

In this section we present a bi-criteria approximation algorithm for U-ICF that achieves an $(\frac{1}{2}, O\left(\frac{x^\epsilon}{\epsilon^4}\right))$-approximation ratio. The algorithm only finds an $x/2$-flow due to monotonicity (see Claim 4.5). The running time of the algorithm is $O\left(x^{2/\epsilon} \cdot n^{1+1/\epsilon}\right)$ which $n$ times the running time of the Charikar *et al.* [CCC+99] algorithm. Our algorithm and its analysis are closely related to the [KP99] and [CCC+99] algorithms.

**Notation.** Fix $1/3 > \epsilon > 0$. We denote the approximation ratio for the Directed $k$-Steiner Trees by $\tau(k)$, namely, $\tau(k) = k^\epsilon/\epsilon^2$. The *density* of a flow $f$ is the ratio $\frac{p(\chi(f))}{|f|}$. We denote the density of a flow $f$ by $\gamma(f)$.

## 5.1 Motivation

In the *submodular cover* scenario, a covering problem $\mathcal{P}$ is given. The optimum solution $OPT$ for the problem can be decomposed into a union of some "local components" $OPT_i$. With each $OPT_i$ there is an associated cost and an associate part of the problem that $OPT_i$ covers. The sum of costs of $OPT_i$ is proportional to the cost of $OPT$. Moreover, the sum of amounts that the $OPT_i$ cover, is larger than what is required by $\mathcal{P}$. A classical example is the set-cover problem. In the set cover problem the optimum collection of covering sets $OPT = \{S_1, \ldots\}$ is decomposed into $OPT_i = \{S_i\}$.

In case a decomposition as above is possible, by an averaging argument, there is a single $\mathcal{P}_i$ whose density, namely, its cost over the amount that it covers, compares well with the optimum density. Thus, it is well known since [J74, W82] that submodular cover problems admit a greedy algorithm (that picks at every iteration the $\mathcal{P}_i$ with minimum density) that delivers a logarithmic approximation.

It is difficult to directly apply the above scenario to ICF. Note that by Assumption 3.4 the underlying support is a tree, and the edges of the tree have directions. We would like to decompose the tree into its immediate subtrees. A subtree $T_i$ can provide all the flow paths that are confined to $T_i$. The optimum tree does not easily decomposes in this way because flow paths can cross from one tree $T_i$ into another. To overcome this, we do the following. Consider a U-ICF instance $(N, x)$. If the tree $\mathcal{T}$ has a node $r$ that belongs to a constant fraction of the flow paths, then $r$ together with the flow paths that traverse it constitutes a junction tree $JT$. Note that the number of sources and sinks in $JT$ is a constant fraction of the total flow $x$. Moreover, the cost of $JT$

is bounded by $p^*(N,x)$. If, on the other hand, the flow traversing every node in $\mathcal{T}$ is less than a fraction of $x$, then $\mathcal{T}$ has a subtree $\mathcal{T}'$ that contains a constant fraction of the flow paths and $p(\mathcal{T}')/p(\mathcal{T})$ is constant. The following lemma formalizes this observation which is the basis for the approximation algorithm.

**Lemma 5.1** *If $k \leq x/6$, then*

$$\min\left\{JT^*(N,k), \frac{x}{6k}\cdot p^*(N,k)\right\} \leq p^*(N,x)$$

**Proof:** Let $f_x^*$ denote an optimal $x$-flow in $N$. By Assumption 3.4, we may assume that the underlying graph of the support of $f_x^*$ is a tree $\mathcal{T}$. Let $S$ denote the active sources in $f_x^*$ (i.e., $|S| = x$). By Lemma 3.5, there exists a subtree $\mathcal{T}' \subseteq \mathcal{T}$ that has a root $r$ with at least $2k$ and no more than $6k$ sources, so that $p(\mathcal{T}')/|S \cap \mathcal{T}'| \leq p(\mathcal{T})/|S|$. We consider the following two cases.

1. If at least $k$ sources in $\mathcal{T}'$ deliver their flow to sinks outside $\mathcal{T}'$, then $\mathcal{T}$ contains a junction tree rooted at $r$ the root of $\mathcal{T}'$. This junction tree has at least $k$ sources and sinks, and hence $JT^*(N,k) \leq p(\mathcal{T}) = p^*(N,x)$.

2. If less than $k$ sources in $\mathcal{T}'$ deliver their flow to sinks outside $\mathcal{T}'$, then there are at least $k$ flow paths within $\mathcal{T}'$, implying that $p^*(N,k) \leq p(\mathcal{T}')$. However, $p(\mathcal{T}') \leq \frac{6k}{x}\cdot p(\mathcal{T})$, and the lemma follows.

∎

## 5.2 The algorithm

The approximation algorithm Find-Flow for U-ICF is listed in Figure 1. The instance of $ICF$ requires $x$ flow units. Our algorithm relaxes that condition and finds only $x/2-$flow. The algorithm computes a flow by computing a sequence of augmenting flows. Note that $\epsilon$ is a parameter that effects the approximation ratio and the running time. Algorithm Find-Flow invokes two procedures. The first procedure is *Min-W-Matching*$(N,d)$ that finds a $d$-flow that is $d$-approximate using a min-weight matching, as described in Section 4. The second procedure, *Find-Aug-Flow*$(N',i)$, computes an augmenting flow $a_i$ in $N'$ such $|a_i|$ is at least $\frac{1-\epsilon}{6}\cdot i$.

The parameter $t$ is a threshold parameter used for a stopping condition of the recursion. The idea is to use an $x-$dependent threshold $t = \tau(x)$ for the stopping condition of the recursion. When the stopping condition is met (namely, the residual required flow units is at most $\tau(x)$), we use a simple brute force min-cost matching solution to find the remaining flow, as described above. The outer call is thus to $Find-Flow(N, x/2, \tau(x))$. Finally, observe that we always employ *Find-Aug-Flow* to find $\epsilon\cdot d$ flow assuming that a $d-$flow exists. This implies that the second parameter in *Find-Aug-Flow* is at most $\epsilon d$ with $d$ the maximum flow value in the network. If $d$ is not an integer, than the assumption that at least $d$ flow units can be delivered in the network is equivalent to assuming $\lceil d \rceil$ possible flow units can be delivered. The way we use *Find-Aug-Flow* makes sure these assumptions are valid.

**Algorithm** $Find - Flow(N, d, t)$

1. **If** $d \leq t$ **then return** $(Min\text{-}W\text{-}Matching(N, d))$.
2. **else**
   (a) $a \leftarrow Find\text{-}Aug\text{-}Flow(N, \epsilon \cdot d)$.
   (b) **Return** $(a \cup Find - Flow(N_a, d - |a|, t))$.

**Algorithm** $Find\text{-}Aug\text{-}Flow(N', i)$

1. Remark: $\tau(x) = x^{\epsilon}/\epsilon^2$.
2. **If** $i \leq \tau(i/\epsilon)$ **then return** $(Min\text{-}W\text{-}Matching(N', i))$.
3. **else**
   (a) **For** $j = 0$ **to** $\log_{1+\epsilon} \frac{3}{1-\epsilon}$ **do**
      i. $i(j) = \frac{1-\epsilon}{6} \cdot (1+\epsilon)^j \cdot i$.
      ii. $f_j \leftarrow Find\text{-}Aug\text{-}Flow(N', i(j))$.
   (b) $f_{JT} \leftarrow JT(N', i/6)$.
   (c) Let $f$ be a flow with minimum density in $\{\{f_j\}_j, f_{JT}\}$.
   (d) **Return** $(f)$.

Figure 1: Algorithm Find-Flow.

## 5.3 Approximation ratio

We prove the approximation ratio of Algorithm $Find - Flow$ by showing that the procedure *Find-Aug-Flow*$(N', i)$ finds a flow the density of which competes with the density of an optimal $(i/\epsilon)$-flow in the reduced network. As mentioned before, we apply *Find-Aug-Flow* appropriately guaranteeing that $(i/eps)-$flow exists. However, the claim is valid even if no such flow exist. Then the price of an $(i/\epsilon)-$ flow is defined to be $\infty$. Since the density is compared with an optimal flow in a reduced network, the analysis relies on monotonicity. By Claim 4.5, monotonicity is guaranteed as long as $|f| \leq x/2$. In this section we prove the following claim.

**Claim 5.2** *Let* $c = \frac{3}{\ln 2}$ *and* $\epsilon < 1/3$. *Algorithm* $Find - Flow(N, x/2, \tau(x))$ *finds an* $x/2$*-flow* $f$ *in* $N$ *that satisfies*

$$\frac{p(f)}{p(f_x^*)} \leq \tau(x) + \frac{6}{c \cdot \epsilon^4} \cdot x^{c \cdot \epsilon} = O\left(\frac{x^{c \cdot \epsilon}}{\epsilon^4}\right).$$

Define the function $\beta(x)$ by

$$\beta(x) \triangleq \frac{6}{\epsilon^3} \cdot x^{c \cdot \epsilon}.$$

The following lemma shows three properties of the function $\beta(x)$.

**Lemma 5.3** *If* $x \geq 1$ *and* $\epsilon < 1/3$, *then*

$$\beta(i/\epsilon) \geq i/\epsilon \quad \text{if } i \leq \tau(i/\epsilon) \tag{1}$$

$$\beta(x/\epsilon) \geq \frac{6}{\epsilon} \cdot \tau(x/6) \tag{2}$$

$$\beta(x) \geq \beta(y) \cdot \frac{1+\epsilon}{1-\epsilon} \quad \text{if } y \leq x/2. \tag{3}$$

**Proof:** Equation 1 follows from $\beta(i/\epsilon) > \frac{1}{\epsilon} \cdot \tau(i/\epsilon) > i/\epsilon$. Equation 2 follows from $\frac{6}{\epsilon} \cdot \tau(x/6) = \frac{6}{\epsilon^3} \cdot (x/\epsilon)^\epsilon \cdot (\epsilon/6)^\epsilon \leq \beta(x/\epsilon)$. Equation 3 follows from $\frac{\beta(y)}{\beta(x)} \cdot \frac{1+\epsilon}{1-\epsilon} \leq \frac{\beta(x/2)}{\beta(x)} \cdot \frac{1+\epsilon}{1-\epsilon} = 2^{-c\epsilon} \cdot \frac{1+\epsilon}{1-\epsilon}$. We need to prove that $\frac{1+\epsilon}{1-\epsilon} \leq 2^{c\epsilon}$ which is equivalent to $\ln\left(1 + \frac{2\epsilon}{1-\epsilon}\right) \leq c\epsilon \cdot \ln 2$. Since $\ln(1+x) < x$, it follows that it suffices if $\frac{2\epsilon}{1-\epsilon} \leq c\epsilon \cdot \ln 2$, which holds for $c = 3/\ln 2$ and for $\epsilon < 1/3$. The lemma follows. ∎

The following claim proves that *Find-Aug-Flow*$(N', i)$ is a bi-criteria approximation algorithm in the following sense; it finds an augmenting flow $a_i$ whose density $\gamma(a_i)$ competes with the density of an $i/\epsilon$-optimal flow. We use the following notation: $f_x^*$ denotes an $x$-optimal flow in $N'$.

**Claim 5.4** *Let* $a_i =$ Find-Aug-Flow$(N', i)$. *Then the density of* $a_i$ *satisfies*

$$\gamma(a_i) \leq \beta(i/\epsilon) \cdot \gamma(f_{i/\epsilon}^*).$$

**Proof:** The proof is by induction on $i$. If $i \leq \tau(i/\epsilon)$, then $a_i$ is computed by the minimum weight matching algorithm. In this case $|a_i| = i$ and $p(a_i) \leq i \cdot p(f_i^*)$. Obviously $p(f_i^*) \leq p(f_{i/\epsilon}^*)$, so

$$\begin{aligned} \gamma(a_i) &\leq p(f_{i/\epsilon}^*) \\ &= \frac{i}{\epsilon} \cdot \gamma(f_{i/\epsilon}^*) \\ \text{(by Eq. 1)} \quad &\leq \beta(i/\epsilon) \cdot \gamma(f_{i/\epsilon}^*). \end{aligned}$$

The induction step, for $i > \tau(i/\epsilon)$, is proved by considering $f^*_{i/\epsilon}$. By Assumption 3.4 it is shown that the underlying graph of the support of $f^*_{i/\epsilon}$ is a tree. Let $S$ denote the set of active sources in $f^*_{i/\epsilon}$. Lemma 3.5 implies that there exists a subtree $\mathcal{T}$ (of the underlying graph of the support of $f^*_{i/\epsilon}$) with an articulation vertex $v$ that satisfies:

$$|\mathcal{T} \cap S| \in [\frac{i}{6\epsilon}, \frac{i}{2\epsilon}) \quad \text{and} \tag{4}$$

$$\frac{p(\mathcal{T})}{|\mathcal{T} \cap S|} \leq \gamma(f^*_{i/\epsilon}), \tag{5}$$

We now consider two cases depending on the amount of flow that traverses the articulation vertex $v \in \mathcal{T}$ in the flow $f^*_{i/\epsilon}$. If the amount of flow traversing $v$ in the flow $f^*_{i/\epsilon}$ is at least $\epsilon \cdot |\mathcal{T} \cap S|$, then this flow defines a junction tree rooted at $v$ that delivers at least $\epsilon \cdot |\mathcal{T} \cap S| \geq i/6$ units of flow. The cost of this junction tree is at most $p(f^*_{i/\epsilon})$. (Note that even though we know that all the sources of this junction tree reside in $\mathcal{T}$, it might be the case that reaching the sinks requires all the support of $f^*_{i/\epsilon}$.) Obviously, $p(JT^*(N', i/6)) \leq p(f^*_{i/\epsilon})$. By Lemma 3.2, the junction tree approximation algorithm returns a flow $f_{JT}$ that satisfies:

$$\begin{aligned} |f_{JT}| &= i/6, \quad \text{and} \\ p(f_{JT}) &\leq \tau(i/6) \cdot p(JT^*(N', i/6)) \\ &\leq \tau(i/6) \cdot p(f^*_{i/\epsilon}). \end{aligned}$$

Hence the density of $f_{JT}$ satisfies:

$$\begin{aligned} \gamma(f_{JT}) &\leq \frac{\tau(i/6)}{i/6} \cdot p(f^*_{i/\epsilon}) \\ &= \frac{6}{\epsilon} \cdot \tau(i/6) \cdot \gamma(f^*_{i/\epsilon}) \\ \text{(by Eq. 2)} \quad &\leq \beta(i/\epsilon) \cdot \gamma(f^*_{i/\epsilon}). \end{aligned}$$

Since $\gamma(a_i) \leq \gamma(f_{JT})$, the claim follows in this case.

We consider now the case that the amount of flow that traverses the articulation vertex $v \in \mathcal{T}$ is less than $\epsilon \cdot |\mathcal{T} \cap S|$. Let $f^*_{i/\epsilon} \cap \mathcal{T}$ denote the flow paths of $f^*_{i/\epsilon}$ that are fully contained in $\mathcal{T}$. The flow $f^*_{i/\epsilon} \cap \mathcal{T}$ delivers at least $(1-\epsilon) \cdot |S \cap \mathcal{T}|$ units of flow. Let $k = \epsilon \cdot (1-\epsilon) \cdot |S \cap \mathcal{T}|$. By Eq. 4 it follows that $k \in [(1-\epsilon) \cdot i/6, i/2]$. Consider the index $j$ that satisfies $i(j) \leq k < i(j)(1+\epsilon)$, and consider the recursive call *Find-Aug-Flow*$(N', i(j))$ in Line 3(a)ii. By the selection of $a_i$ (see Line 3c in *Find-Aug-Flow*), it follows that $\gamma(a_i) \leq \gamma(a_{i(j)})$. The induction hypothesis implies that

$$\gamma(a_{i(j)}) \leq \beta(i(j)/\epsilon) \cdot \gamma(f^*_{i(j)/\epsilon}). \tag{6}$$

Observe that $f^*_{i/\epsilon} \cap \mathcal{T}$ is a flow of at least $k/\epsilon > i(j)/\epsilon$ units, therefore $p(f^*_{i(j)/\epsilon}) \leq p(\mathcal{T})$. It follows that

$$\begin{aligned} \gamma(f^*_{i(j)/\epsilon}) &\leq \frac{p(\mathcal{T})}{i(j)/\epsilon} \\ &= \frac{k/\epsilon}{i(j)/\epsilon} \cdot \frac{|S \cap \mathcal{T}|}{k/\epsilon} \cdot \frac{p(\mathcal{T})}{|S \cap \mathcal{T}|} \\ \text{(by Eq. 5)} \quad &\leq (1+\epsilon) \cdot \frac{1}{1-\epsilon} \cdot \gamma(f^*_{i/\epsilon}) \end{aligned} \tag{7}$$

From $\gamma(a_i) \leq \gamma(a_{i(j)})$, and from Eq. 6 and 7 it follows that

$$\begin{aligned} \gamma(a_i) &\leq \beta(i(j)/\epsilon) \cdot \frac{1+\epsilon}{1-\epsilon} \cdot \gamma(f^*_{i/\epsilon}) \\ \text{(by Eq. 3)} &\leq \beta(i/\epsilon) \cdot \gamma(f^*_{i/\epsilon}), \end{aligned}$$

and the claim follows. ∎

Charikar *et al.* [CCC+99] proved the approximation ratio achieved by an algorithm that computes a solution by successively finding partial solutions that have a small density. The following claim summarizes the approximation ratio achieved by Algorithm $Find - Flow$ using the same result and proof.

**Claim 5.5** *Let $f$ denote a flow in $N$, $f \leq x/2$. Let $x' = x/2 - |f|$ and $N' = N_f$. Let $f_{x'} = Find - Flow(N', x', t)$. Then,*

$$\frac{p(f_{x'})}{p^*(N', x')} \leq t + \int_0^{x'} \frac{\beta(z)}{z} dz.$$

**Proof:** The proof is by induction on $x'$. If $x' \leq t$, then $f_{x'} = \textit{Min-W-Matching}(N', x')$. Hence, $p(f_{x'}) \leq x' \cdot p^*(N', x')$. As $p^*(N', x') \leq p^*(N, x)$, the induction basis follows.

The induction step is proved as follows. Let $a = \textit{Find-Aug-Flow}(N', \epsilon \cdot x')$. By Claim 5.4,

$$p(a) \leq |a| \cdot \beta(x') \cdot \frac{p^*(N', x')}{x'}. \tag{8}$$

Observe that $\beta(z)/z$ is a non-increasing function. Hence $|a| \cdot \beta(x')/x' \leq \int_{x'-|a|}^{x'} \beta(z)/z \, dz$, and

$$p(a) \leq \int_{x'-|a|}^{x'} \frac{\beta(z)}{z} dz \cdot p^*(N', x'). \tag{9}$$

The induction hypothesis applied to $x' - |a|$ and the network $N'_a$ implies that the flow $f_{x'-|a|}$ computed by Algorithm $Find - Flow(N'_a, x' - |a|, t)$ satisfies

$$p(f_{x'-|a|}) \leq \left( t + \int_0^{x'-|a|} \frac{\beta(z)}{z} dz \right) \cdot p^*(N'_a, x' - |a|). \tag{10}$$

Weak monotonicity means that

$$p^*(N'_a, x' - |a|) \leq p^*(N', x').$$

The claim follows by concatenating the intervals of integrations in Equations 9 and 10. ∎

Claim 5.2 follows from Claim 5.5 by using a zero flow $f$.

## 5.4 Time complexity

The following claim shows that the asymptotic running time of Algorithm *Find-Aug-Flow* is $n$ times bigger than that of the approximation algorithm for the directed Steiner Problem.

**Claim 5.6** *The running time of algorithm* Find-Aug-Flow$(N, i)$ *is* $O(i^{2/\epsilon} \cdot n^{1+1/\epsilon})$*, where* $n$ *denotes the number of vertices in the network* $N$*.*

**Proof:** Recall that (up to constant factors) the running time of the approximate directed Steiner Problem is $t(x)$, where $t(x) = x^{2/\epsilon} \cdot n^{1/\epsilon}$.

The running time $t'(i)$ of *Find-Aug-Flow*$(N, i)$ satisfies the following recurrence:

$$t'(i) \leq \begin{cases} n^3 & \text{if } i \leq \tau(i/\epsilon) \\ \sum_{j=0}^{\log_{1+\epsilon} 3/(1-\epsilon)} t'(i(j)) + n \cdot t(i/6) & \text{otherwise.} \end{cases}$$

A simple inductive proof gives that, if $\epsilon < 1/3$, then $t'(i) \leq n \cdot t(i)$. ∎

The following claim summarizes the running time of Algorithm $Find - Flow$.

**Claim 5.7** *The running time of algorithm* $Find - Flow(N, d, \tau(d))$ *is* $O(d^{2/\epsilon} \cdot n^{1+1/\epsilon})$*, where* $n$ *denotes the number of vertices in the network* $N$*.*

**Proof:** Let $t_{FF}(x)$ denote the running time of $Find - Flow(N, x, t)$, and $t_{FAF}(x)$ denote the running time of *Find-Aug-Flow*$(N', x)$. The running time of algorithm $Find - Flow$ satisfies the following recurrence:

$$t_{FF}(x) \leq t_{FAF}(\epsilon x) + t_{FF}\left(x \cdot \left(1 - \left(\frac{1-\epsilon}{6}\right) \cdot \epsilon\right)\right).$$

This gives a geometric sum and the claim easily follows. ∎

# 6 Improved bi-criteria approximation for U-ICF

In this section we present a bi-criteria algorithm for U-ICF that finds a $(1-\epsilon) \cdot x$-flow that is $f(\epsilon) \cdot \rho$-approximate given a $(1/2, \rho)$-bi-criteria approximation algorithm for U-ICF. Even though $f(\epsilon)$ is exponential in $1/\epsilon$, it is constant if $\epsilon$ is. The running time required to increase the flow from an $x/2$-flow to an $(1-\epsilon) \cdot x$-flow is also exponential in $1/\epsilon$, which is again constant if $\epsilon$ is.

For simplicity we assume that the number of sinks equals the number of sources, and that $x = |S|$. It is easy to reduce a U-ICF instance so that it satisfies this assumption.

## 6.1 The reachability graph and alternating trees

Given a subset $F \subseteq E$ of edges in the network $N$, we define the reachability bipartite graph $\mathcal{G} = (\mathcal{S}, \mathcal{T}, \mathcal{E})$ between sources and sinks as follows. The edge $(s, t)$ is in $\mathcal{E}$ if $t$ is reachable from $s$ in $N$ using only edges of $F$.

**Definition 6.1** *An* alternating tree *is an edge-induced subgraph* $\mathcal{T}$ *of* $\mathcal{G}$ *and is defined recursively as follows:*

1. *A single sink* $t$ *is an alternating tree.*
2. *If* $\mathcal{T}'$ *is alternating tree,* $(s, t) \in \mathcal{E}$*,* $(s, t') \in \mathcal{E}$*,* $s, t \notin \mathcal{T}'$*, and* $t' \in \mathcal{T}'$*, then* $\mathcal{T}' \cup \{(s, t)\} \cup \{(s, t')\}$ *is an alternating tree.*

An *alternating forest* is a collection of vertex-disjoint alternating trees.

The following properties are easily proven by induction.

**Claim 6.2** *For every alternating forest $\mathcal{F}$, the following holds:*

1. *Every source in the forest has degree* 2.
2. *For every tree $\mathcal{T} \in \mathcal{F}$ and sink $t \in \mathcal{T}$, there exists a perfect matching in $\mathcal{T} - t$.*

## 6.2 Collections of singletons, pairs, and alternating trees

Given a reachability graph $\mathcal{G}$, we are interested in an edge induced subgraph of $\mathcal{G}$ consisting of a disjoint collection of single sources, source-sink pairs, and alternating trees. We refer to such an edge induced subgraph as an SPT.

Consider an SPT denoted by $\mathcal{H}$, spanning all the sources and sinks. We partition $\mathcal{H}$ into two parts $\mathcal{H}_s$ and $\mathcal{H}_t$: $\mathcal{H}_s$ consists of the single sources and source-sink pairs in $\mathcal{H}$. $\mathcal{H}_t$ consists of the alternating trees in $\mathcal{H}$. Observe that, in every alternating tree, the number of sinks equals the number of sources plus 1. It follows that, in every SPT, the number of single sources equals the number of alternating trees (recall the assumption that $|S| = |T|$).

## 6.3 The flow increasing algorithm

The starting point of the flow increasing algorithm it an $x/2$-flow $f$. This flow defines a reachability graph $\mathcal{G}$. Let $\mathcal{H}$ denote the SPT induced by $f$: $\mathcal{H}_s$ contains the source-sink pairs matched by $f$ and all unmatched sources. $\mathcal{H}_t$ contains all the unmatched sinks.

The goal is to decrease the number, $z$, of single sources, since every SPT with $z$ single sources implies an $(x - z)$-flow. The algorithms stops if $z \leq \epsilon \cdot x$.

Consider the matching between sources and sinks induced by an optimal $x$-flow. We refer to a source-sink pair matched by an optimal $x$-flow as an optimum-pair. The cut $(\mathcal{H}_s, \mathcal{H}_t)$ separates at least $z$ optimum-pairs. It follows that there are $z/2$ optimum pairs that are separated either by (i) the cut between source-sink pairs in $\mathcal{H}_s$ and sinks in alternating trees in $\mathcal{H}_t$, or (ii) the cut between single sources in $\mathcal{H}_s$ and sinks in alternating trees in $\mathcal{H}_t$.

An iteration of the algorithm invokes the $(1/2, \rho)$-bi-criteria U-ICF approximation algorithm two times. Both instances are over the network $N$ with a required total flow $z/2$, and a sink set that equals the sinks in $\mathcal{H}_t$. In the first instance, the source set is the set of single sources in $\mathcal{H}_s$, and in the second instance the source set is the set of sources in the source-sink pairs in $\mathcal{H}_s$.

The algorithm keeps the cheapest $z/4$-flow among the two computed flows (recall that the U-ICF approximation algorithm only finds a flow with half the required total flow). Keeping a flow means that the edge costs of edges in the support are zeroed.

We refer to the case that the first flow is cheaper (i.e., flow from single sources to alternating trees) as a *good* iteration. Assume that an iteration is good. There are two kinds of alternating trees: (a) trees $\mathcal{T}$ in which the flow matches between a sink $t \in \mathcal{T}$ and a single source $s$. Note that such a matching may match several $t$ in the same alternating tree to several other $s$. In type (b) trees, all the sinks of which do not receive flow. Trees of type (a) are shattered into source-sink pairs, and trees of the type (b) are shattered into single sources and single sinks. Consider an alternating tree $\mathcal{T}$ of the first type. Pick exactly one single source $s$ that is matched by the flow

to a sink $t \in \mathcal{T}$. By Claim 6.2, $\mathcal{T} - t$ contains a perfect matching, hence $\mathcal{T} + s$ can be partitioned into source-sink pairs, all of which are added to $\mathcal{H}_s$. Consider an alternating tree $\mathcal{T}$ of the second type. The tree $\mathcal{T}$ is shattered into single source and sinks as follows. The sources of $\mathcal{T}$ are declared single sources, and are added to $\mathcal{H}_s$. The sinks of $\mathcal{T}$ are declared single sinks, and are added to $\mathcal{H}_t$. . Add that perfect matching and $(s,t)$ to $\mathcal{H}_s$, and delete $\mathcal{T}$ from $\mathcal{H}_t$.

We refer to the case that the second flow is cheaper (i.e., flow from pairs to alternating trees) as a *bad* iteration. Assume that an iteration is bad. For every source-sink pair $(s,t)$ in $\mathcal{H}_s$ for which $s$ is matched by the flow to a sink $t'$ in an alternating tree $\mathcal{T}'$, add $(s,t)$ and $(s,t')$ to $\mathcal{T}'$ to obtain a larger alternating tree $\mathcal{T}$, and delete $(s,t)$ from $\mathcal{H}_s$.

### 6.4 Analysis

Observe that the number of consecutive bad iterations is at most $4/\epsilon$. The reason is that in every bad iteration at least $z/4$ source-sink pairs migrate to the alternating trees. There are at most $x$ such pairs, therefore the number of consecutive bad iterations is bounded by $x/(z/4) \leq 4/\epsilon$.

In each bad iteration, the number of sinks in every alternating tree at most doubles. This implies that by the time the first good iteration occurs, the number of sinks in every alternating tree is at most $2^{4/\epsilon}$. In a good iteration, the $z/4$-flow induces a matching between single sources and sinks in alternating trees. This implies that at most $2^{4/\epsilon}$ sources are matched to sinks in the same alternating tree. It follows that a good iteration increases the number of source-sink pairs by at least $\frac{z/4}{2^{4/\epsilon}}$.

Let $f(\epsilon)$ denote the number of iterations required to reduce the number of single sources to $\epsilon \cdot x$. It follows that $f(\epsilon) = O(1/\epsilon \cdot 2^{4/\epsilon} \cdot \log(1/\epsilon))$. Therefore, both the running time and the approximation ratio are increased by a factor of $f(\epsilon)$ which is constant if $\epsilon$ is constant.

## 7 An approximation algorithm for ICF: undirected networks

In this section we present an $O(\log x)$-approximation algorithm for ICF when the network is undirected. We avoid the reduction to U-ICF to get a slightly better approximation ratio.

Consider an undirected network $N$ with a set of sources $S$ and a set of sinks $T$. Let $f_x^*$ denote an optimal $x$-flow in $N$ the support of which has the fewest number of edges. By the Forest Lemma, it follows that the underlying graph of $\chi(f_x^*)$ is a forest $F^*$. Given a tree $\mathcal{T}$, let $c(\mathcal{T})$ denote the value $\min\{c(S \cap \mathcal{T}), c(T \cap \mathcal{T})\}$, where $c(S \cap \mathcal{T})$ ($c(T \cap \mathcal{T})$) equals the sum of the supplies (demands) of the sources (sinks) in $\mathcal{T}$. Observe that a tree $\mathcal{T}$ can be used to deliver $c(\mathcal{T})$ flow. The density of a tree $\mathcal{T}$ in this case is the ratio

$$\gamma(\mathcal{T}) \triangleq \frac{p(\mathcal{T})}{c(\mathcal{T})}.$$

The $O(\log x)$-approximation algorithm is a greedy algorithm that accumulates edges in iterations until an $x$-flow is found. In each iteration, a tree $\mathcal{T}$ is found the density of which is minimum up to a constant. The edges of the tree are added to the solution. Before the next iteration, the following updates take place: (i) The supplies and demands of sources and sinks in $\mathcal{T}$ are updated

as follows: Decrease the supplies of sources in $\mathcal{T}$ such that $c(S \cap \mathcal{T})$ is decreased by $c(\mathcal{T})$. Similarly, demands of sinks in $\mathcal{T}$ are decreased by $c(\mathcal{T})$. (ii) The set of edges in $\mathcal{T}$ are given zero price (or, equivalently, these edges are contracted).

A tree $\mathcal{T}$ with an almost minimum density is found using the constant approximation algorithm for node-weighted-$k$-Steiner-Tree (W$k$ST) of Blum *et al.* [BRV99] as follows. In the node-weighted-$k$-Steiner-Tree Problem the input consists of (i) an undirected graph with edge prices $p(e)$, (ii) a set of terminal $S$ is given, each terminal $s$ has a weight $w(s)$, and (iii) a root $r$. The goal is to find a min-cost tree rooted at $r$ that spans a subset of terminals of weight at least $k$.

Consider $F^*$, the forest induced by an optimal $x$-flow $f^*_x$. Consider the updated supplies and demands of sources and sinks in the beginning of an iteration. The forest $F^*$ has at least one tree $\mathcal{T}$ the density of which is not greater than the density of $F^*$. The procedure for finding a low density tree competes with the density of $\mathcal{T}$.

The reduction is based on guessing, within a factor of 2, the total flow $c(\mathcal{T})$ that is delivered in $\mathcal{T}$. Let $k$ denote the guess (i.e., $k \leq c(\mathcal{T}) < 2k$). The algorithm considers all possible roots $r$ and applies that W$k$ST approximation algorithm twice. Both times it searches for a node-weighted-$k$-Steiner-Tree rooted at $r$. The difference is in the terminal set; the first time it uses the set of sources (with their updated supplies) and the second time it uses the set of sinks (with their updated demands).

The approximation ratio of $O(\log x)$ follows from the fact that the algorithm greedily constructs a solution by accumulating trees, the density of which is optimal within a constant. See Johnson [J74] for a proof of this special case of Claim 5.5.

The running time of each iteration is $\log x$ times the running time of the W$k$ST approximation algorithm, which is $O(n^2 \cdot \log^4 n)$. Note that in each iteration the supply of at least one source or the demand of at least one sink is zeroed. Hence, the number of iterations is bounded by $n$. We summarize the resulting algorithm in the following claim.

**Claim 7.1** *There exists a $O(\log x)$-approximation algorithm for undirected ICF that runs in $O(n^3 \cdot \log^4 n \cdot \log x)$ time.*

# 8 The Covering Steiner Problem

In this section we present a randomized approximation algorithm for the Covering Steiner Problem both in the undirected and the directed cases. The algorithm is based on a randomized reduction of the Covering Steiner Problem to the $\frac{1}{2}$-Group Steiner Problem.

The following theorem is proved in this section.

**Theorem 8.1** *The undirected Covering Steiner Problem has a randomized approximation algorithm that finds a cover that is $O(\log n \cdot \log\log n \cdot \log(\max_i |g_i|) \cdot \log(\sum_i d_i))$-approximate with high probability.*

*The directed Covering Steiner Problem has a randomized approximation algorithm that in time $\widetilde{O}(n^{3/\epsilon})$ finds a cover that is $O(\log n \cdot n^{\epsilon} \cdot \frac{1}{\epsilon^2})$-approximate with high probability.*

In the undirected case, this result matches the approximation-ratio of [KRS01].

## 8.1 The $\frac{1}{2}$-Group Steiner Problem

In the $\frac{1}{2}$-Group Steiner Problem, the input consists of a graph $G = (V, E)$ with edge prices $p(e)$ and $q$ (disjoint) subsets $\{g_i\}_i$ of vertices. A tree $\mathcal{T}$ in $G$ *covers* a set $g_i$ if $\mathcal{T} \cap g_i$ is not empty. The goal in the $\frac{1}{2}$-Group Steiner Problem is to find a min-cost tree that covers at least half the groups.

In the directed case, the $\frac{1}{2}$-Group Steiner Problem is reducible to the $k$-Directed Steiner Problem. See the reduction of the Group Steiner Tree Problem in [CCC+99]. This implies an approximation as in Theorem 3.1.

In the undirected case, the $\frac{1}{2}$-Group Steiner Problem (for every constant $\epsilon > 0$) can be solved by modifying the approximation algorithm for the Group Steiner Problem of Garg *et al.* [GKR00]. The modified algorithm has 3 stages:

First, a fractional relaxation is defined. The fractional relaxation is a min-cost single-source multi-sink flow problem. We guess the root $r$ of an optimal solution. This root serves as the source. For every group $g_i$, we define a commodity $i$ and a flow $f_i$ from $r$ to $g_i$ (namely, the sinks are the nodes of $g_i$). The total flow $|f_i|$ is at most 1. The sum of total flows $\sum_i |f_i|$ is at least $q/2$. The max-flow $f(e)$ along an edge $e$ equals $\max_i f_i(e)$. The objective is to minimize the sum $\sum_e p(e) \cdot f(e)$.

Second, following [GKR00], a rounding procedure for the above fractional relaxation is applied when the graph is a tree. If the graph is a tree, then the union of the supports $\bigcup_i \chi(f_i)$ induces a tree. Moreover, the directions of the flows induce an arborescence. Let $pre(e)$ denote the predecessor edge of the edge $e$. The rounding first picks an edge $e$ with probability $f(e)/f(pre(e))$, and then keeps $e$ provided that all the edges along the path from the root $r$ to $e$ are picked.

Observe that, for at least $q/4$ groups, $|f_i| \geq 1/4$. Otherwise the total flow is less than $q/2$. In [GKR00], it is proved that if $|f_i|$ is constant, then the rounding procedure covers $g_i$ with probability $\Omega(\frac{1}{\log |g_i|})$. Moreover, the expected cost of the picked edges is the fractional optimum. It follows that $O(\log \max_i |g_i|)$ iterations suffice to cover half the groups, and an $O(\log \max_i |g_i|)$ expected approximation ratio follows.

Finally, the graph is approximated by a tree using the tree-metric technique of Bartal [B98]. The approximation by a tree increases the approximation ratio by $O(\log n \log \log n)$. We point out that Charikar *et al.* [CCG+-98] presented a derandomization for an approximate tree metric and the above rounding technique.

It follows that in the undirected case the $\frac{1}{2}$-Group Steiner admits an $O(\log n \cdot \log \log n \cdot \log(\max_i |g_i|))$-approximation.

## 8.2 A randomized reduction

In this section we present a randomized reduction of the Covering Steiner Problem to the $\frac{1}{2}$-Group Steiner Problem.

Given a Covering Steiner instance, consider an optimal tree $\mathcal{T}$. For every group $g_i$, the tree $\mathcal{T}$ includes at least $d_i$ vertices in $g_i$. We randomly partition $g_i$ into $d_i$ bins. The $j$th bin in $g_i$ is denoted by $g_i^j$. A bin $g_i^j$ is *empty* if $\mathcal{T} \cap g_i^j$ is empty. The probability that a bin is empty is less than $1/e$ (as this corresponds to throwing $d_i$ balls to $d_i$ bins). Thus, the expected number of empty bins is less than $\sum_i d_i/e$. We consider this randomized reduction to be successful if the number of empty bins is less than $\sum_i d_i/2$. The Markov Inequality implies that the reduction is successful

with a constant probability. We may decrease the probability of a failure to a polynomial fraction by repeating the reduction $\Theta(\log n)$ times.

Observe that if the randomized reduction is successful, then the reduction yields a $\frac{1}{2}$-Group Steiner instance with a solution, the cost of which is at most the cost of $\mathcal{T}$.

The $\frac{1}{2}$-Group Steiner approximation algorithm finds a tree that covers half of the total demands. After $O(\log(\sum_i d_i)) = O(\log n)$ iterations, all the groups are covered.

## 8.3 Derandomization

The randomized reduction can be derandomized using 2-universal hash functions (c.f. [MR95]). First, we may assume that $d_i \leq n/\log_2 n$ for every $i$. Indeed, as the $g_i$ are disjoint there could be at most $O(\log n)$ "large" groups with $d_i > n/\log n$. Thus, we can consecutively compute a feasible solution containing at least $d_i$ terminals of $g_i$, for one "large" $g_i$ after the other. When there is only a single group, the resulting problem is the $d_i-$Steiner problem. In this problem we are given a unique set of terminals and the the goal is to span at least a pre-described number of terminals. The $k-$Steiner problem, in the undirected case, admits a constant approximation (see [CRW-01]). The $O(\log n)$ applications of the $d_i-$Steiner algorithm incur a cost $O(\log n)$ times the optimum covering steiner cost, which is negligible in our context.

The reduction is now to the $1/3-$group Steiner (and not $1/2-$group Steiner) problem. Let $p \leq 2n$ the minimum prime that is no smaller than $n$. For every $g_i$ create $d_i$ (initially empty) bins $B^i_j,\ 0 \leq j \leq d_i - 1$. Choose at random two numbers $a, b \in \{0, \ldots, p\}$. Ordering the $g_i$ vertices in some arbitrary order, the $j$ vertex in $g_i$ is mapped into bin $B^i_k$ with $k = (a_i + j \cdot b_i \bmod p) \bmod d_i$. The resulting bins define the groups that give the reduction to a $1/3-$group Steiner instance.

Let $T_i$ be the collection of exactly $d_i$ terminals of $g_i$ included by the optimum in the covering Steiner tree. Let $k \in \{0, \ldots, p-1\}$. Observe that the number integers between 0 and $p-1$ that are congruent to $k$ modulo $d_i$ is $\lfloor p/d_i \rfloor$. Since $a_i + j \cdot b_i$ is a random number in $\mathcal{Z}_p$, the probability that $k = (a_i + j \cdot b_i \bmod p) \bmod d_i$ is $(\lfloor p/d_i \rfloor)/p$ or $(\lceil p/d_i \rceil)/p$. Therefore,

$$\frac{1}{d_i} - \frac{1}{p} \leq Pr(v \in B^i_k) \leq \frac{1}{d_i} + \frac{1}{p}.$$

Observe that by the inclusion-exclusion formula:

$$Pr(\bigcup_{v \in T_i} v \in B^i_k) \geq \sum_{v \in T_i} Pr(v \in B^i_k) - \sum_{v,u \in T_i} Pr(v \in B^i_k \text{ and } u \in B^i_k).$$

By pairwise independence: $Pr(v \in B^i_k \text{ and } u \in B^i_k) \leq 1/d_i^2 + O(1/d_i p)$. Thus,

$$\begin{aligned} Pr(\bigcup_{v \in T_i} v \in B^i_k) &\geq \textstyle\sum_{v \in T_i} Pr(v \in T_i) - \sum_{v,u \in T_i} Pr(v \in B^i_k \text{ and } u \in B^i_k) \\ &\geq d_i \left(\tfrac{1}{d_i} - \tfrac{1}{p}\right) - \tfrac{d_i \cdot (d_i - 1)}{2} \cdot \left(\tfrac{1}{d_i^2} + O\left(\tfrac{1}{d_i p}\right)\right) = \tfrac{1}{2} - o(1) \end{aligned}$$

We used the inequality $d_i \leq n/\log_2 n < p$ to prove the last inequality.

Since a bin has probability at most $1/2 - o(1)$ to be empty, the expected number of nonempty bins over all groups is at least $\sum d_i/2 - o(\sum d_i) \geq \sum d_i/3$. Choosing the best $a$ and $b$ pair (the one giving minimum cost), we get an instance of the $1/3-$group Steiner problem of cost at most the optimum.

The derandomization incurs a quadratic increase in the running time.

# 9 A hardness result for MECF

It is natural to ask if a strong lower bound on the approximability of ICF holds. We leave this question open. However, we try to make a first step toward this direction, by giving a strong lower bound for MECF. In addition, the study of MECF is interesting by its own right, for the many important applications of this problem.

**Theorem 9.1** *The Minimum Edge-Cost Flow Problem with uniform edge-prices does not admit a $2^{\log^{1-\epsilon} n}$-ratio approximation for any constant $\epsilon > 0$ unless NP $\subseteq$ DTIME($n^{\text{polylogn}}$) . This hardness holds even if only two edge capacity values are allowed, namely, $c(e) \in \{1, poly(n)\}$, for every $e$.*

Non-uniform polynomial edge prices are easily reducible to uniform edge prices. The reduction replaces every edge $(u, v)$ by a path of length $p(u, v)$.

## 9.1 Background

We essentially prove hardness by giving a reduction from a one-round two-prover, interactive proof system. In [AL96] the Labelcover-Max Problem is introduced using graph theory for presenting the one-round two-provers scenario. We use an alternative formulation, called $Max - Rep$, of the Labelcover-Max Problem defined in [K98].

In $Max - Rep$ we are given a bipartite graph $\mathcal{B}(\mathbf{V_1}, \mathbf{V_2}, \mathbf{E})$. The sets $V_1$ and $V_2$ are partitioned into a disjoint union of $k$ sets: $V_1 = \bigcup_{i=1}^{k} A_i$ and $V_2 = \bigcup_{j=1}^{k} B_j$. The sets $A_i$ and $B_j$ all have size $N$. The bipartite graph and the partition of $V_1$ and $V_2$ induce a super-graph $\mathcal{H}$ in the following way: The vertices in $\mathcal{H}$ are the sets $A_i$ and $B_j$. Two sets $A_i$ and $B_j$ are connected by a (super) edge in $\mathcal{H}$ iff there exist $a_i \in A_i$ and $b_j \in B_j$ which are adjacent in $G$. For our purposes, it is convenient (and possible) to assume that graph $\mathcal{H}$ is regular. Say that every vertex in $\mathcal{H}$ has degree $d$, and hence, the number of super-edges is $h = k \cdot d$.

In $Max - Rep$ the goal is to select a unique "representative" vertex $a_i \in A_i$ from each subset $A_i$, and a unique "representative" vertex $b_j \in B_j$ from each $B_j$. We say that a super-edge $(A_i, B_j)$ is *covered* if the two corresponding representatives are neighbors in $G$, i.e., $(a_i, b_j) \in E$. The goal is to select unique representatives so as to maximize the number of covered super-edges.

The Satisfiability Problem ($SAT$) is defined as follows. A CNF Boolean formula $I$ is given, and the question is whether there is an assignment satisfying all the clauses. Raz proved the following result [R95].

**Theorem 9.2** *[R95] Let $I$ be an instance of SAT. For any $0 < \epsilon < 1$, there exists a (quasi-polynomial) reduction of each instance of the Satisfiability Problem to an instance $G$ of $Max - Rep$ with $n$ vertices. If $I$ is satisfiable, then there exists a set of unique representatives which covers all $h = k \cdot d$ super-edges. If $I$ is not satisfiable, then every set of unique representatives covers no more than $h/2^{\log^{1-\epsilon} n}$ of the super-edges.*

In the above reduction, the size, $n$, of the $Max - Rep$ instance, $G$, is quasi-polynomial in the size of the SAT formula. The following easily follows from Theorem 9.2.

**Theorem 9.3** *Unless NP $\subseteq$ DTIME($n^{\text{polylogn}}$), the $Max - Rep$ Problem admits no $2^{\log^{1-\epsilon} n}$-ratio approximation, for any $\epsilon > 0$.*

## 9.2 Proof of Theorem 9.1

**Proof:** We reduce $Max-Rep$ to MECF as follows. Let $\mathcal{B}$ be the bipartite instance of $Max-Rep$ at hand. Form an instance $G$ for MECF as follows.

Figure 2 depicts the instance $G$ of MECF. Add $\mathcal{B}$ into $G$. Give all the edges of $\mathcal{B}$ directions from the $A_i$ vertices to the $B_j$ vertices. The edges of $\mathcal{B}$ are assigned zero cost and capacity 1. Add a source $s$ and a sink $t$. Add a special vertex $a_i$ for every set $A_i$ and a vertex $b_j$, for every set $B_j$. Introduce a zero cost edge $(s, a_i)$, for every $i$, with capacity $d$ (where $d$ is the degree of every $A_i, B_j$ in the super-graph $\mathcal{H}$). Add an edge $(a_i, v)$, for every $v \in A_i$, with capacity $d$ and cost $n^3$ (where $n$ is the number of vertices in $\mathcal{B}$). Add an edge $(u, b_j)$ with capacity $d$ and cost $n^3$, for every $u \in B_j$. Finally, add a zero cost edge $(b_j, t)$ with capacity $d$, for every $j$.

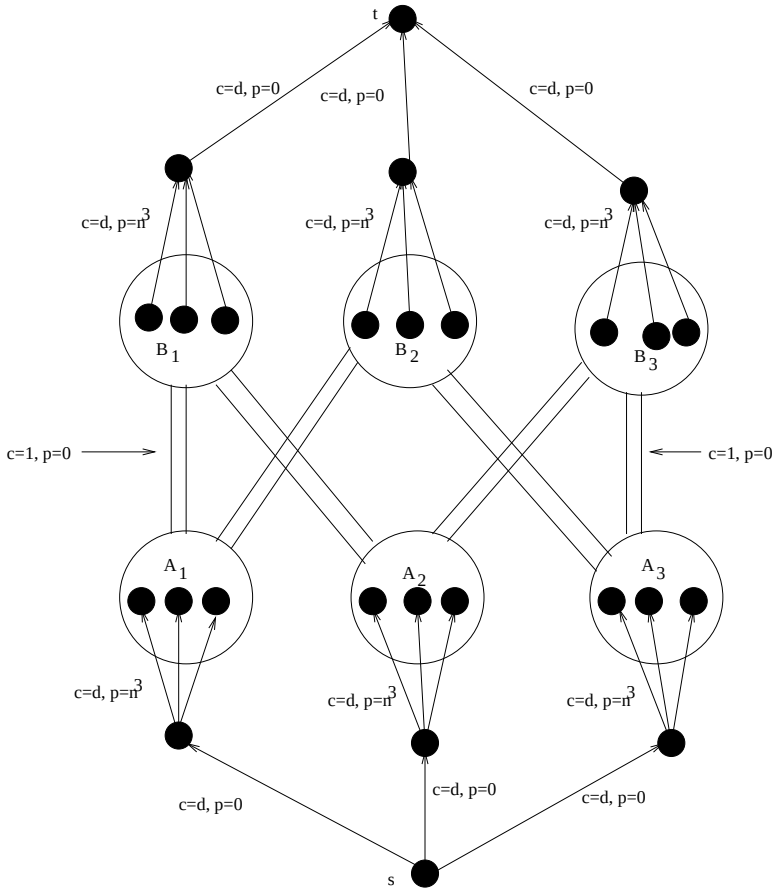


Figure 2: A reduction of $Max - Rep$ to MECF.

A direct inspection shows that the maximum flow in the network $G$ is $h = dk$.

Let $\mathcal{B}$ be a $Max-Rep$ instance resulting from a yes instance of SAT and let $G_Y$ be the resulting MECF instance.

**Lemma 9.4** *The graph $G_Y$ admits a maximum flow of $h$ with cost $2k \cdot n^3$.*

**Proof:** Let $v_i \in A_i$, $u_j \in B_j$ be a collection of unique representatives covering all the super-edges in $\mathcal{H}$. Deliver $d$ units of flow from $s$ to $a_i$ and into $v_i$. Let $(A_i, B_j)$ be a super-edge. Deliver one unit of flow from $v_i$ to $u_j$. Note that the edge $(v_i, u_j)$ exists as the chosen representatives cover all the super-edges. From $u_j$ the flow continues to $b_j$ and into $t$. The cost incurred is $2 \cdot k \cdot n^3$, as required. ∎

We prove that only yes instances reduce to MECF instances with flows that cost roughly $2k \cdot n^3$. We prove this by counter-position; namely, we show how to find a cover of at least $h/(8192\rho^4)$ super-edges from every max-flow that costs $\rho \cdot 2k \cdot n^3$.

**Lemma 9.5** *Given a solution $\mathcal{S}$ to an MECF instance of cost $\rho \cdot 2k \cdot n^3$, it is possible to find (in polynomial time) a collection of unique representatives covering at least $h/(8192\rho^4)$ of the super-edges.*

**Proof:** Call a vertex $v$ *active* (in $\mathcal{S}$, namely, in the MECF assumed solution) if (positive) flow traverses (enters and leaves) $v$. Let $X_i$ (respectively, $Y_j$) be the collection of active vertices in $A_i$ (respectively, $B_j$).

The sets $X_i$ and $Y_j$ may contain many vertices in $A_i$ and $B_j$, respectively. The cost incurred by flow along the edges from the vertex $a_i$ to vertices $X_i$ and from $Y_j$ to the vertex $b_j$ is $\sum_i (|X_i| + |Y_i|) \cdot n^3$. It follows that $\sum_i |X_i| \leq \rho \cdot 2k$ and $\sum_i |Y_i| \leq \rho \cdot 2k$. The average of $|X_i|$ ($|Y_j|$) is at most $2\rho$. Call an $A_i$ "bad" if $|X_i| > 8\rho$. Similarly, $B_j$ is bad if $|Y_j| > 8\rho$.

Remove from the super-graph $\mathcal{H}$ all the bad sets $A_i$ and $B_j$. Clearly, the number of bad $A_i$ sets is no larger than $k/4$ and the same holds for $B_j$. Now we update (namely, reduce) the flow to a legal one. The loss of flow incurred by the removal of a bad $A_i$ or bad $B_j$ is at most $d$. Hence, the removal of bad $A_i$ and $B_j$ sets incurs a loss of at most $2 \cdot k/4 \cdot d = h/2$. Hence, at least $h/2$ flow units still reach $t$ after this update.

We now dilute the flow so that at most one flow path remains between every pair of sets $A_i, B_j$. Since the remaining sets $A_i$ and $B_j$ are not bad, the number of active vertices in each set is bounded by $8\rho$. Hence, the total flow between every pair of sets $A_i$ and $B_j$ is at most $(8\rho)^2$. Therefore, the dilution results with a total flow of at least $\frac{h}{128 \cdot \rho^2}$.

Observe that the number of active vertices in each set $A_i$ and $B_j$ is still more than one (but less than $8\rho$). Unique representatives are drawn uniformly at random among the active vertices in each set $A_i$ and $B_j$. The expected number of super-edges covered is at least $h/(128\rho^2 \cdot 64\rho^2) = h/(8192\rho^4)$. This implies that under the best choice of unique representatives at least this many super-edges can be covered. We can find such a collection of unique representatives in polynomial time using the method of conditional expectations. ∎

Now consider a yes instance of SAT. Assume we have a $\rho$-ratio approximation of MECF for $\rho = o(n)$. The corresponding graph $G_Y$ admits (by Lemma 9.4) a $2k \cdot n^3$ cost solution. Thus, by the assumed approximation ratio, a $2\rho k n^3$ cost solution for $G_Y$ is computed. By Lemma 9.5 a solution to $Max - Rep$ is found covering at least $h/(8192\rho^4)$ of the super-edges.

Let $G_N$ denote a graph corresponding to a no instance of SAT. The best choice of representatives in $\mathcal{B}$ covers at most $h/2^{\log^{1-\epsilon} n}$ super-edges. Hence, if $8192\rho^4 < 2^{\log^{1-\epsilon} n}$, then a $\rho$-ratio approximation algorithm gives a separation between yes and no instances of SAT. This implies that it is hard to approximate MECF within $2^{1/4 \log^{1-\epsilon} n}/10$, and the hardness result follows.

Finally, observe that the edge prices are not uniform in the reduction. Uniform edge prices can be obtained by (a) setting the price of every zero cost edge to be 1, and (b) splitting every edge

of cost $n^3$ into a path of $n^3$ unit price edges. The same analysis holds, since the increase in the cost of an $h$- flow is $h + 2k = O(n^2)$. This increase is $o(n^3)$, and does not effect the analysis. ∎

# Acknowledgments

We thank Roby Krautgammer for pointing out that 2-universal hash functions suffice for derandomizing the reduction from the Covering Steiner Problem to the $\frac{1}{2}$-Group Steiner Problem. We thank an anonymous referee for helpful comments.

# References

[AKR95] Ajit Agrawal, Philip Klein, and R. Ravi, "When trees collide : An approximation algorithm for the generalized Steiner problem on networks", SIAM Journal on Computing, 24(3) 445-456 (1995).

[AIM+00] K. Arata, S. Iwata, K. Makino and S. Fujishige. Source location: Locating sources to meet flow demands in undirected networks. *SWAT 2000.*

[AL96] S. Arora and C. Lund, "Hardness of Approximations", In *Approximation Algorithms for NP-hard Problems*, Dorit Hochbaum, Ed., PWS Publishing , 1996.

[B98] Y. Bartal, "On approximating arbitrary metrics by tree metrics", STOC, 1998.

[B93] M. Bellare. Interactive proofs and approximation: reduction from two provers in one round. *The second Israeli Symposium on the Theory of Computing*, pages 266–274, 1993.

[BRV99] A. Blum and R. Ravi and S. Vempala A constant-factor approximation algorithm for the $k$-MST Problem. *JCSS*, 58:101–108, 1999.

[CFL+00] R.D. Carr, L.K. Fleischer, V.J. Leung and C.A. Phillips. Strengthening Integrality Gaps for Capacitated Network Design and covering Problems. *Proc. of the 11th ACM/SIAM Symposium on Discrete Algorithms*, January 2000.

[CCC+99] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha and M. Li. "Approximation Algorithms for directed Steiner Problems", J. of Algs., 33, p. 73-91, 1999.

[CCG+-98] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin "Approximating a finite metric by small number of trees", FOCS, 1998.

[CRW-01] F. Chudak and T. Roughgarden and D. Williamson. Approximate k-MSTs and k-Steiner Trees via the Primal-Dual Method and Lagrangean Relaxation. *IPCO*, pages 60-70, 2001.

[CCP+98] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver *Combinatorial Optimization*, John Wiley and Sons, New York, 1998.

[CFG00] T.G. Crainic, A. Frangioni and B. Gendron. Bundle-based Relaxation Methods for Multicommodity Capacitated Fixed Charge Network Design Problems. *Discrete Applied Mathematics*, to appear, 2000.

[CK94] P. Crescenzi and V. Kann. A compendium of NP optimization problems, 1994. *See ftp://www.nada.kth.se/~viggo/problemlist/compendium.html*

[D00] Reinhard Diestel, Graph Theory, 2nd Edition, *Springer-Verlag*, 2000. url: http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[EGM+97] L. Equi, G. Gallo, S. Marziale and A. Weintraub. A combined transportation and scheduling problem. *European Journal of Operational Research*, 97(1):94–104, 1997.

[FA92] P.C. Fetterolf and G. Anandalingam. A Lagrangian relaxation technique for optimizing interconnection of local area networks. *Oper. Res.*, 40(4):678–688, 1992.

[GJ79] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *W.H. Freeman and Company*, 1979.

[G96] N. Garg. A 3-approximation for minimum spanning tree spanning $k$ vertices. *the 37th Symposium on Foundation of Computer Science*, 302-309, 1996.

[GVY96] N. Garg, M. Yannakakis and V.V. Vazirani. Approximating max-flow min-(multi)cut theorems and their applications. *Siam J. on Computing*, 25:235-251, 1996.

[GVY97] N. Garg, M. Yannakakis and V.V. Vazirani. Primal-dual approximation algorithms for integral flow and multicuts in trees. *Algorithmica*, 18:3–20.

[GKR00] N. Garg and G. Konjevod and R. Ravi, A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. of Algorithms*, volume 37, number 1, pages 66-84, 2000.

[GL94] M. Goethe-Lundgren and T. Larsson. A set covering reformulation of the pure fixed charge transportation problem. *Discrete Appl. Math.*, 48(3):245–259, 1994.

[GGK+03] Luca Di Gaspero and Johannes Gärtner and Guy Kortsarz and Nysret Musliu and Andrea Schaerf and Wolfgang Slany, The minimum shift design problem: theory and practice. *The European Symposium on Algorithms (ESA)* 2003, pages 593–604.

[GC96] B. Gendron and T.G. Crainic. Bounding Procedures for Multicommodity Capacitated Fixed Charge Network Design Problems. *Publication CRT-96-06*, Centre de recherche sur les transports, Université de Montréal, 1996.

[GKR+99] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd and M. Yannakakis. Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and related Problems. *J. Comput. Syst. Sci.*, 67(3): 473-496, 2003.

[GW97] M.X. Goemans and D.P. Williamson, The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems, in Approximation Algorithms, D. Hochbaum, Ed., 1997.

[HS89] D.S. Hochbaum and A. Segev. Analysis of a flow problem with fixed charges. *Networks*, 19(3):291–312, 1989.

[HY97] K. Holmberg and D. Yuan. A Lagrangean heuristic based branch-and-bound approach for the capacitated network design problem. *Proceedings 4th meeting of the Nordic section of the Mathematical Programming Society, Arhus, Denmark, August 16–18, 1996.* Kim Allan Andersen (ed.), Arhus: Univ. of Arhus, Department of Operations Research, Publications at the Departments of Mathematical Sciences. *Working Papers 97-1*, pages 62–97, 1997.

[J74] Johnson, D. S. "Approximation algorithms for combinatorial problems", J. Comput. System Sci. 9, 256-278, 1974.

[KiPa99] D. Kim and P.M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Oper. Res. Lett.*, 24(4):195–203, 1999.

[KR00] G. Konjevod and R. Ravi, An Approximation Algorithm for the Covering Steiner Problem. *SODA 2000*, 338–334, 2000.

[KRS01] G. Konjevod, R. Ravi, and Aravind Srinivasan, Approximation Algorithms for the Covering Steiner Problem. Random Struct. Algorithms 20(3): 465-482 (2002)

[K98] G. Kortsarz. On the hardness of approximating spanners. *The first International Workshop Approx-98*, pages 135–146, 1998.

[KP99] G. Kortsarz and D. Peleg. "Approximating the Weight of Shallow Steiner Trees", Discrete Applied Math, vol 93, pages 265-285, 1999.

[KNS+98] S.O. Krumke, H. Noltemeier, S. Schwarz, H.-C. Wirth and R. Ravi. Flow Improvement and Network Flows with Fixed Costs. *OR-98*, Zürich, 1998.

[L96] H.C. Lau. Combinatorial approaches for hard problems in manpower scheduling. *J. Oper. Res. Soc. Japan*, 39(1):88–98, 1996.

[LS91] C.E. Leiserson and J.B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, 6(1):5–35, 1991.

[LMSL92] C. L. Li, S. T. McCormick and D. Simchi-Levi. The point-to-point delivery and connection problems: Complexity and Algorithms. Discrete Applied Mathematics, 36:267-292, 1992.

[MRS+98] Madhav V. Marathe, R. Ravi, Ravi Sundaram, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Bicriteria network design problems. *J. of Algorithms*, 28:142–171, 1998.

[MW84] T.L. Magnanti and R.T. Wong. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, 18:1–55, 1984.

[MMW86] T.L. Magnanti, P. Mireault and R.T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Math. Program. Study*, 26:112–154, 1986.

[MR95] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge Univ. Press, 1995.

[PS82] C.H. Papadimitriou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. *Prentice-Hall*, 1982.

[RSMRR-96] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees short or small", SIAM Journal on Discrete Mathematics, 9(2), 178-200, 1996.

[R95] R. Raz. A parallel repetition theorem. *Proc. 27th ACM Symposium on the Theory of Computing*, 1995, 447–456. Full version in *SIAM J. Comput.*, 27(3):763–803, 1998.

[S97] A. Srinivasan. Improved approximation for edge-disjoint paths, unsplittable flow, and related routing problems. *FOCS-97*, 416–425.

[W82] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

[Z97] A. Zelikovsky. A series of approximation algorithms for the Acyclic directed Steiner Tree Problem. *Algorithmica*, 18:99–110, 1997.

# A Reductions to undirected ICF

Undirected ICF is equivalent to the problem of finding a min-cost subset of edges $F$ such that the subgraph $G_F = (V, F)$ satisfies the following property: In every connected component $H$ of $G_F$, the sum of the demands of the sinks in $H$ is not greater than the sum of the supplies of the sources in $H$.

Using this equivalent formulation we show that undirected ICF generalizes the following Steiner Problems.

1. The minimum cost Steiner Tree Problem. This problem can be reduced to undirected ICF as follows. Guess a root $r$ and set the source set $S = \{r\}$. The sink set $T$ is set to be the set of terminals. The demand of every terminal is 1, and the supply of $r$ is $|T|$. The required flow amount is $|T|$.

2. The $k$-MST Problem (see [G96]). In this problem the goal is to find a min-cost subset of edges that spans at least $k$ vertices. The $k$-MST problem can be reduced to undirected ICF as follows. Guess a root $r \in V$. Set the source set to be $\{r\}$ with $c(r) = k - 1$. Set the sink set $T$ to be $V - r$, and set the demand of every sink $t$ to be $c(t) = 1$. The required flow amount is $k - 1$.

3. The Point-to-point Connection Problem (see [LMSL92, GW97]). In this problem there is a set of $k$ sources $s_1, \ldots s_k$ and $k$ sinks $t_1, \ldots t_k$. The goal is to find a min-cost subset of edges $F$ such that each source-sink pair is connected in $F$.

   In the non-fixed destination case, every source may be connected to every sink. This case is equivalent to U-ICF where the required total flow equals the number of sources.

   The fixed destination case is a special case of the Generalized Steiner Tree Problem discussed below.

4. The Generalized Steiner Tree Problem (see [GW97, AKR95]). In this problem the input consists of a collection of terminal sets $T_i \subseteq V$, for $i = 1, \ldots, p$. The goal is to find a min-cost forest that connects, for every $i$, all the terminals in $T_i$. A special case of this problem is the Point-to-point Generalized Steiner Tree Problem, where each set of terminals $T_i$ consists of exactly two terminals.

   Observe that the Generalized Steiner Tree Problem is reducible to its point-to-point version as follows. For every $t \in T_i$, create $|T_i| - 1$ leaves connected to $t$ by zero-cost edges. The index set of the leaves created for $t$ is $T_i - t$ (i.e., $t_{t'}$ denotes a leaf connected to $t$ whose index is a terminal $t' \in T_i - t$). For every $T_i$, define $\frac{1}{2}|T_i| \cdot (|T_i| - 1)$ sets of terminal pairs $\{s_t, t_s\}$, where $s, t \in T_i$ and $s \neq t$. Observe that $T_i$ is connected iff all these pairs are connected (we add all zero cost edges to the forest).

   Finally we reduce the Point-to-point Generalized Steiner Tree Problem to undirected ICF as follows. Given a collection $\{s_i, t_i\}_{i=1}^k$ of terminal pairs, the goal is to find a min-cost forest that connects every terminal pair. The ICF problem defines the source set to be $\{s_i\}_i$ and the sink set to be $\{t_i\}_i$. We assign the supply amount of $s_i$ to be $c(s_i) = 2^{i-1}$, and the demand amount of $t_i$ is set to be also $c(t_i) = 2^{i-1}$. The required total flow is $\sum_i c(s_i) = 2^k - 1$. A direct inspection shows that every feasible solution of the ICF instance must connect every terminal pair $s_i, t_i$.

   Note that the total flow amount is $2^k - 1$, and hence our undirected ICF approximation only yields trivial approximation ratios for these problems.

# B Augmenting paths in bipartite graphs

**Claim B.1** *Let $M_1$ and $M_2$ denote matchings in a bipartite graph $G(A, B, E)$. Assume that $|M_1| < |M_2|$. There exists a set $\Pi$ of maximal paths in the symmetric difference $M_1 \triangle M_2$ such that $|\Pi| \geq |M_2| - |M_1|$ and, for every path $p \in \Pi$, the following two conditions hold: (a) the first and last edges in $p$ belong to $M_2$, and (b) the first vertex in $p$ belongs to $A$ and the last vertex in $p$ belongs to $B$.*

**Proof:** The proof is basically the proof of [D00, Ex. 1, p. 40]. Decompose the symmetric difference $M_1 \triangle M_2$ into cycles and paths. Since $|M_1| \leq |M_2|$, there must be at least $|M_2| - |M_1|$ paths in $M_1 \triangle M_2$, where each such path has more edges in $M_2$ than edges in $M_1$. Let $p$ denote such a path. It follows that the first and last edges in $p$ belong to $M_2$. The parity of the number of edges in $p$ implies that its endpoints belong to different sides of $G$, and the claim follows. ∎