An Improved Algorithm for Packing T-Paths in Inner Eulerian Networks

Maxim A. Babenko*, Kamil Salikhov, and Stepan Artamonov

Moscow State University Yandex LLC
{maxim.babenko,salikhov.kamil}@gmail.com,
swatmad@list.ru

Abstract. A digraph $G = (V, E)$ with a distinguished set $T \subseteq V$ of *terminals* is called *inner Eulerian* if for each $v \in V - T$ the numbers of arcs entering and leaving v are equal. By a *T-path* we mean a simple directed path connecting distinct terminals with all intermediate nodes in $V - T$. This paper concerns the problem of finding a maximum T-path packing, i.e. a maximum collection of arc-disjoint T-paths.

A min-max relation for this problem was established by Lomonosov. The capacitated version was studied by Ibaraki, Karzanov, and Nagamochi, who came up with a strongly-polynomial algorithm of complexity $O(\phi(V, E) \cdot \log T + V^2 E)$ (hereinafter $\phi(n, m)$ denotes the complexity of a max-flow computation in a network with n nodes and m arcs).

For unit capacities, the latter algorithm takes $O(\phi(V, E) \cdot \log T + V E)$ time, which is unsatisfactory since a max-flow can be found in $o(V E)$ time. For this case, we present an improved method that runs in $O(\phi(V, E) \cdot \log T + E \log V)$ time. Thus plugging in the max-flow algorithm of Dinic, we reduce the overall complexity from $O(V E)$ to $O(\min(V^{2/3} E, E^{3/2}) \cdot \log T)$.

1 Preliminaries

1.1 Introduction

Computing a maximum integer flow, i.e. a maximum packing of paths connecting a given pair of terminals subject to edge capacities, is widely regarded as a central problem in combinatorial optimization. This problem has myriads of applications, both theoretical and practical.

Given a graph $G = (V, E)$ (either directed or undirected) and arbitrary integer capacities $e \colon E \to \mathbb{Z}_+$, one of the best strongly-polynomial max-flow algorithm [12] runs in $O(V E \log(V^2/E))$ time. (Hereinafter, in notation involving functions of numerical arguments or time bounds, we indicate sets for their cardinalities.) More efficient methods are known for the special case of unit capacities. The oldest one belongs to Dinic [3] and runs in $O(\min(E^{3/2}, V^{2/3} E))$ time (as shown independently by Karzanov [17] and Even and Tarjan [5]). Better results

* Supported by RFBR grants 10-01-93109 and 12-01-00864.

were recently discovered conerning max-flows in undirected unit-capacitated networks. Karger proposed an $M^*(V^{3/2}E^{2/3})$-time randomized algorithm [14,15], Goldberg and Rao gave an $O(V^{3/2}E^{1/2})$-time deterministic algorithm [11], and finally Karger and Levine presented a deterministic $O(V^{7/6}E^{2/3})$-time algorithm and a randomized $M^*(V^{20/9})$-time algorithm [16] (here $M^*(\cdot)$ denotes expected time with omitted polylogarithmic factors).

Multiflows are similar to usual flows but involve a set of terminals T that can be arbitrarily large. Most versions of the maximum integer multiflow problem are NP-hard. Still if the network is undirected and paths in a multiflow are allowed to connect arbitrary pairs of (distinct) terminals then the problem is tractable [24]. (This case is sometimes referred to as a *free multiflow.*) For directed networks one must additionally require capacities at all inner (non-terminal) nodes to be *balanced.* Considering this version of the problem, Lomonosov derived a simple min-max formula.

Ibaraki, Karzanov, and Nagamochi [13] applied a "divide and conquer" approach (originally introduced by Karzanov [19]) and devised a strongly-polynomial $O(\phi(V, E) \cdot \log T + V^2E)$-time algorithm for T-path packing problem in a capacitated digraph (V, E). The latter algorithm incorporates an arbitrary max-flow routine that runs in $\phi(n, m)$ time for a network with n nodes and m arcs. Since $\phi(V, E) = O(VE \log(V^2/E))$ (as, e.g., in the algorithm of Goldberg and Tarjan [12]), the term $O(V^2E)$ becomes a bottleneck. This issue was partially resolved in [1], where the total complexity was decreased to $O((\phi(V, E) + VE) \cdot \log T + VE \log(V^2/E))$.

The present paper concerns unit-capacitated networks. For this case, the algorithm from [13] takes $O(\phi(V, E) \cdot \log T + VE)$ time and since $\phi(V, E) = o(VE)$ the second term remains a bottleneck.

We introduce a novel *discrepancy scaling* technique and prove the following:

Theorem 1. *A maximum integer multiflow in an inner Eulerian digraph $G = (V, E)$ with terminals $T \subseteq V$ can be found in $O(\phi(V, E) \cdot \log T + E \log V)$ time.*

Our approach also extends to integer capacities as follows:

Theorem 2. *A maximum integer multiflow in an inner Eulerian digraph $G = (V, E)$ with terminals $T \subseteq V$ and integer capacities not exceeding C can be found in $O(\phi(V, E) \cdot \log T + E \log V \log T + E \log(V^2/E) \log(VC))$ time.*

The latter improves the bound $O(\phi(V, E) \log T + VE)$ from [1] provided that $\phi(n, m) = o(mn)$ (e.g. if the weakly-polynomial algorithm of Goldberg and Rao [11] is applied). In sense of capacity scaling, this improvement is ultimate since for all currently known scaling max-flow algorithms the first term is dominant.

1.2 Basic Notation and Facts

We shall use some standard definitions and notation. For a graph G, its set of nodes and edges (or arcs) are denoted by $V(G)$ and $E(G)$, respectively. A similar notation is used for paths and cycles.

Let G be a digraph and X be a subset of nodes. Then $G[X]$ denotes the subgraph of G induced by X. Also $\delta_G^{\text{in}}(X)$ (respectively $\delta_G^{\text{out}}(X)$) denotes the set of arcs entering X (respectively leaving X) and $\delta_G(X) := \delta_G^{\text{in}}(X) \cup \delta_G^{\text{out}}(X)$. When G is clear from the context, it is omitted. Also for $X = \{v\}$ we write just $\delta^{\text{in}}(v)$ and $\delta^{\text{out}}(v)$.

For an arbitrary real-valued function $h\colon U \to \mathbb{R}$ and $U' \subseteq U$ we write $h(U')$ to denote $\sum_{u \in U'} h(u)$.

A digraph $G = (V, E)$ with a distinguished subset $T \subseteq V$ is said to be *inner Eulerian* if for each $v \in V - T$ the in-degree and out-degree of v are equal. Nodes in T and in $V - T$ are called *terminals* and *inner* nodes, respectively. A simple path in G is called a *T-path* if its endpoints are distinct terminals and the other nodes are inner.

By a network we mean a triple $N = (G, T, c)$ consisting of a digraph $G = (V, E)$, a set of terminals T, and a non-negative function $c\colon E \to \mathbb{Z}_+$ of arc *capacities*. The notion of inner Eulerianess is extended to N in a natural way, namely $c(\delta^{\text{in}}(v)) = c(\delta^{\text{out}}(v))$ should hold for all $v \in V - T$. In case of unit capacities we omit c from notation and deal with the pair $N = (G, T)$.

A collection $\mathcal{P}$ consisting of T-paths P_i endowed with real weights $\alpha_i \in \mathbb{R}_+$ such that

$$\sum (\alpha_i : e \in E(P_i)) \le c(e) \qquad \text{for all } e \in E \tag{1}$$

is called a *free multiflow* or a *T-path packing* (the adjective “free” is used to emphasize that any pair of distinct terminals is allowed to be connected, i.e., the commodity graph in the corresponding multiflow problem is complete). The *value* of $\mathcal{P}$ is the sum

$$\text{val}(\mathcal{P}) := \alpha_1 + ... + \alpha_k$$

and $\mathcal{P}$ is called maximum if $\text{val}(\mathcal{P})$ is maximum.

If all numbers α_i are integers then $\mathcal{P}$ is called *integer*. Integer multiflows can be viewed as multisets of T-paths. If $c(e) = 1$ for all $e \in E$ then T-paths forming an integer multiflow are arc-disjoint. This case will be our primary focus.

The maximum integer multiflow problem in inner Eulerian networks admits a min-max relation, which is due to Lomonosov. (See also [18, Sec. 4], [8] and [25, p. 1289].) For $t \in T$, call $X \subset V$ a *t-cut* if $X \cap T = \{t\}$. Denote by $\lambda^{\text{out}}(t)$ the minimum of $c(\delta^{\text{out}}(X))$ over all t-cuts X.

Theorem 3 (Lomonosov (unpublished, 1978), Frank [8]). *For $N = (G, T, c)$ as above, there exists a maximum integer directed multiflow $\mathcal{P}$ in N with*

$$\text{val}(\mathcal{P}) = \sum_{t \in T} \lambda^{\text{out}}(t).$$

For $t \in T$, let X_t be a t-cut with $c(\delta^{\text{out}}(X_t))$ minimum. Note that inner Eulerianness of N implies

$$c(\delta^{\text{out}}(X)) - c(\delta^{\text{in}}(X)) = c(\delta^{\text{out}}(t)) - c(\delta^{\text{in}}(t)) \quad \text{for each } t\text{-cut } X. \tag{2}$$

Thus the minima of $c(\delta^{\text{out}}(X_t))$ and $c(\delta^{\text{in}}(X_t))$ are obtained simultaneously. Also note that

$$\sum_{t\in T}\big(c(\delta^{\text{out}}(t)) - c(\delta^{\text{in}}(t))\big) = 0$$

and hence

$$\sum_{t\in T} c(\delta^{\text{out}}(X_t)) = \sum_{t\in T} c(\delta^{\text{in}}(X_t)).$$

Therefore every optimal $\mathcal{P}$ saturates the cuts $\delta^{\text{out}}(X_t)$ and $\delta^{\text{in}}(X_t)$ for $t \in T$.

We shall also rely on analogous facts concerning *undirected* networks. A network $N = (G, T, c)$ consisting of an undirected graph $G = (V, E)$, a set of terminals $T \subseteq V$, and integer capacities $c\colon E \to \mathbb{Z}_+$ is called *inner Eulerian* if $c(\delta(t))$ is even for all $t \in V - T$. The notions of T-paths, t-cuts ($t \in T$), and free multiflows in N are defined similar to the directed case. For $t \in T$, let $\lambda(t)$ be the capacity of a minimum t-cut X_t. Lovász and Cherkassky, independently, established the following min-max relation:

Theorem 4 (Lovász [22], Cherkassky [2]). *For $N = (G, T, c)$ as above, there exists a maximum integer undirected multiflow $\mathcal{P}$ in N with*

$$\operatorname{val}(\mathcal{P}) = \frac{1}{2}\sum_{t\in T}\lambda(t).$$

As above, every optimal multiflow $\mathcal{P}$ saturates cuts $\delta(X_t)$ for $t \in T$.

2 Algorithm

2.1 Outline

Modern efficient methods for computing maximum free multiflows rely on "divide and conquer" approach, which was originally applied in [19] to find, in $O(\phi(V,E)\cdot\log T)$ time, a *half-integer* maximum multiflow in a undirected graph $G = (V, E)$ with integer edge capacities. Subsequently, this method was improved and extended in [13] so as to find an integer maximum free multiflow in an inner Eulerian undirected network in the same time $O(\phi(V,E)\cdot\log T)$, and in an inner Eulerian directed network in $O(\phi(V,E)\cdot\log T + V^2E)$ time.

Consider a directed unit-capacitated network $N = (G, T)$ with $|T| \geq 4$. The problem for N is recursively reduced to problems in two networks $N' = (G', T')$ and $N'' = (G'', T'')$ such that $|T'|, |T''| \leq \lceil |T|/2 \rceil + 1$. First, fix an arbitrarily partition $\{S', S''\}$ of T into parts of almost equal sizes. Second, compute an S'-cut X' (i.e. a subset $X' \subset V(G)$ with $X' \cap T = S'$) of minimum capacity $c(\delta^{\text{out}}(X'))$ by running a max-flow routine and considering nodes S' as sources and nodes S'' as sinks. Form graph G' by contracting X' in G into a new *composite* node (denoted by t'). We identify arcs in graphs resulting from contractions with their pre-images in original graphs. Define the set of terminals in G' as $T' := \{t'\} \cup S''$. Similarly $N'' = (G'', T'')$ is constructed by contracting $X'' := V(G) - X'$ in G into a new node t'' and setting $T'' := \{t''\} \cup S'$.

Remark 1. The complexity of this *separation* procedure is clearly dominated by the min-cut computation. Note that in view of (2) (which holds for S'-cuts as well) the requested minimum cut can be found by computing a max-flow in the unit-capacitated underlying *undirected* graph (obtained from G by dropping arc directions). This enables to use faster max-flows algorithms designed to handle undirected graphs (e.g. [16]).

Next, the algorithm proceeds to N' and N'' recursively and computes optimal solutions $\mathcal{P}'$ and $\mathcal{P}''$ to N' and N'', respectively. Finally, the *aggregation* procedure combines $\mathcal{P}'$ and $\mathcal{P}''$ into a maximum integer multiflow $\mathcal{P}$ in N as follows. We shall assume that $\mathcal{P}'$ and $\mathcal{P}''$ are given explicitly, i.e. as collections of arc-disjoint T'- and T''-paths, respectively. Define $\mathcal{P}'_0 := \{P \in \mathcal{P}' \mid t' \text{ is not an endpoint of } P\}$ and, symmetrically, $\mathcal{P}''_0 := \{P \in \mathcal{P}'' \mid t'' \text{ is not an endpoint of } P\}$. By the maximality of $\mathcal{P}'$ and $\mathcal{P}''$ and the minimality of X', paths in $\mathcal{P}' - \mathcal{P}'_0$ saturate the cut $\delta_{G'}(t')$ and, symmetrically, paths in $\mathcal{P}'' - \mathcal{P}''_0$ saturate the cut $\delta_{G''}(t'')$. This enables to recombine these paths into a collection $\mathcal{P}_1$ of T-paths (connecting terminals in S' with terminals in S''). The final packing $\mathcal{P}$ in N is defined as $\mathcal{P} := \mathcal{P}'_0 \cup \mathcal{P}''_0 \cup \mathcal{P}_1$. For the proof of maximality of $\mathcal{P}$ and a more detailed exposition, see [13]. The aggregation procedure runs in $O(V + E)$ time, which, compared to the separation phase, is negligible.

For $|T| = 3$, the above method is inapplicable (since it yields $|T'| = |T''| = 3$) so this basic case is handled separately, as explained below in Subsection 2.3.

Let $T(n, m, k)$ denote the complexity of the algorithm in a network N with n nodes, m arcs, and k terminals. Then for $k \geq 4$

$$T(n, m, k) = T(n', m', k') + T(n'', m'', k'') + \phi(n, m) + O(m + n), \qquad (3)$$

where (n', m', k') and (n'', m'', k'') denote analogous size parameters for networks N' and N'', respectively, and $\phi(n, m)$ is the complexity of a max-flow routine in a network with n nodes and m arcs. (As indicated earlier, the latter routine can be assumed to deal with an undirected unit-capacitated graph.) Also, as we shall show in Subsection 2.3, for $k \leq 3$

$$T(n, m, k) = O(\phi(n, m) + m \log n). \qquad (4)$$

Assuming that $\phi(n, m)$ is "reasonable" one can solve (3) and (4) as follows (see [13] for a detailed proof):

$$T(n, m, k) = O(\phi(n, m) \log k + m \log n).$$

It remains to show how to solve the problem for a directed unit-capacitated network $N = (G, T)$ with $|T| \leq 3$ terminals in $O(\phi(V, E) + E \log V)$ time. For this we shall need some terminology and basic facts concerning *skew-symmetric graphs* (which were earlier introduced as a convenient tool for solving flow and matching problems; see [10,1] for a survey).

2.2 Skew-Symmetric Graphs

A *skew-symmetric graph* is a digraph $G = (V, E)$ endowed with two bijections σ_V, σ_E such that: σ_V is an involution on the nodes (i.e., $\sigma_V(v) \neq v$ and $\sigma_V(\sigma_V(v)) = v$ for each $v \in V$), σ_E is an involution on the arcs, and for each arc e from u to v, $\sigma_E(e)$ is an arc from $\sigma_V(v)$ to $\sigma_V(u)$. For brevity, we combine the mappings σ_V, σ_E into one mapping σ on $V \cup E$ and call σ the *symmetry* (rather than skew-symmetry) of G. For a node (arc) x, its symmetric node (arc) $\sigma(x)$ is also called the *mate* of x, and we will often use notation with primes for mates, denoting $\sigma(x)$ by x'. Obviously $\delta^{\text{in}}(v)' = \delta^{\text{out}}(v')$ and $\deg^{\text{in}}(v) = \deg^{\text{out}}(v')$ for each $v \in V$.

We admit parallel arcs, but not loops in G. Observe that if G contains an arc e from a node v to its mate v', then e' is also an arc from v to v' (so the number of arcs of G from v to v' is even and these parallel arcs are partitioned into pairs of mates).

The symmetry σ is extended in a natural way to paths, circuits, subsets etc. Namely, two paths are symmetric to each other if the elements of one of them are symmetric to those of the other and go in the reverse order: for a path $P = (v_0, e_1, v_1, \ldots, e_k, v_k)$, the symmetric path P' is $(v'_k, e'_k, v'_{k-1}, \ldots, e'_1, v'_0)$.

2.3 Case $|T| \leq 3$

Consider a directed unit-capacitated network $N = (G = (V, E), T)$ with $|T| \leq 3$. Adding an isolated terminal (if needed) one may assume that $|T| = 3$. The best known algorithm that finds a maximum free multiflow in N runs in $O(\phi(V, E) + VE)$ time [13] . We shall improve this to $O(\phi(V, E) + E \log V)$ as follows.

Stage 1: Define $T = \{t_1, t_2, t_3\}$ and let $\overline{G}$ be the underlying undirected graph of G. Apply the algorithm for inner Eulerian undirected graphs from [13, Sec.2.1] to find a maximum integer free multiflow $\overline{\mathcal{P}}$ in $\overline{N} = (\overline{G}, T)$ (endowed with unit capacities). This takes $O(\phi(V, E))$ time.

Remark 2. The latter algorithm involves computing two max-flows, say g_1 and g_2, where only g_1 is undirected. Namely, g_1 is a max-flow in $\overline{G}$ from t_1 to $\{t_2, t_3\}$, and g_2 is a max-flow from t_2 to t_3 in the *residual network* w.r.t. g_1. Hence the undirected max-flow algorithm of Karger [16] may seem inapplicable here. Fortunately, Karger's algorithm also works if a small number of edges is directed (which is the case for the residual network since g_1 is acyclic and hence uses few edges, see [16]).

Stage 2: For reasons that will soon become clear, we need $\overline{\mathcal{P}}$ to saturate all edges of $\overline{G}$. Let E_0 be the set of edges in $\overline{G}$ that are not used by paths in $\overline{\mathcal{P}}$.

Lemma 1. $\deg_{\overline{G}_0}(v)$ *is even for all* $v \in V$.

Proof. This is clear for $v \in V - T$ since $\deg_{\overline{G}}(v)$ is even, paths in $\overline{\mathcal{P}}$ are edge-disjoint, and every path in $\overline{\mathcal{P}}$ uses an even number of edges incident to v. Consider

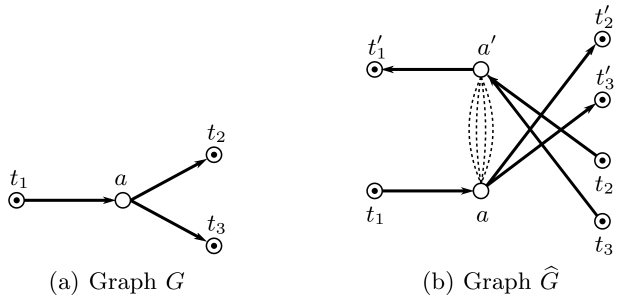

Fig. 1. Graph G with terminals $\{t_1, t_2, t_3\}$ and the corresponding graph $\widehat{G}$ (directions of auxiliarly arcs are not shown)

$v \in T$. Since $\overline{\mathcal{P}}$ is maximum, by Theorem 4 there exist t_i-cuts X_i such that $\overline{\mathcal{P}}$ saturates $\delta_{\overline{G}}(X_i)$ $(i = 1, 2, 3)$. Hence in $\overline{G}_0[X_i]$ at most one node, namely t_i, can have odd degree. In every undirected graph the number of nodes of odd degree is even, therefore $\deg_{\overline{G}_0}(v)$ is also even, as claimed. □

By Lemma 1 the algorithm can decompose E_0 into a collection of undirected circuits and attach these circuits to arbitrary paths in $\overline{\mathcal{P}}$. This takes $O(V + E)$ time and ensures that $\overline{\mathcal{P}}$ covers all edges of $\overline{G}$, as desired.

Stage 3: Construct an auxiliary skew-symmetric graph as follows. First take a disjoint symmetric copy $V' := \{v' \mid v \in V\}$ of V. For each arc $(u, v) \in E$ add two symmetric *regular* arcs (u, v) and (v', u'). Adjust the endpoints of regular arcs to ensure that each regular arc incident to t $(t \in T)$ leaves t and, symmetrically, each regular arc incident to t' enters t'. To this aim, replace every arc (x, t), $x \in V \cup V'$, by (x, t') and, symmetrically, replace every arc (t', x), $x \in V \cup V'$, by (t, x). Finally, for each $v \in V - T$ add four *auxiliary* arcs: two (symmetric) arcs (v, v') and two (also symmetric) arcs (v', v). Denote the resulting skew-symmetric graph by $\widehat{G}$. An example is depicted on Fig. 1.

Remark 3. The need for adding *two* auxiliary arcs (v, v') instead of just one is dictated by the definition of skew-symmetric graphs and is thus purely technical.

Stage 4: Multiflow $\overline{\mathcal{P}}$ in $\overline{G}$ gives rise to a certain weighted collection $\widehat{\mathcal{P}}$ of directed paths in $\widehat{G}$. Consider a path $\overline{P} \in \overline{\mathcal{P}}$ from, say, t_i to t_j $(i \neq j)$:

$$\overline{P} = (t_i = v_0, e_1, v_1, \ldots, e_l, v_l = t_j),$$

where $v_i \in V(G)$ $(i = 0, \ldots, l)$, $e_i \in E(\overline{G})$, e_i connects nodes v_{i-1} and v_i $(i = 1, \ldots, l)$.

Since edge capacities are 1 and $\overline{\mathcal{P}}$ is integer, $\overline{P}$ has weight 1. We transform $\overline{P}$ into a directed path $\widehat{P}$ (also of weight 1) in $\widehat{G}$ by taking appropriate regular arcs (corresponding to edges e_i) and inserting auxiliary arcs where needed. More formally, for each edge $e_i = \{v_{i-1}, v_i\}$, $\widehat{G}$ contains a unique regular arc $a_i =$

(x, y), where $x = v_{i-1}$ or $x = v'_{i-1}$ and $y = v_i$ or $y = v'_i$. Consider the sequence $(a_1, \ldots, a_l)$ and turn it into a directed path $\widehat{P}$ by inserting auxiliary arcs (x, x'), $x \in V \cup V'$, between a_{i-1} and a_i if a_{i-1} ends at x and a_i starts at x'. Then $\widehat{\mathcal{P}}$ consists of paths $\widehat{P}$ and their symmetric mates $\widehat{P}'$.

Note the following:

(5) (i) $\widehat{\mathcal{P}}$ is symmetric, i.e. $\widehat{P} \in \widehat{\mathcal{P}}$ implies $\widehat{P}' \in \widehat{\mathcal{P}}$;
(ii) Each regular arc belongs to at most one path in $\widehat{\mathcal{P}}$;
(iii) Each path $P \in \widehat{\mathcal{P}}$ connects a node t_i with a node t'_j, where $i \neq j$.

A collection $\widehat{\mathcal{P}}$ of directed paths in $\widehat{G}$ obeying (5) will be referred to as a *integer skew-symmetric multiflow*. Suppose the following property holds for $\widehat{\mathcal{P}}$:

(6) Paths in $\widehat{\mathcal{P}}$ do not contain auxiliary arcs.

Then such $\widehat{\mathcal{P}}$ induces an integer multiflow $\mathcal{P}$ in G obeying $\mathrm{val}(\mathcal{P}) = \frac{1}{2}\,\mathrm{val}(\widehat{\mathcal{P}})$. Indeed, consider pairs of symmetric paths $\{\widehat{P}, \widehat{P}'\}$ forming $\widehat{\mathcal{P}}$ and let us show that each such pair corresponds to a T-path in G and these T-paths are arc-disjoint. Let us say that arcs e', e'' form a *transit pair* if e', e'' have a common endpoint v, one of e', e'' enters x, and the other leaves v. Consider $\widehat{P}$ as a sequence of arcs $\widehat{\tau} = (\widehat{e}_1, \ldots, \widehat{e}_l)$. For each $i = 1, \ldots, l-1$, arcs $\widehat{e}_i, \widehat{e}_{i+1}$ in $\widehat{G}$ form a transit pair and thus their pre-images e_i, e_{i+1} in G also form a transit pair (as it follows from the construction of $\widehat{G}$). Therefore either the sequence of pre-images $\tau = (e_1, \ldots, e_l)$ or the reverse one $\tau^{-1} = (e_l, \ldots, e_1)$ gives rise to a directed T-path in G. These paths are arc-disjoint by (5)(ii).

Stage 5: At this final stage we rearrange paths in $\widehat{\mathcal{P}}$ while maintaining (5) and preserving $\mathrm{val}(\widehat{\mathcal{P}})$ to get rid of auxiliary arcs. A similar subtask was earlier addressed in [1]. The latter algorithm scans nodes $v \in V - T$ one by one and removes auxiliary arcs between v and v'. Handling each node involves computing certain flow decompositions and takes $O(V+E)$ time (assuming unit capacities), which gives $O(VE)$ in total. We choose a different way of dealing with auxiliary arcs. Instead of processing inner nodes one at a time we apply certain global transformations aimed to decrease the total flow on auxiliary arcs.

For $1 \leq i, j \leq 3$, $i \neq j$, combine all t_i–t'_j paths in $\widehat{\mathcal{P}}$ and form an integer t_i–t'_j flow f_{ij} (see Fig. 2). For a function h on $E(\widehat{G})$, define $h'(e) := h(e')$. The symmetry of $\widehat{\mathcal{P}}$ implies $f'_{ij} = f_{ji}$ for all $1 \leq i, j \leq 3$, $i \neq j$. This property will be maintained throughout the iterations.

For each $v \in V - T$, define

$$\alpha(v) := f_{1,2}[v, v'], \quad \beta(v) := f_{1,3}[v, v'], \quad \gamma(v) := f_{2,3}[v, v'],$$

where $f_{ij}[v, v']$ is the *flow between* v *and* v', i.e. the sum of f_{ij}-values on auxiliary arcs (v, v') minus the sum of f_{ij}-values on auxiliary arcs (v', v).

Lemma 2. $\alpha(v) + \beta(v) + \gamma(v) = 0$ *for all* $v \in V - T$.

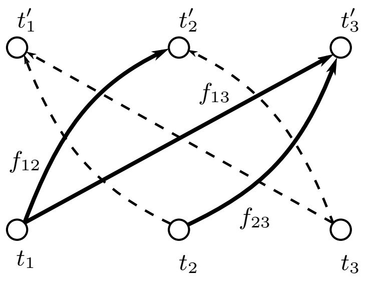

Fig. 2. Network $\widehat{G}$ and flows f_{12}, f_{13}, and f_{23}. Symmetric flows f_{21}, f_{31}, and f_{32} are shown by dashed lines.

Proof. Consider a node $v \in V - T$. Since G is inner balanced, $\deg_G^{\text{in}}(v) = \deg_G^{\text{out}}(v)$. From the construction of $\widehat{G}$ it follows that v has equal numbers of incoming and outcoming regular arcs. Define $g := \sum(f_{ij} : 1 \le i,j \le 3, i \ne j)$. Recall that $\overline{\mathcal{P}}$ saturates all edges of $\overline{G}$. Hence g saturates all regular arcs of $\widehat{G}$ and thus the total flow on incoming regular arcs equals the total flow on outcoming regular arcs. Therefore $g[v, v'] = 0$ and thus $\alpha(v) + \beta(v) + \gamma(v) = 0$. $\square$

Define the *discrepancy* at v by $\Delta(v) := |\alpha(v)| + |\beta(v)| + |\gamma(v)|$ and the *total discrepancy* by $\Delta := \sum(\Delta(v) : v \in V - T)$. Note that 2Δ is exactly the sum of flows f_{ij}, $1 \le i,j \le 3$, $i \ne j$, on all auxiliary arcs (factor 2 comes from the symmetry) and thus (6) is equivalent to $\Delta = 0$.

The algorithm executes a series of scaling phases. Each phase decreases Δ by at least a factor of $\frac{11}{12}$. Since Δ is integer and initially $\Delta \le E/2 = O(V^2)$ (recall that G has unit capacities and each unit of discrepancy corresponds to an inner node of a T-path in $\overline{\mathcal{P}}$) it follows that $O(\log V)$ scaling phases are sufficient to achieve $\Delta = 0$. Flows f_{ij} are finally decomposed into an integer skew-symmetric multiflow $\widehat{\mathcal{P}}$ obeying (6).

A phase works as follows. Call a node $v \in V - T$ *active* if $|\alpha(v)| \ge |\beta(v)| \ge |\gamma(v)|$. By permuting terminals t_1, t_2, and t_3 (and thus values α, β, and γ) one can assume w.l.o.g. that

$$\sum(\Delta(v) : v \textit{ is active}) \ge \frac{1}{6}\Delta. \tag{7}$$

To decrease Δ, define $h := f_{12} + f_{13}$ and cancel flows on oppositely directed auxiliary arcs. Note that h is an integer t_1–$\{t'_2, t'_3\}$ flow in $\widehat{G}$ and is thus decomposable into a sum of a t_1–t'_2 flow h_{12} and a t_1–t'_3 flow h_{13}. Computing h and decomposing it into h_{12} and h_{13} takes $O(V + E)$ time. The algorithm resets $(f_{12}, f_{21}, f_{13}, f_{31}) := (h_{12}, h'_{12}, h_{13}, h'_{13})$ and then proceeds to the next phase.

To estimate the decrease of Δ on each phase consider an arbitrary node $v \in V - T$ and let $\alpha'(v)$, $\beta'(v)$, and $\gamma'(v)$ $(= \gamma(v))$ be the corresponding values for the updated triple flows f_{ij}. Clearly $|\alpha'(v)| + |\beta'(v)| + |\gamma'(v)| \le |\alpha(v)| + |\beta(v)| + |\gamma(v)|$. The update also maintains the property given in Lemma 2.

Now suppose that v is active. Note that $\alpha(v) + \beta(v) = \alpha'(v) + \beta'(v)$. Also $\alpha(v)$ and $\beta(v)$ are of different signs (since $\alpha(v)$, $\beta(v)$, and $\gamma(v)$ add up to zero, and $\alpha(v)$ has the largest magnitude) while $\alpha'(v)$ and $\beta'(v)$ are of the same sign (since h_{12} and h_{13} come from a decomposition of h and thus use auxiliary arcs of the same direction). Therefore

$$\begin{aligned}&\big(|\alpha(v)| + |\beta(v)| + |\gamma(v)|\big) - \big(|\alpha'(v)| + |\beta'(v)| + |\gamma'(v)|\big) =\\ &|\alpha(v) - \beta(v)| + |\alpha'(v) + \beta'(v)| =\\ &|\alpha(v) - \beta(v)| + |\alpha(v) + \beta(v)| =\\ &2\,|\beta(v)|\,.\end{aligned}$$

Since $|\beta(v)| \geq \frac{1}{2}\,|\alpha(v)| \geq \frac{1}{4}\Delta(v)$ (which follows from $\alpha(v) + \beta(v) + \gamma(v) = 0$ and $|\alpha(v)| \geq |\beta(v)| \geq |\gamma(v)|$) the total discrepancy decreases by at least

$$\sum \left(\frac{1}{2}\Delta(v) : v \textit{ is active}\right) \geq \frac{1}{12}\Delta.$$

This concludes the proof of Theorem 1. □

3 General Capacities

The above approach also extends to the case when arc capacities are integers in range $[0, C]$ and leads to a weakly-polynomial $o(VE)$-time algorithm, as claimed in Theorem 2. The detailed proof is rather technical so we shall only give a brief sketch here.

During the course of the algorithm we maintain each multiflow $\mathcal{P}$ as a collection $\{(A_1, B_1, f_1), \ldots, (A_q, B_q, f_q)\}$, where for $i = 1, \ldots, q$, $A_i \cap B_i = \emptyset$, $A_i, B_i, \subset T$, and f_i is an acyclic integer A_i–B_i flow. Flows f_i are kept in a compact form, i.e. as lists $\{(e, f_i(e)) \mid f_i(e) \neq 0\}$. As in [13], such a representation of $\mathcal{P}$ additionally obeys $q = O(\log T)$ and occupies $O(E \log T)$ space. To merge a pair of such representations on each recursion level we solve $O(\log T)$ flow decomposition problems (each taking linear time due to acyclicity), extract flows corresponding to $\mathcal{P}' - \mathcal{P}'_0$ and $\mathcal{P}'' - \mathcal{P}''_0$, merge these flows, and finally decycle the result (which takes $O(m \log n)$ time [26]). Totally all aggregations take $O(E \log V \log T)$ time. Separations require, as earlier, $O(\phi(V, E) \cdot \log T)$ time.

It remains to deal with *leaf* subproblems (those corresponding to $|T| \leq 3$). Fix a leaf subproblem with n nodes and m arcs. Computing a maximum multiflow $\overline{\mathcal{P}}$ in the underlying undirected network $\overline{N}$ takes $O(\phi(n, m))$ time; the resulting $\overline{\mathcal{P}}$ is then turned into a collection of three integer flows $\overline{f}_{ij}$, $1 \leq i < j \leq 3$, in $O(m \log(n^2/m))$ time with the help of the fast flow decomposition routine [1]. Residual edge capacities (those corresponding to slacks in (1)) are Eulerian (cf. Lemma 1). Similar to the unit-capacitated case one can compute (in linear time) a weighted collection of additional cycles (not necessarily simple) that exhaust the residual capacities and attach these cycles to an arbitrary component of $\overline{\mathcal{P}}$. Each $\overline{f}_{ij}$ is turned into a pair of symmetric integer t_i–t'_j and t_j–t'_i flows f_{ij} and

f_{ji} in $\widehat{G}$. The latter does not require explicit path-packing representations of $\overline{f}_{ij}$ and can be done in linear time. The algorithm from Subsection 2.3 is applied to get rid of flows on auxiliary arcs; this requires $O(\log(n\,\mathrm{val}(\overline{\mathcal{P}}))) = O(\log(nC))$ scaling phases. The resulting flows are finally decycled in $O(m \log n)$ time [26].

From the above estimates it follows that for a network $N = (G = (V, E), T, c)$ the new algorithm takes $O(\phi(V, E) \cdot \log T + E \log V \log T + E \log(V^2/E) \log(VC))$ time. This concludes the proof of Theorem 2. □

Acknowledgements. The authors are thankful to anonymous referees for helpful comments and suggestions.

References

1. Babenko, M.A., Karzanov, A.V.: Free multiflows in bidirected and skew-symmetric graphs. Discrete Applied Mathematics 155, 1715–1730 (2007)
2. Cherkassky, B.V.: A solution of a problem on multicommodity flows in a network. Ekonomika i Matematicheskie Metody 13(1), 143–151 (1977) (in Russian)
3. Dinic, E.A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. Dokl. Akad. Nauk. SSSR 194, 754–757 (in Russian) (translated in Soviet Math. Dokl. 111, 277–279)
4. Edmonds, J., Johnson, E.L.: Matching: a well-solved class of integer linear programs. In: Guy, R., Hanani, H., Sauer, N., Schönhein, J. (eds.) Combinatorial Structures and Their Applications, pp. 89–92. Gordon and Breach, NY (1970)
5. Even, S., Tarjan, R.E.: Network Flow and Testing Graph Connectivity. SIAM Journal on Computing 4, 507–518 (1975)
6. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton Univ. Press, Princeton (1962)
7. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Sci. 10, 111–121 (1980)
8. Frank, A.: On connectivity properties of Eulerian digraphs. Ann. Discrete Math. 41, 179–194 (1989)
9. Goldberg, A.V., Karzanov, A.V.: Path problems in skew-symmetric graphs. Combinatorica 16, 129–174 (1996)
10. Goldberg, A.V., Karzanov, A.V.: Maximum skew-symmetric flows and matchings. Mathematical Programming 100(3), 537–568 (2004)
11. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. In: Proc. 38th IEEE Symposium Foundations of Computer Science (1997); adn Journal of the ACM 45, 783–797 (1998)
12. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum flow problem. J. ACM 35, 921–940 (1988)
13. Ibaraki, T., Karzanov, A.V., Nagamochi, H.: A fast algorithm for finding a maximum free multiflow in an inner Eulerian network and some generalizations. Combinatorica 18(1), 61–83 (1998)
14. Karger, D.R.: Random sampling in cut, flow, and network design problems. Mathematics of Operations Research (1998)
15. Karger, D.R.: Better random sampling algorithms for flows in undirected graphs. In: Proc. 9th Annual ACM+SIAM Symposium on Discrete Algorithms, pp. 490–499 (1998)

16. Karger, D.R., Levine, M.S.: Finding Maximum flows in undirected graphs seems easier than bipartite matching. In: Proc. 30th Annual ACM Symposium on Theory of Computing, pp. 69–78 (1997)
17. Karzanov, A.V.: O nakhozhdenii maksimalnogo potoka v setyakh spetsialnogo vida i nekotorykh prilozheniyakh. In: Matematicheskie Voprosy Upravleniya Proizvodstvom, vol. 5. University Press (1973) (in Russian)
18. Karzanov, A.V.: Combinatorial methods to solve cut-dependent problems on multiflows. In: Combinatorial Methods for Flow Problems, Inst. for System Studies, Moscow, vol. (3), pp. 6–69 (1979) (in Russian)
19. Karzanov, A.V.: Fast algorithms for solving two known problems on undirected multicommodity flows. In: Combinatorial Methods for Flow Problems, Inst. for System Studies, Moscow, vol. (3), pp. 96–103 (1979) (in Russian)
20. Kupershtokh, V.L.: A generalization of Ford-Fulkerson theorem to multiterminal networks. Kibernetika 7(3), 87–93 (1971) (in Russian) (Translated in Cybernetics 7(3) 494-502)
21. Lawler, E.L.: Combinatorial Optimization: Networks and Matroids. Holt, Reinhart, and Winston, NY (1976)
22. Lovász, L.: On some connectivity properties of Eulerian graphs. Acta Math. Akad. Sci. Hung. 28, 129–138 (1976)
23. Lovász, L.: Matroid matching and some applications. J. Combinatorial Theory, Ser. B 28, 208–236 (1980)
24. Mader, W.: Über die Maximalzahl kantendisjunkter A-Wege. Archiv der Mathematik (Basel) 30, 325–336 (1978)
25. Schrijver, A.: Combinatorial Optimization. Springer (2003)
26. Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. J. Comput. Syst. Sci. 26(3), 362–391 (1983)
27. Tutte, W.T.: Antisymmetrical digraphs. Canadian J. Math. 19, 1101–1117 (1967)