Technical Report #97002

A Fast Algorithm for Finding a Maximum Free Multiflow in an Inner Eulerian Network and Some Generalizations

Toshihide Ibaraki, Alexander V. Karzanov, and Hiroshi Nagamochi

January 1997

Graduate School of Engineering
Department of Applied Mathematics and Physics
Kyoto University, Kyoto, Japan 606-01
Netaddress: naga@kuamp.kyoto-u.ac.jp

Alexander V. Karzanov
Institute for System Analysis of RAS,
9, Prospect 60 Let Oktyabrya,
117312 Moscow, Russia.

Toshihide Ibaraki *and* Hiroshi Nagamochi
Department of Applied Mathematics and Physics
Graduate School of Engineering
Kyoto University
Kyoto, Japan 606-01

A Fast Algorithm for Finding a Maximum Free Multiflow in an Inner Eulerian Network and Some Generalizations

Toshihide Ibaraki[♯], Alexander V. Karzanov[♭], and Hiroshi Nagamochi[♯*]

Abstract. Let $N = (G, T, c)$ be a network, where G is an undirected graph with n nodes and m edges, T is a set of p specified nodes of G, called *terminals*, and each edge e of G has a nonnegative integer capacity $c(e)$. If the total capacity of edges with one end at v is even for every non-terminal node v, then N is called *inner Eulerian*. A free multiflow is a collection of flows between arbitrary pairs of terminals such that the total flow through each edge does not exceed its capacity.

In this paper we first generalize a method in Karzanov [1979a] to find a maximum integer free multiflow in an inner Eulerian network, in $O(\varphi(n, m)\log p)$ time, where φ is the complexity of finding a maximum flow between two terminals. Next we extend our algorithm to solve the so-called *laminar locking problem* on multiflows, also in $O(\varphi(n, m)\log p)$ time.

We then consider analogs of the above problems in *inner balanced directed* networks, which means that for each non-terminal node v, the sums of capacities of arcs entering v and leaving v are the same. We show that for such a network a maximum integer free multiflow can be constructed in $O(\varphi(n, m)\log p + n^2 m)$ time, and then extend this result to the corresponding locking problem.

Keywords: Maximum flow, Multi(commodity)flow, Multiflow locking problem, Eulerian graph.

1. Introduction

By a *network* we mean a triple $N = (G, T, c)$ consisting of an undirected graph $G = (V_G, E_G)$, a subset T of nodes of G, called *terminals*, and a nonnegative integer-valued function $c : E_G \to \mathbb{Z}_+$ of *capacities* of edges of G. We denote $n = |V_G|$, $m = |E_G|$ and $p = |T|$ throughout this paper. A *T-path* is a simple path in G which connects two

♯ Department of Applied Mathematics and Physics, Graduate School of Engineering, Kyoto University, Kyoto 606, Japan.

♭ Institute for System Analysis of RAS, 9, Prospect 60 Let Oktyabrya, 117312 Moscow, Russia. This research was done when the second author was visiting Kyoto University and supported by the Scientific Grant in Aid by the Ministry of Education, Science and Culture of Japan (Joint Research 06044112).

* Supported by the subsidy from the Inamori Foundation.

distinct terminals. A *multiflow* F consists of T-paths $P_1, \ldots, P_k$ along with nonnegative reals $\alpha_1, \ldots, \alpha_k$, and F is called *c-admissible* if it obeys the capacity constraint

$$\zeta^F(e) := \sum(\alpha_i \,:\, e \in P_i) \leq c(e) \quad \text{for all } e \in E_G. \tag{1}$$

Following Lomonosov (1985), such a multiflow is called *free* (because no pair of terminals is forbidden to use as end nodes of paths in F). Unless otherwise stated, we assume that any multiflow we deal with is c-admissible and free. The (total) *value*, $\mathrm{val}(F)$, of F is $\alpha_1 + \ldots + \alpha_k$, and F is called *maximum* if its value is as large as possible. For disjoint subsets $A, B \subseteq T$, the subset of F concerning all the paths with one end in A and the other in B forms the *flow* in F *between* A and B, and is denoted by $F_{A,B}$ or $F_{B,A}$. Also F_A stands for $F_{A,T-A}$. A single element set $\{s\}$ is sometimes denoted by s; e.g., F_s stands for $F_{\{s\}}$, and $A \cup s$ stands for $A \cup \{s\}$.

For $X \subseteq V$, $\delta(X) = \delta_G(X)$ is the set of edges with one end in X and the other in $V - X$ (a *cut* in G). For a function $g : S \to \mathbb{R}$ and a subset $S' \subseteq S$, $g(S')$ denotes $\sum(g(e) \,:\, e \in S')$; so $c(\delta(X))$ is the capacity of a cut $\delta(X)$. For $A \subseteq T$, $\lambda_A = \lambda_A(c)$ is the *minimum capacity* of a cut $\delta(X)$ *separating* A and $T - A$, i.e., with $X \cap T = A$ or $T - A$. Because of (1), the value of F_A does not exceed the capacity of such a cut, therefore, $\mathrm{val}(F_A) \leq \lambda_A$.

We say that N (and also c) is *inner Eulerian* if $c(\delta(v))$ is even for each *inner node* $v \in V_G - T$ (or, equivalently, $c(\delta(X))$ is even for every $X \subseteq V_G - T$). Lovász and Cherkassky, independently, obtained the following result.

Theorem 1 [Lovász 1976, Cherkassky 1977]. *Let $N = (G, T, c)$ be inner Eulerian. Then there exists a maximum free multiflow F which is* integral *(i.e., all numbers α_i in F are integers). Moreover, such an F satisfies* $\mathrm{val}(F) = \frac{1}{2}\sum(\lambda_s \,:\, s \in T)$, *or, equivalently,*

$$\mathrm{val}(F_s) = \lambda_s \quad \text{for all } s \in T.$$

This implies that if c is integral but not necessarily inner Eulerian, then there exists a maximum *half-integer* multiflow (i.e., with all α_i's being multiples of $1/2$). Note that the proof of this theorem given by Cherkassky (1977) is constructive and provides a solution algorithm. It applies standard augmenting path techniques combined with the so-called T-operations, and if a shortest augmenting path is always chosen at each iteration, the algorithm becomes strongly polynomial, like those of Dinitz (1970) and Edmonds and Karp (1972) for the maximum flow problem.

A different idea was suggested in Karzanov (1979a) to design a significantly faster algorithm for finding a *half-integer* maximum multiflow for an arbitrary nonnegative integer-valued c. It is based on the divide and conquer approach which recursively

reduces the problem in the current network G', T', c' to the same problems in two smaller networks with at most $\lceil T'/2 \rceil + 1$ terminals in each. In the final stage, one obtains a set of irreducible problems, each of which deals with exactly three terminals and is shown to be solved by three maximum flow computations. In fact, the whole algorithm has complexity equivalent to $O(\log p)$ max flow computations in networks with $O(n)$ nodes and $O(m)$ edges (the time bound $O(n^3 \log p)$ pointed out in that work comes up when an $O(n^3)$ max flow algorithm is applied). However, this algorithm needs not construct an *integer* maximum multiflow in the inner Eulerian case.

In this paper we first improve the above method to find an integer maximum multiflow in the inner Eulerian case. The algorithm we design has similar complexity; it runs in $O(\varphi(n,m)\log p)$ time, where $\varphi(n',m')$ is the complexity of finding a maximum flow and minimum cut in a network with n' nodes and m' edges. If we apply, as a subroutine, the currently fastest $O(nm\log(n^2/m))$ max flow min cut algorithm due to Goldberg and Tarjan (1988), then our algorithm runs in $O(nm\log(n^2/m)\log p)$ time.

We then show that this algorithm can be modified to solve an important special case of the so-called *multiflow locking problem*. As the input of this problem, a family $\mathcal{A} \subseteq 2^T$ of subsets of T is given, and one requires to

(2) Find a multiflow F in N which locks simultaneously all members of $\mathcal{A}$,

where a multiflow F is said to *lock* a set $A \subseteq T$ if $\mathrm{val}(F_A) = \lambda_A$. If such a multiflow for $\mathcal{A} \subseteq 2^T$ exists for every network (G,T,c) with T fixed, then $\mathcal{A}$ is called *lockable*. The following theorem gives a complete characterization of such families. Call a family of subsets of T 3-*cross-free* if it has no three members such that each two of them are crossing, where sets $A, B \subset T$ are *crossing* if none of $A \cap B$, $A-B$, $B-A$ and $T-(A \cup B)$ is empty.

Theorem 2 [Karzanov and Lomonosov 1978]. *$\mathcal{A}$ is lockable if and only if $\mathcal{A}$ is 3-cross-free. Moreover, if $\mathcal{A}$ is lockable and c is inner Eulerian, then there exists an* integer *multiflow which locks all members of $\mathcal{A}$.*

(For a detailed proof of this theorem, see Karzanov (1979b) and Lomonosov (1985). Shorter proofs, based on splitting-off techniques, are given in Karzanov (1984) and Frank et al. (1992).) Now call two sets $A, B \subseteq T$ *laminar* if they are not crossing. If every two members of $\mathcal{A}$ are laminar, $\mathcal{A}$ is called *laminar*. In particular, the collection $\mathcal{A}$ of all single element sets $\{s\}$, $s \in T$, is laminar, and we observe from Theorem 1 that for this particular $\mathcal{A}$, a multiflow F is a solution to the locking problem if and only if F is maximum. Thus, the multiflow locking problem for laminar families $\mathcal{A}$ generalizes the maximum free multiflow problem, and Theorem 1 follows from Theorem 2. In this

paper we show the following.

Theorem 3. *Let $\mathcal{A}$ be laminar and c inner Eulerian. Then an integer multiflow that locks all sets in $\mathcal{A}$ can be found in $O(\varphi(n,m)\log p)$ time.*

Next we consider analogs of the above problems for the directed case. We deal with a network $N^{\rightarrow} = (D, T, c)$ consisting of a digraph $D = (V_D, E_D)$, a set $T \subseteq V_D$ of terminals, and a function $c : E_D \to \mathbb{Z}_+$ of *arc* capacities. For $X \subseteq V_D$, $\delta^+(X) = \delta^+_D(X)$ is the set of arcs $(x,y) \in E_D$ *leaving* X (i.e., $x \in X \not\ni y$) and $\delta^-(X) = \delta^-_D(X)$ is the set of arcs (x,y) *entering* X (i.e., $x \notin X \ni y$). Given $A \subseteq T$, $\delta^+(X)$ is called a *dicut from* A if $X \cap T = A$, and the minimum capacity $c(\delta(X))$ for such X's is denoted by $\lambda^+_A = \lambda^+_A(c)$. Analogously, $\delta^-(X)$ is a *dicut to* A if $X \cap T = A$, and $\lambda^-_A = \lambda^-_A(c)$ is the minimum $c(\delta^-(X))$ among such dicuts. In the definition of a multiflow F in $N^{\rightarrow}$, the only difference from the undirected case is that now the paths in F are *directed* T-paths in D. For disjoint $A, B \subseteq T$, $F_{A,B}$ denotes the subset of F concerning paths *from* A *to* B (so $F_{A,B}$ and $F_{B,A}$ are now different). As before, F_A stands for $F_{A,T-A}$.

We impose the condition that $N^{\rightarrow}$ (and also c) is *inner balanced*, or *flow-like*. This means that $c(\delta^+(v)) = c(\delta^-(v))$ holds for all non-terminal nodes $v \in V_D - T$ (i.e., c itself is a flow with T as the set of terminals). An important property of such networks was established by Lomonosov.

Theorem 4 [Lomonosov 1978]. *Let $N^{\rightarrow}$ be inner balanced. Then there exists a maximum multiflow F which is integral. Moreover,* $\mathrm{val}(F_s) = \lambda^+_s$ *and* $\mathrm{val}(F_{T-s}) = \lambda^-_s$ *hold for all $s \in T$.*

(A similar result for totally balanced networks was independently obtained by Frank (1989). A strongly polynomial algorithm for finding a maximum integer multiflow in an inner balanced network is designed in Karzanov (1979b).) Note that the flow-likeness of $N^{\rightarrow}$ implies the following property: for any $s \in T$ and any $X \subset V_D$ with $X \cap T = \{s\}$,

$$c(\delta^+(X)) - c(\delta^-(X)) = c(\delta^+(s)) - c(\delta^-(s)). \tag{3}$$

Therefore, $\lambda^+_s - \lambda^-_s = c(\delta^+(s)) - c(\delta^-(s))$ and, moreover, the minimum dicuts from s and to s are induced by the same sets of nodes. In other words, in order to find λ^+_s and λ^-_s it suffices to find a minimum cut separating s and $T - s$ in the corresponding undirected network $N = (G, T, c)$, where G is the underlying undirected graph for D in which each edge e has the same capacity as that of the corresponding arc $\vec{e}$ in D, $c(e) = c(\vec{e})$.

Based on this property, one can easily reduce the maximum multiflow problem in an inner Eulerian (undirected) network $N = (G, T, c)$ to that in an inner balanced

network, as follows. Let Q be the subset of $s \in T$ with $c(\delta_G(s))$ odd. Partition Q into pairs and connect each pair by an edge with unit capacity. This makes the network $N' = (G', T, c')$ which is totally Eulerian. So we can represent c as a nonnegative integer combination of (the incidence vectors of the edge-sets) of some cycles. Therefore, one can replace each edge $e = \{x, y\}$ by two arcs (x, y) and (y, x), and assign integer capacities to these arcs with the sum to be $c(e)$ so that the directed network obtained this way is totally balanced. Now if we delete the "artificial" arcs between those pairs in Q and find a maximum integer multiflow for the resulting directed network, this gives a maximum multiflow in N.

In this paper we show that some reverse reduction is also possible. More precisely, given an inner balanced network $N^{\rightarrow} = (D, T, c)$, one can transform a maximum integer multiflow in the underlying undirected network into a maximum integer multiflow in $N^{\rightarrow}$. As a result, an $O(\varphi(n, m)\log p + n^2 m)$ algorithm is obtained to solve the maximum integer multiflow problem in $N^{\rightarrow}$.

Finally, one can suggest an analog of the locking problem for the directed case. Let us say that a multiflow F in $N^{\rightarrow}$ *locks* a set $A \subseteq T$ (in both directions) if $\mathrm{val}(F_A) = \lambda^+_A$ and $\mathrm{val}(F_{T-A}) = \lambda^-_A \ \ (= \lambda^+_{T-A})$. Then, given a family $\mathcal{A} \subseteq 2^T$, the *directed version of the locking problem* is to find a multiflow F in $N^{\rightarrow}$ that locks all members of $\mathcal{A}$. For T fixed, $\mathcal{A}$ is called *lockable* with respect to the directed case if this problem has a solution for every *inner balanced* $N^{\rightarrow} = (D, T, c)$. We prove the following fact.

Theorem 5. *For the directed case, $\mathcal{A}$ is lockable if and only if $\mathcal{A}$ is laminar. Moreover, if $\mathcal{A}$ is laminar and $N^{\rightarrow} = (D, T, c)$ is inner balanced, then the locking problem has a solution, which is* integral *and can be found in $O(\varphi(n, m)\log p + n^2 m)$ time, where $n = |V_D|$, $m = |E_D|$ and $p = |T|$.*

Remark 1. An instance of the locking problem may be solvable even if $\mathcal{A}$ is not laminar. Given an inner balanced $N^{\rightarrow} = (D, T, c)$, suppose that $\mathcal{A} - \mathcal{A}_D$ is laminar, where $\mathcal{A}_D$ consists of all $A \in \mathcal{A}$ such that for some $X \subset V_D$ with $X \cap T = A$, at least one of the sets $\delta^+(X)$ and $\delta^-(X)$ is empty. Then an integer multiflow that locks all sets in $\mathcal{A}$ does exist (even if $\mathcal{A}$ is not laminar), as we show in Section 4. This extends the part of Theorem 5 concerning the solvability of the locking problem.

The structure of this paper is as follows. The algorithm for finding an integer maximum free multiflow in an inner Eulerian network is given in Section 2. Section 3 describes how to modify this algorithm to find an integer solution to the locking problem with $\mathcal{A}$ laminar and c inner Eulerian. The algorithm for inner balanced networks and the proof of Theorem 5 are given in Section 4.

It should be noted that, in the above discussion, flows and multiflows are rep-

resented in the *paths packing* form (i.e., via a set of weighted T-paths), as this is often more convenient from the combinatorial viewpoint. However, in order to get the above-mentioned time bounds, our algorithms need to handle, in intermediate steps, the standard *node-arc* representation of flows. More precisely, return to the undirected case, and let $\vec{G}$ be the digraph with the same node-set V_G whose arc-set $\vec{E}_G$ is obtained by replacing each edge $e = \{x, y\}$ of G by two arcs (x, y) and (y, x). For disjoint subsets $A, B \subseteq T$, a flow from *source* set A to *target* set B is a function $f' = f'_{A,B} : \vec{E}_G \to \mathbb{R}_+$ satisfying the *conservation condition*

$$\mathrm{d}_{f'}(x) := \sum_y f'(x, y) - \sum_y f'(y, x) = 0 \qquad \text{for each } x \in V_G - (A \cup B),$$

and its value, $\mathrm{val}(f')$, is defined to be $|\mathrm{d}_{f'}(A)| = |\mathrm{d}_{f'}(B)|$, where $\mathrm{d}_{f'}(x)$ is called the *divergency* of f' at x. We usually assume that $\mathrm{d}_{f'}$ is nonnegative within A, and nonpositive within B, which will lead to no loss of generality in our considerations. When needed, a flow F' between A and B in the paths packing form can be transformed into a node-arc flow f' from A to B so that $\mathrm{val}(f') = \mathrm{val}(F')$ and $f'(x, y) + f'(y, x) \leq \zeta^{F'}(e)$ holds for all $e = \{x, y\} \in E_G$, and, conversely, a node-arc flow f' from A to B can be transformed into a paths packing flow F' between A and B so that $\mathrm{val}(F') = \mathrm{val}(f')$ and $\zeta^{F'}(e) \leq |f'(x, y) - f'(y, x)|$ holds for all $e = \{x, y\} \in E_G$ (due to the well-known flow decomposition procedure, see Ford and Fulkerson (1962)). (In this moment we do not touch computational complexity aspects for either procedure.) To distinguish between these two representations, we use capital (resp. small) letters for flows in the former (resp. latter) form, and similarly for multiflows.

For terminals $s, s' \in T$ and disjoint sets $A, B \subseteq T$, we write $\{s, s'\} \in \{A, B\}$ if s belongs to one and s' to the other of A, B. We will essentially use a non-expensive representation of a free multiflow f in the node-arc form as being a collection $\{f_{A_1,B_1}, \ldots, f_{A_r,B_r}\}$ of flows, where the pairs $\{A_i, B_i\}$ form a *covering nested family*. This means that

(4) (i) for any distinct $s, s' \in T$, there is a unique i such that $\{s, s'\} \in \{A_i, B_i\}$; and

(ii) for $1 \leq i < j \leq r$, either $(A_i \cup B_i) \cap (A_j \cup B_j) = \emptyset$ or $A_j \cup B_j \subseteq A_i$ or $A_j \cup B_j \subseteq B_i$.

One can see that r is exactly $p - 1$, and that for each $s \in T$, there is an i such that either $A_i = \{s\}$ or $B_i = \{s\}$. The (total) *value*, $\mathrm{val}(f)$, of f is $\sum(\mathrm{val}(f^i) : i = 1, \ldots, r)$, where f^i stands for f_{A_i,B_i}. Besides, we will also deal with certain "local values" of f. Such a value $\mathrm{val}(f, Z)$ is defined for a subset $Z \subseteq T$, to be the sum of numbers $|\mathrm{d}_{f^i}(Z \cap (A_i \cup B_i))|$ among all $i \in \{1, \ldots, r\}$ such that neither Z nor $T - Z$ entirely includes $A_i \cup B_i$ (then $\mathrm{val}(f, Z) = \mathrm{val}(f, T - Z)$). In particular, for a terminal s, $\mathrm{val}(f, s)$ coincides with $\sum(|\mathrm{d}_{f^i}(s)| : i = 1, \ldots, r)$. The above definitions are justified by the

easy fact that if F is a paths packing multiflow corresponding to f (i.e., the union of paths packing flows obtained from the f^i's as said above), then $\mathrm{val}(F) = \mathrm{val}(f)$ and $\mathrm{val}(F_s) = \mathrm{val}(f, s)$ for all $s \in T$.

Similar representations of flows and multiflows in the node-arc form are used for the directed case as well (with the difference that for a pair $\{A_i, B_i\}$, there are two flows f_{A_i,B_i} and f_{B_i,A_i}).

For a survey of some relevant results on flows and multiflows, we also refer the reader to Ahuja et al. (1993), Frank (1995), and Goldberg et al. (1990).

2. Maximum multiflow algorithm

We may assume that $p \geq 3$. If $p > 3$, the algorithm recursively applies the following *network decomposition procedure.* Partition T into T_1 and $T_2 = T - T_1$ such that $|T_i| \leq \lceil |T|/2 \rceil$, $i = 1, 2$, and find a minimum cut $C = \delta(X)$ in (G, c) which separates T_1 and T_2. Let for definiteness $T_1 \subseteq X$; so $T_2 \subseteq V_G - X$. Shrink the subnetwork induced by $V_G - X$ to a special node t', forming $N' = (G', T', c')$. Here $T' = T_1 \cup t'$; $V_{G'} = X \cup t'$; c' coincides with c within the subgraph induced by X; each edge e of the form $\{t', x\}$ is created by merging the (nonempty) set of edges in C with one end at x, and $c'(e)$ is the sum of capacities of these edges. Similarly, shrink the subnetwork induced by X to a special node t'', forming $N'' = (G'', T'', c'')$. Let C' and C'' be the sets of edges incident to t' and t'', respectively. Then

$$c'(C') = c''(C'') = c(C). \tag{5}$$

Unless $|T'| = 3$, repeat the procedure for N', decomposing it into two networks with smaller sets of terminals, do similarly for N'', and so on. Eventually, we obtain a collection of networks, each one containing exactly three terminals. Obviously, each network appeared is inner Eulerian.

In the rest of this section we first describe how to find a maximum integer multiflow in a resulting network with three terminals (which is the core of the method), and then we explain how to recursively combine the obtained multiflows so as to get the desired maximum multiflow in the original network.

2.1. Algorithm for three terminal case. For convenience we keep the same notation $N = (G, T, c)$. Let $T = \{s_1, s_2, s_3\}$. First we find a maximum integer flow f from $\{s_2, s_3\}$ to s_1. Then

$$-\mathrm{d}_f(s_1) = \lambda_{s_1} = c(\delta(X)), \tag{6}$$

where $\delta(X)$ is a minimum cut separating $\{s_2, s_3\}$ and s_1. Next we transform f so as to make the divergency at s_2 as large as possible while preserving the divergency at s_1. This is done by use of standard techniques of constructing a maximum flow in the digraph $\vec{G}$ endowed with the arc capacities c_f, defined by

$$c_f(e) = c(e) - f(e) + f(\overline{e}) \quad \text{for } e \in \vec{E}_G \tag{7}$$

(hereinafter, for an arc $e = (x, y)$, $\overline{e}$ denotes the reverse arc (y, x), and $c(e) = c(\overline{e})$ is defined to be the capacity c on the corresponding edge $\{x, y\}$). More precisely, find a maximum integer c_f-admissible flow g from s_2 to s_3, and update f by

$$f'(e) = \max\{0, f(e) + g(e) - f(\overline{e}) - g(\overline{e})\} \quad \text{for } e \in \vec{E}_G. \tag{8}$$

This together with (7) implies that f' is c-admissible, and $\mathrm{d}_{f'}(v) = \mathrm{d}_f(v) + \mathrm{d}_g(v)$ for each node v. In particular, $-\mathrm{d}_{f'}(s_1) = -\mathrm{d}_f(s_1) = \lambda_{s_1}$. To see that f' is maximum at s_2, consider the set Y of nodes reachable from s_2 by augmenting paths with respect to c_f and g (recall that a path $P = (x_0, e_1, x_1, \ldots, e_k, x_k)$ is *augmenting* if each forward arc $e_i = (x_{i-1}, x_i)$ satisfies $g(e_i) < c_f(e_i)$ and each backward arc $e_i = (x_i, x_{i-1})$ satisfies $g(e_i) > 0$). Since g is maximum, $s_3 \notin Y$; also $Y \cap X = \emptyset$ (as (6) and (7) imply that for each $e \in \delta^-(X)$, $c_f(e) = 0$, whence $g(e) = g(\overline{e}) = 0$). For each $e \in \delta^+(Y)$, we have $g(e) = c_f(e) = c(e) - f(e) + f(\overline{e})$ and $g(\overline{e}) = 0$. This implies

$$\mathrm{d}_g(s_2) = g(\delta^+(Y)) - g(\delta^-(Y)) = c(\delta(Y)) - f(\delta^+(Y)) + f(\delta^-(Y)) = c(\delta(Y)) - \mathrm{d}_f(s_2).$$

Therefore, $\mathrm{d}_{f'}(s_2) = c(\delta(Y)) = \lambda_{s_2}$.

Now our aim is to transform f' into a maximum integer multiflow. The method will rely on the following lemma. Note that each $e \in \vec{E}_G$ satisfies $c_{f'}(e) + c_{f'}(\overline{e}) = 2c(e)$, therefore, $c_{f'}(e)$ and $c_{f'}(\overline{e})$ have the same parity. For $x \in V_G$, let $Q(x)$ be the set of edges $\{x, y\} \in \delta(x)$ with $c_{f'}(x, y)$ odd.

Lemma 2.1. *$|Q(x)|$ is even for each node $x \in V_G$.*

Proof. Consider $x \in V_G$, and let $a = c_{f'}(\delta^+(x))$ and $b = c(\delta(x))$. For any $e \in \delta^+(x)$, $c_{f'}(e) = c(e) - f'(e) + f'(\overline{e})$. Therefore, $a = b - \mathrm{d}_{f'}(x)$. Obviously a and $|Q(x)|$ have the same parity, so it suffices to show that a is even. If $x \in V_G - T$, the latter follows from the facts that b is even (as c is inner Eulerian) and $\mathrm{d}_{f'}(x) = 0$.

Let $x = s_1$. Consider the above minimum cut $\delta(X)$ separating s_1 and $\{s_2, s_3\}$. Since $c(\delta(y))$ is even for each $y \in X - s_1$, the numbers b and $c(\delta(X))$ have the same parity. Also $-\mathrm{d}_{f'}(s_1) = c(\delta(X))$. Therefore, a is even. For $x = s_2$, the argument is similar, considering the minimum cut $\delta(Y)$ separating s_2 and $\{s_1, s_3\}$ and using the equality $c(\delta(Y)) = \mathrm{d}_f(s_2)$. Finally, for $x = s_3$, the evenness of a follows from that for all

other nodes and the facts that $\sum(c(\delta(y)) : y \in V_G)$ is even and $\sum(\mathrm{d}_{f'}(y) : y \in V_G) = 0$.
•

Thus, the set $Q = \cup(Q(x) : x \in V_G)$ forms an Eulerian subgraph in G and, therefore, can be partitioned into pairwise edge-disjoint circuits $C_1, \ldots, C_r$. We change f' by one along each circuit $C_i = (x_0, \{x_0, x_1\}, x_1, \ldots, \{x_{k-1}, x_k\}, x_k = x_0)$. More precisely, for each arc $e_j = (x_{j-1}, x_j)$, if $f'(e_j) < c(e_j)$, then update $f'(e_j) := f'(e_j)+1$, while if $f'(e_j) = c(e_j)$ (implying $f'(\overline{e}_j) > 0$ as $c_{f'}(e_j) \neq 0$), then update $f'(e_j) := f'(\overline{e}_j) - 1$. Clearly the resulting flow f' is c-admissible and preserves the divergencies at all nodes. Moreover, now $c_{f'}(e)$ is even for all arcs e.

Using this property of f', we construct the desired multiflow as follows. Without loss of generality one may assume that

$$-\mathrm{d}_{f'}(s_1) \geq \mathrm{d}_{f'}(s_2), \tag{9}$$

or, equivalently, $\mathrm{d}_{f'}(s_3) \geq 0$ (otherwise take the reverse flow f'', defined by $f''(e) = f'(\overline{e})$, and permute s_1 and s_2). Define $\sigma = c_{f'}/2$; then σ is integer-valued. Find a maximum integer σ-admissible flow h from s_3 to s_2. This h is just the flow from s_3 to s_2 in the desired multiflow that we construct. Note that $2h$ is a maximum $c_{f'}$-admissible flow from s_3 to s_2, and we are going to use the second copy of h to update the flow f' from $\{s_2, s_3\}$ to s_1.

Let Z be the set of nodes reachable from s_3 by augmenting paths concerning σ and h. Then $s_2 \notin Z$ and $X \cap Z = \emptyset$. Furthermore, Z is exactly the reachable set for $c_{f'}$ and $2h$, whence we conclude that

$$\mathrm{d}_{f'}(s_3) + \mathrm{d}_{2h}(s_3) = c(\delta(Z)). \tag{10}$$

So $\delta(Z)$ is a minimum cut separating s_3 and $\{s_1, s_2\}$.

Without loss of generality one may assume that for each arc e, at least one of $h(e)$ and $h(\overline{e})$ is zero, and similarly for f' (using the fact that $c_{f'}$ does not change under the same decrease of $f'(e)$ and $f(\overline{e})$). Let f'' be obtained as in (8) with f' and h in place of f and g, respectively. Since h is integral, f'' is integral too. We assert that the multiflow f^* consisting of the two flows f'' and h is c-admissible and maximum.

By (10), $\mathrm{d}_{f''}(s_3) + \mathrm{d}_h(s_3) = \lambda_{s_3}$. Also $\mathrm{d}_{f''}(s_1) = \mathrm{d}_{f'}(s_1) = \lambda_{s_1}$ (as $\mathrm{d}_h(s_1) = 0$), while $\mathrm{d}_{f''}(s_2) = \mathrm{d}_{f'}(s_2) + \mathrm{d}_h(s_2)$ together with $\mathrm{d}_{f'}(s_2) = \lambda_{s_2}$ and $\mathrm{d}_h(s_2) \leq 0$ gives $\mathrm{d}_{f''}(s_2) + |\mathrm{d}_h(s_2)| = \lambda_{s_2}$ (note that $2h(e) \leq c_{f'}(e) = f'(\overline{e})$ for $e \in \delta^-(Y)$ implies $|\mathrm{d}_h(s_2)| \leq \mathrm{d}_{f'}(s_2)$; so $\mathrm{d}_{f''}(s_2)$ if nonnegative). So f^* is a maximum multiflow, provided that f^* is c-admissible.

To see that f^* is c-admissible, consider arcs $e, \overline{e}$ and show that $q = f''(e) + h(e) + f''(\overline{e}) + h(\overline{e})$ does not exceed $c(e)$. Three cases are possible, letting for definiteness

$f'(\overline{e}) = 0$. (i) Let $h(\overline{e}) = 0$. Then $2h(e) \le c_{f'}(e) = c(e) - f'(e)$ gives $q = f''(e) + h(e) = f'(e) + 2h(e) \le c(e)$. (ii) Let $h(e) = 0$ and $f'(e) \ge h(\overline{e})$. Then $f''(e) = f'(e) - h(e)$ and $f''(\overline{e}) = 0$, whence $q = f''(e) + h(\overline{e}) = f'(e) \le c(e)$. (iii) Let $h(e) = 0$ and $f'(e) < h(\overline{e})$. Then $f''(e) = 0$ and $f''(\overline{e}) = h(\overline{e}) - f'(e)$, and now $2h(\overline{e}) \le c_{f'}(\overline{e}) = c(e) + f'(e)$ yields $q = f''(\overline{e}) + h(\overline{e}) = 2h(\overline{e}) - f'(e) \le c(e)$.

Thus, f^* is integer, c-admissible and maximum, as required. The above algorithm applies three maximum flow computations in $\vec{G}$ (namely, it constructs flows f, g and h), and it is easy to see that the number of other operations is linear in $|V_G| + |E_G|$. Therefore, the complexity of this algorithm is $O(\varphi(n, m))$.

2.2. Finding a maximum multiflow in the initial network. In the process described in the beginning of this section, we, in fact, implicitly constructed a binary tree τ of networks, in which every pair of children networks was created from the parent one by applying the network decomposition procedure. Then we found a maximum integer multiflow in each leaf (final) network of this tree. Now, going in the reverse direction, we recursively combine the occurring multiflows in each pair of children networks in order to eventually construct the desired multiflow in the initial network. More precisely, the *multiflow aggregation procedure* considers two children networks $N' = (G', T', c')$ and $N'' = (G'', T'', c'')$ of the same parent network $\widetilde{N} = (\widetilde{G}, \widetilde{T}, \widetilde{c})$, along with their maximum integer multiflows f' and f'', respectively, and outputs a maximum integer multiflow f in $\widetilde{N}$.

We assume by induction that f' and f'' are given in the node-arc form as being collections of flows $f'_1 = f'_{A'_1, B'_1}, \ldots, f'_r = f'_{A'_r, B'_r}$ and $f''_1 = f''_{A''_1, B''_1}, \ldots, f''_q = f''_{A''_q, B''_q}$, respectively, where $\omega' = (\{A'_1, B'_1\}, \ldots, \{A'_r, B'_r\})$ and $\omega'' = (\{A''_1, B''_1\}, \ldots, \{A''_q, B''_q\})$ are nested families on T' and T'', respectively (cf. (4)). Recall that by Theorem 1 the maximality of f' means that $\mathrm{val}(f', s) = \lambda_s$ holds for all $s \in T'$, and similarly for f'' (the local values $\mathrm{val}(\cdot, s)$ are defined in the Introduction).

Let I (resp. J) be the set of indices i (resp. j) such that $t' \in A'_i \cup B'_i$ (resp. $t'' \in A''_j \cup B''_j$), where t' and t'' are the special terminals in N' and N'' that respectively appeared when $\widetilde{N}$ was decomposed as described in the beginning of this section. Reversing some flows in f' and f'' if needed, one may assume that $t' \in A'_i$ for each $i \in I$, and $t'' \in B''_j$ for each $j \in J$.

For each $i \in I$, apply a flow decomposition procedure to extract from f'_i a maximum integer subflow g_i from t' to B'_i. That is, $g_i(e) \le f'_i(e)$ for each $e \in \vec{E}_{G'}$, and $\mathrm{d}_{g_i}(t') = \mathrm{d}_{f'_i}(t')$. Put these flows together, forming the flow g from t' to $T' - t'$ such that $\mathrm{d}_g(t') = \sum(\mathrm{d}_{g_i}(t') : i \in I) = \lambda_{t'}(c')$. Formally, for $e \in \vec{E}_{G'}$, define $g(e) = \max\{0, \sum_{i \in I}(g_i(e) - g_i(\overline{e}))\}$. Similarly, for each $j \in J$, extract from f''_j a maximum integer flow h_j from A''_j to t'', and combine them to the flow h from $T'' - t''$ to t'' such that $-\mathrm{d}_h(t'') = \lambda_{t''}(c'')$.

From the minimality of the cut $\delta_{G'}(t')$ and the maximality of f' at t' it follows that $g(e) = c'(e)$ and $g(\overline{e}) = 0$ for all $e \in \delta^+(t')$; similarly, $h(e) = c''(e)$ and $h(\overline{e}) = 0$ for all $e \in \delta^-(t'')$. Therefore, we can combine g and h in a flow $\widehat{f}$ from $\widetilde{T}_2 = T'' - t''$ to $\widetilde{T}_1 = T' - t'$ in $\widetilde{N}$ in a natural way, using the fact that the cuts $\delta(t')$, $\delta(t'')$ and $\delta(X)$ have the same capacities in their networks, where X is the set of nodes in $\widetilde{G}$ corresponding to $V_{G'} - t'$ (cf. (5)). Formally, $\widehat{f}$ coincides with g within the subgraph of $\widetilde{G}$ induced by X, coincides with h within the subgraph induced by $V_{\widetilde{G}} - X$, and obeys $\widehat{f}(e) = \widetilde{c}(e)$ and $\widehat{f}(\overline{e}) = 0$ for all $e \in \delta^-(X)$. Then $\widehat{f}$ is a maximum flow from $\widetilde{T}_2$ to $\widetilde{T}_1$.

Now we form the multiflow f in $\widetilde{N}$ by adding to $\widehat{f}$ each flow $\widehat{f}'_i$ that is equal to $f'_i - g_i$ if $i \in I$, and f'_i otherwise, and each flow $\widehat{f}''_j$ that is equal to $f''_j - h_j$ if $j \in J$, and f''_j otherwise. If for some $i \in I$, the set A'_i consists of the only terminal t', then $\widehat{f}'_i$ vanishes in f (in this case $\widehat{f}'_i$ has zero divergency at each node). Similarly, $\widehat{f}''_j$ vanishes if $B''_j = \{t''\}$. Clearly the representation of f matches the covering nested family ω on $\widetilde{T}$ consisting of the pair $\{\widetilde{T}_1, \widetilde{T}_2\}$, the pairs $\{A'_i - t', B'_i\}$ with $A'_i \neq t'$, and the pairs $\{A''_j, B''_j - t''\}$ with $B''_j \neq t''$. Additional properties are exhibited in the following statement (this will be use in full to prove locking properties in Section 3, while now we only need case $Z = \{s\}$ for $s \in \widetilde{T}$).

Statement 2.2. *For any $Z \subseteq \widetilde{T}_1$, $\mathrm{val}(f, Z) = \mathrm{val}(f', Z)$. Similarly, for any $Z \subseteq \widetilde{T}_2$, $\mathrm{val}(f, Z) = \mathrm{val}(f'', Z)$.*

Proof. Consider $Z \subseteq \widetilde{T}_1$ and $\{A'_i, B'_i\} \in \omega'$. Let a and b be the divergencies of $\widehat{f}'_i$ at $A = (A'_i - t') \cap Z$ and $B = B'_i \cap Z$, respectively. If $t' \notin A'_i$, then $\widehat{f}'_i = f'_i$, whence $a = \mathrm{d}_{f'_i}(A)$ and $b = \mathrm{d}_{f'_i}(B)$. And if $t' \in A'_i$, then $\widehat{f}'_i = f'_i - g_i$; we have $a = \mathrm{d}_{f'_i}(A)$ (since $t' \notin Z$, and $\mathrm{d}_{g_i}(s) = 0$ for each $s \in A'_i - t'$), and $\mathrm{d}_{f'_i}(B) = b + \mathrm{d}_{g_i}(B) = b + \mathrm{d}_{\widehat{f}}(B)$ (by the construction of $\widehat{f}$). This implies $\mathrm{val}(f, Z) = \mathrm{val}(f', Z)$. For $Z \subseteq \widetilde{T}_2$ the proof is similar. •

Since $\lambda_s(\widetilde{c})$ is equal to $\lambda_s(c')$ for $s \in \widetilde{T}_1$, and to $\lambda_s(c'')$ for $s \in \widetilde{T}_2$, the maximality of both f' and f'' together with Statement 2.2 implies $\mathrm{val}(f, s) = \lambda_s(\widetilde{c})$ for all $s \in \widetilde{T}$, i.e., f is a maximum multiflow in $\widetilde{N}$.

One can estimate the number of operations in the aggregation procedure for the current $\widetilde{N}, N', N''$ as follows. Let $n(\overline{N}), m(\overline{N}), p(\overline{N})$ denote the numbers of nodes, edges and terminals in a network $\overline{N}$. We assume by induction that each flow f'_i forming f' is *acyclic*, i.e., the subgraph S'_i spanned by the *support* $\mathrm{supp}(f'_i) = \{e \in \vec{E}_{G'} : f'_i(e) \neq 0\}$ of f'_i contains no (directed) cycle; and similarly for f''. Then the above aggregation procedure can be implemented in $O(m(\widetilde{N})(|I| + |J|))$ time. Indeed, the acyclicity of each f'_i enables us to extract g_i in $O(m(N'))$ time, by a straightforward method using a topological ordering of the node set of S'_i, i.e., one compatible with

its acyclic structure (details are left to the reader). Similarly, extraction of each h_j is performed in $O(m(N''))$ time. The other operations spent take $O(m(\widetilde{N})(|I|+|J|))$ time in total. Let $I=\{i(1),,\dots,i(q)\}$ and $i(1)<\dots<i(q)$. Since ω' is nested, $A'_{i(1)}\supset A'_{i(2)}\supset\dots\supset A'_{i(q)}$. Moreover, the network decomposition procedure guarantees that for $k=1,\dots,q-1$, the cardinality of $A'_{i(k+1)}$ is at most $1+\lceil|A'_{i(k)}|/2\rceil$. Therefore, $|I|=O(\log p(N'))$, and similarly, $|J|=O(\log p(N''))$. Thus, the construction of f from f' and f'' takes $O(m(\widetilde{N})\log p(\widetilde{N}))$ time. Note that the reduced flows f'_i-g_i and f''_j-h_j are obviously acyclic because so are f'_i and f''_j; however, the new flow $\widehat{f}$ may not be acyclic. Therefore, to be consistent with the above assumption, we must make $\widehat{f}$ acyclic as well. A routine procedure that iteratively searches for a cycle within the support of $\widehat{f}$ and cancels it by uniformly reducing $\widehat{f}$ along the cycle carries out this task in $O(n(\widetilde{N})m(\widetilde{N}))$ time (such a procedure should also be applied to make the flows in all the leaf networks in τ acyclic before the aggregation process).

2.3. Complexity of the whole algorithm. We wish to show that the above algorithm runs in time $\mathcal{T}$ that is at most $C_1\varphi(n,m)\log p+C_2nm\log p$ for some appropriately chosen constants C_1 and C_2. This is true if $p=3$, as was shown in Subsection 2.1. For $p>3$, we assume that the time bound φ of the max flow min cut algorithm we apply is subject to some reasonable restrictions, namely: φ is monotone in both variables, i.e., $\varphi(\widetilde{n}',\widetilde{m}')\le\varphi(\widetilde{n},\widetilde{m})$ if $\widetilde{n}'\le\widetilde{n}$ and $\widetilde{m}'\le\widetilde{m}$, and

$$\varphi(\widetilde{n}',\widetilde{m})+\varphi(\widetilde{n}'',\widetilde{m})\le\varphi(\widetilde{n}'+\widetilde{n}'',\widetilde{m}); \tag{11}$$
$$\varphi(\widetilde{n}+2,\widetilde{m})\le\varphi(\widetilde{n},\widetilde{m})+D\widetilde{m}\log\widetilde{n},$$

where D is a constant. For instance, this is valid for the bound $O(nm\log(n^2/m))$ in Goldberg and Tarjan (1988).

For $p>3$, N is decomposed into two networks $N'=(G',T',c')$ and $N''=(G'',T'',c'')$, and we have

$$\mathcal{T}\le\mathcal{T}'+\mathcal{T}''+\varphi(n,m)+Knm. \tag{12}$$

Here $\mathcal{T}'$ and $\mathcal{T}''$ are the time to solve the problem in N' and N'', respectively, $\varphi(n,m)$ comes up when the network decomposition procedure finds a minimum cut in N, K is a constant, and the bound Knm concerns the aggregation procedure for N' and N'' (including all other operations, of smaller order, incurred during the network decomposition for N).

Assume by induction that

$$\mathcal{T}'\le C_1\varphi(n',m')\log p'+C_2n'm'\log p' \quad\text{and} \tag{13}$$
$$\mathcal{T}''\le C_1\varphi(n'',m'')\log p''+C_2n''m''\log p'',$$

denoting the corresponding set sizes for N' with primes, and for N'' with two primes.

We know that $m', m'' \leq m$ and $n' + n'' = n + 2$. Therefore,

$$\varphi(n', m') + \varphi(n'', m'') \leq \varphi(n', m) + \varphi(n'', m) \leq \varphi(n + 2, m) \leq \varphi(n, m) + Dm\log n.$$

Also $n'm' + n''m'' \leq nm + 2m$. Let for definiteness $p' \geq p''$ and let $\gamma = \log(p/p')$. Note that there is a constant $\gamma_0 > 0$ such that $\gamma \geq \gamma_0$ regardless of p and p'. Then $\log p' \leq \log p - \gamma_0$, and we have

$$\begin{aligned}(14)\qquad &\varphi(n', m')\log p' + \varphi(n'', m'')\log p'' \\ &\qquad \leq \varphi(n, m)\log p - \gamma_0\varphi(n, m) + Dm\log n \log p - D\gamma_0 m\log n\end{aligned}$$

and

$$(15)\qquad n'm'\log p' + n''m''\log p'' \leq nm\log p - \gamma_0 nm + 2m\log p - 2\gamma m_0.$$

Without loss of generality one may assume that $\log n \log p \leq n$ and $4\log p \leq \gamma_0 n$ (otherwise p, n, m are bounded, and we can get the desired time bound by choosing C_1, C_2). Then, comparing (13) with (14) and (15), one obtains

$$(16)\quad T' + T'' \leq C_1\varphi(n, m)\log p + C_2 nm\log p - C_1\gamma_0\varphi(n, m) - \frac{1}{2}C_2\gamma_0 nm + Dnm.$$

Finally, choosing C_1 and C_2 so that $C_1\gamma_0 \geq 1$ and $\frac{1}{2}C_2\gamma_0 \geq D + K$, we conclude from (12) and (16) that

$$T \leq C_1\varphi(n, m)\log p + C_2 nm\log p.$$

Thus, the algorithm runs in $O(\varphi(n, m)\log p)$ time, as required (assuming that $\varphi(n, m) \geq O(n, m)$).

Remark 2. A similar, though slightly worse, bound can be obtained by direct calculation of the number of operations, as follows. Let Δ be the *height* of the above-mentioned binary tree τ, i.e., the maximum path length from the root network N to a leaf network. For $i = 0, 1, \ldots, \Delta$, let Q_i be the set of networks in depth exactly i, and let $\overline{Q}_i$ be obtained by adding to Q_i all leaves with depth at most $i - 1$. Then $\Delta = O(\log p)$, and $\overline{Q}_\Delta$ is the set of all leaf networks. Let n_i and m_i denote the total numbers of nodes and edges, respectively, in all the members of $\overline{Q}_i$. It is easy to show by induction that $n_i \leq n + 2^{i+1}$ and $m_i \leq m + in + 2^{i+1}$, which for $i = \Delta$ gives $n_\Delta = n + O(p)$ and $m_\Delta = m + O(n\log p)$. Thus, to compute maximum multiflows in all leaf networks takes $O(\varphi(n, m + n\log p))$ time (one can even perform this by use of one maximum multiflow computation in a combined network formed by identifying the terminal sets

of all the leaf networks). Similarly, the running time to decompose all networks in a layer Q_i is $O(\varphi(n, m+in))$, and the running time for all aggregations in these networks is $O(nm + in^2)$. This gives total time of the algorithm to be $O(\varphi(n, m + n\log p)\log p)$. There is however no contradiction with the above bound because our direct calculation can be refined so as to get rid of the term $n\log p$ in φ.

Remark 3. One can imagine a situation (at least in future) that a bound $\eta(n, m)$ for the minimum cut problem is smaller than the φ for the maximum flow problem. In this case the complexity of our algorithm becomes $O(\varphi(n, m) + \eta(n, m)\log p)$. Also if the bound $\varphi(n, m)$ would be smaller than $O(nm)$, one should add $O(nm\log p)$ to the above bound.

In conclusion of this section, note that the maximum "node-arc" multiflow f found by the above algorithm can be transformed into the paths packing form in $O(nm\log p)$ time. Indeed, one may assume that every flow $f_i = f_{A_i,B_i}$ in f is described via its support and the corresponding function on it. Then to represent f_i in the paths packing form takes $O(n|\mathrm{supp}(f_i)|)$ time. Note that if f_j is another flow with $A_j \cup B_j$ disjoint from $A_i \cup B_i$, then the supports of f_i and f_j are also disjoint (this is easily shown by induction on the height of τ). This together with the fact that every chain of comparable members in the nested family for f has length $O(\log p)$ implies the above bound. Note also that if we do not need to output the multiflow in the paths packing form, then the algorithm requires $O(m\log p)$ space (assuming $n \leq O(m)$).

3. Algorithm for the laminar locking problem

In fact, the algorithm in the previous section constructs a multiflow F in $N = (G, T, c)$ which, in addition to every single terminal set $\{s\}$, locks a number of other subsets $A \subset T$. Those sets A together with their complements $T - A$ correspond to the partitions of terminals occurred in the network decomposition process. We can use this fact in order to solve the locking problem for a laminar family $\mathcal{A} \subset T$. As before, N is assumed to be inner Eulerian.

First of all we may assume that $\mathcal{A}$ consists of different proper subsets and is inclusion maximal; otherwise we simply add, step by step, new laminar sets to $\mathcal{A}$ to provide its maximality (obviously, a solution to the resulting $\mathcal{A}$ is a solution of the initial $\mathcal{A}$). Since $\mathcal{A}$ is maximal, it contains all singletons and contains the complement $T - A$ of each member $A \in \mathcal{A}$. One may assume that $p \geq 4$; else the locking problem turns into the maximum free multiflow (or flow) problem.

For $A, B, C \subseteq T$, C is said to *strictly separate* A and B if either $A \subset C \subset T - B$ or $B \subset C \subset T - A$. We say that $A, B \in \mathcal{A}$ are *neighbors* if $A \cap B = \emptyset$, $A \neq T - B$

and no third set in $\mathcal{A}$ separates A and B. One can see that if two pairs among distinct A, B, C are neighbors, then so is the third pair. The algorithm relies on the following property.

Statement 3.1. *There exists an $A \in \mathcal{A}$ such that $\frac{1}{3}|T| \leq |A| \leq \frac{1}{2}|T|$.*

Proof. Let $A \in \mathcal{A}$ be a maximum cardinality set among those satisfying $|A| \leq |T|/2$; such a set exists since $\{s\} \in \mathcal{A}$ for any $s \in T$. Let $B_1, \ldots, B_k$ be all the maximal sets in $\mathcal{A} - \{T - A\}$ which are disjoint from A (at least one such set exists since $|T - A| \geq 2$ and $\{s\} \in \mathcal{A}$ for $s \in T - A$). Then each B_i is a neighbor to A and, therefore, each two B_i, B_j $(i \neq j)$ are neighbors as well. The maximality of $\mathcal{A}$ implies that $k \leq 2$; otherwise $B_1 \cup B_2$ is not in $\mathcal{A}$ and laminar to all members of $\mathcal{A}$. Moreover, the set $T' = (T - A) \cap (T - B_1) \cap \ldots \cap (T - B_k)$ is empty; otherwise $\mathcal{A}$ is again not maximal. Hence, $k = 2$ and $\{A, B_1, B_2\}$ is a partition of T. Note that $|B_1| \leq |A|$ (otherwise $|B_1| > |T|/2$, whence $|A| < |T - B_1| < |T|/2$, contrary to the choice of A), and similarly $|B_2| \leq |A|$. This implies $|A| \geq |T|/3$. •

Now we modify the algorithm of Section 2 as follows. Given a maximal laminar family $\mathcal{A}$, we take the partition $\{T_1, T_2\}$ in the first network decomposition to be $\{A, T - A\}$ for the A as in Statement 3.1. For the network N' (with $T' = A \cup t'$), the corresponding family $\mathcal{A}' \subset 2^{T'}$ to be locked consists of all sets $B' \in \mathcal{A}$ such that $B' \subseteq A$, and their complements to T' (in particular, $\{t'\} \in \mathcal{A}'$ as $t' = T' - A$). Similarly, for the network N'' (with $T'' = (T - A) \cup t''$), the corresponding family $\mathcal{A}''$ consists of all $B'' \in \mathcal{A}$ such that $B'' \subseteq T - A$, and their complements to T''. It is easy to see that the maximality of $\mathcal{A}$ ensures that both $\mathcal{A}', \mathcal{A}''$ are maximal. Then we decompose N' and N'' in a similar way, and so on. Statement 3.1 ensures that the height of the binary tree τ of networks constructed this way is $O(\log p)$. Therefore, the resulting multiflow f in the initial network N (obtained upon termination of the aggregation process) is constructed with the same, up to a constant factor, complexity as the complexity of the maximum multiflow algorithm in Section 2, i.e., $O(\varphi(n, m)\log p)$. Finally, we transform f into a corresponding multiflow F in the paths packing form, which takes $O(nm\log p)$ additional operations, as discussed in the end of Section 2. Thus, the whole algorithm runs in $O(\varphi(n, m)\log p)$ time, as required in Theorem 3. The fact that F solves the locking problem rests on the following statement.

Statement 3.2. $\mathrm{val}(f, A) = \lambda_A$ *holds for all $A \in \mathcal{A}$.*

Proof. By induction on p. If $p = 3$, then $\mathcal{A}$ coincides with the set of singletons $\{s\}$ and their complements, and $\mathrm{val}(f, s) = \lambda_s$ is provided by the algorithm in Subsection 2.1. So assume that $p > 3$, and consider the networks $N' = (G', T', c')$ and $N'' = (G'', T'', c'')$

and the families $\mathcal{A}' \subseteq 2^{T'}$ and $\mathcal{A}'' \subseteq 2^{T''}$ obtained in the network decomposition when T is partitioned into sets $T_1 = T' - t'$ and $T_2 = T'' - t'$ with $T_1 \in \mathcal{A}$. By induction, for the corresponding multiflow f' in N', $\mathrm{val}(f', A') = \lambda_{A'}(c')$ holds for all $A' \in \mathcal{A}'$, and for the corresponding multiflow f'' in N'', $\mathrm{val}(f'', A'') = \lambda_{A''}(c'')$ holds for all $A'' \in \mathcal{A}''$.

It suffices to consider $A \in \mathcal{A}$ such that either $A \subseteq T_1$ or $A \subset T_2$ (taking into account that A and T_1 are laminar, $\mathrm{val}(f, A) = \mathrm{val}(f, T - A)$, and $T - A \in \mathcal{A}$). If $A \subseteq T_1$, then A belongs to $\mathcal{A}'$ by the construction of $\mathcal{A}'$, and now $\mathrm{val}(f, A) = \lambda_A$ follows from Statement 2.2 and the fact that $\lambda_A(c') = \lambda_A(c)$. For $A \subset T_2$, the argument is similar since $A \in \mathcal{A}''$. •

Now the required locking property for F follows from the observation that for any $A \subseteq T$, $\mathrm{val}(F_A) \geq \mathrm{val}(f, A)$.

4. Algorithm for balanced networks

Let $N^{\rightarrow} = (D, T, c)$ be an inner balanced network. First of all we note that the maximum multiflow problem in two terminal case is reduced to one maximum flow computation, due to the following simple, but important, fact.

Lemma 4.1. [Lomonosov 1978]. *Let $N'^{\rightarrow}(D', T', c')$ be an inner balanced network in which T' consists of two terminals s and t. Let g be a maximum c'-admissible flow from s to t. Let $h = c' - g$. Then h is a maximum flow from t to s, and therefore, $\{g, h\}$ is a maximum multiflow in $N'^{\rightarrow}$.*

Proof. Consider $X \subset V_{D'}$ such that $s \in X \not\ni t$ and $c'(\delta^+(X))$ is minimum. Since g is maximum, we have $h(e) = c'(e) - g(e) = 0$ for all $e \in \delta^+(X)$ and $h(e) = c'(e) - g(e) = c'(e)$ for all $e \in \delta^-(X)$. For each $x \in V_{D'} - T$, $\mathrm{d}_{c'}(x) = \mathrm{d}_g(x) = 0$ implies $\mathrm{d}_h(x) = 0$, i.e., h is a flow. Moreover, h is a maximum flow from t to s in $N'^{\rightarrow}$ since $\mathrm{d}_h(t) = h(\delta^-(X)) - h(\delta^+(X)) = c'(\delta^-(X))$. •

In case $p > 3$, we apply the network decomposition method similar to that in Subsection 2.1. More precisely, at each iteration, the network $N^{\rightarrow} = (D, T, c)$ in question is decomposed into two networks $N'^{\rightarrow} = (D', T', c')$ and $N''^{\rightarrow} = (D'', T'', c'')$ by choosing a partition $\{T_1, T_2\}$ of T (with $|T_i| \leq \lceil |T|/2 \rceil$) and finding a minimum dicut $\delta^+(X)$ from T_1 (implying that $\delta^-(X)$ is a minimum dicut to T_1 since $N^{\rightarrow}$ is inner balanced). The special terminal t' $(= T' - T_1)$ in $N'^{\rightarrow}$ obeys $c'(\delta^+(t')) = \lambda^+_{t'}(c') = c(\delta^-(X))$ and $c'(\delta^-(t')) = \lambda^-_{t'}(c') = c(\delta^+(X))$; and similarly for t'' in $N''^{\rightarrow}$.

Once maximum integer multiflows f' in $N'^{\rightarrow}$ and f'' in $N''^{\rightarrow}$ are found, the ag-

gregation procedure similar to that in Subsection 2.2 combines these into a maximum integer multiflow in $N^{\rightarrow}$. The only difference is that, instead of one maximum flow g between t' and T_1 in N', we extract from f' two flows: a flow g' from t' to T_1 with value $c'(\delta^+(t'))$ and a flow g'' from T_1 to t' with value $c'(\delta^-(t'))$; similarly, in $N''^{\rightarrow}$, there are a flow h' from t'' to T_2 with value $c''(\delta^+(t''))$ and a flow h'' from T_2 to t'' with value $c''(\delta^-(t''))$. We combine g'' with h' and g' with h'' in an obvious way to obtain maximum flows from T_1 to T_2 and from T_2 to T_1 in $N^{\rightarrow}$.

To complete this algorithm, we have to explain how to solve the problem (or subproblem) for a network $N^{\rightarrow} = (D, T, c)$ with three terminals.

4.1. Algorithm for three terminal case. Let $T = \{s_1, s_2, s_3\}$. First we apply the algorithm of Section 2 to find a maximum integer multiflow f in the underlying undirected network N; one may assume that f consists of three flows $f_{1,2}, f_{1,3}, f_{2,3}$, where $f_{i,j}$ is a flow from s_i to s_j given in the node-arc form. Denote $V = V_D$, $E = E_D$, $\delta^+ = \delta^+_D$, $\delta^- = \delta^-_D$. Let $\overline{E}$ be the set of (new) arcs $\bar{e}$ reverse to arcs e in E. Then f is such that: (i) each $g = f_{i,j}$ is a nonnegative integer function on $E \cup \overline{E}$ satisfying

$$\mathrm{d}_g(x) = \sum(g(e) - g(\bar{e}) \ : \ e \in \delta^+(x)) - \sum(g(e) - g(\bar{e}) \ : \ e \in \delta^-(x)) = 0 \qquad \text{for all } x \in V - \{s_i, s_j\}$$

(ii) f is c-admissible, i.e.,

$$\zeta^f(e) = \sum(f_{i,j}(e) + f_{i,j}(\bar{e}) \ : \ 1 \le i < j \le 3) \le c(e) \qquad \text{for all } e \in E;$$

and (iii) $|\mathrm{d}_{f_{1,2}}(s_i)| + |\mathrm{d}_{f_{1,3}}(s_i)| + |\mathrm{d}_{f_{2,3}}(s_i)| = \lambda^+_{s_i} + \lambda^-_{s_i}$ for $i = 1, 2, 3$.

For our purposes, it is more convenient to assume that all arcs are saturated, i.e.,

$$\zeta^f(e) = c(e) \qquad \text{for all } e \in E. \tag{17}$$

This does not lose generality since the above f can be modified so that (17) holds. To see this, consider the residual capacities $c_r(e) = c(e) - \zeta^f(e)$, $e \in E$, for the original f. Obviously, for each $x \in V - T$, $c_r(\delta^+(x)) + c_r(\delta^-(x))$ is an even integer; moreover, from the maximality of f (cf. (iii) above) one can see that this value is even also for $x = s_1, s_2, s_3$. This implies that c_r can be decomposed into a collection of integer weighted cycles (not necessarily directed or simple ones); this task is carried out in $O(nm)$ time. In other words, we can find a function (circulation) $h : E \cup \overline{E} \to \mathbb{Z}_+$ with $\mathrm{d}_h(x) = 0$ for all $x \in V$ and $\zeta^h(e) = c_r(e)$ for all $e \in E$. Now add h to $f_{1,2}$ by updating $f_{1,2} := f_{1,2} + h$. Then the new multiflow f is again integer and maximum, and it satisfies (17).

For a function g on $E \cup \overline{E}$ and a node $x \in V$, define the *discrepancy* of g at x to be

$$\Delta_g(x) = \sum(g(e) + g(\bar{e}) \ : \ e \in \delta^+(x)) - \sum(g(e) + g(\bar{e}) \ : \ e \in \delta^-(x)).$$

We say that g is *regular at* x if $\Delta_g(x) = 0$. If g is a flow from s_i to s_j which is regular at each node in $V - \{s_i, s_j\}$, we call g *regular*. The importance of this notion becomes clear from the following statement.

Statement 4.2. *Let a flow $g = f_{i,j}$ be regular. For $e \in E$, define $c'(e) = g(e) + g(\overline{e})$. Then* (i) *$c'$ is balanced at each node $x \in V - \{s_i, s_j\}$, i.e., $c'(\delta^+(x)) = c'(\delta^-(x))$; and* (ii) *$\lambda^+_{s_i}(c') + \lambda^-_{s_i}(c') = \mathrm{val}(g)$; in other words, there is an integer multiflow f' in $(D, \{s_i, s_j\}, c')$ for which $\mathrm{val}(f') = \mathrm{val}(g)$.*

Proof. Part (i) immediately follows from $\Delta_g(x) = 0$ for $x \in V - \{s_i, s_j\}$. To see (ii), assume for definiteness that $i < j$ and consider $X \subset V$ with $s_i \in X \not\ni s_j$. We have

$$(18) \qquad c'(\delta^+(X)) + c'(\delta^-(X)) = \sum(g(e) + g(\overline{e}) \ : \ e \in \delta^+(X) \cup \delta^-(X))$$
$$\geq \sum(g(e) - g(\overline{e}) \ : \ e \in \delta^+(X)) - \sum(g(e) - g(\overline{e}) \ : \ e \in \delta^-(X)) = \mathrm{val}(g).$$

Since c' is inner balanced, this implies $\lambda^+_{s_i}(c') + \lambda^-_{s_j}(c') \geq \mathrm{val}(g)$ (in fact, this turns into equality; this can be observed from (18) by considering a set X which gives a minimum cut separating s_i and $T - s_i$ in the underlying network N and using the maximality of f at s_i). •

Thus, if all $f_{1,2}, f_{1,3}, f_{2,3}$ are regular, then, in view of Lemma 4.1, the problem is reduced to three max flow computations. Now we explain how to make our flows regular. Let $x \in V - T$. Denote, for brevity, $g = f_{1,2}$, $g' = f_{1,3}$, $g'' = f_{2,3}$, $\Delta = \Delta_g(x)$, $\Delta' = \Delta_{g'}(x)$ and $\Delta'' = \Delta_{g''}(x)$. From (17) and $c(\delta^+(x)) = c(\delta^-(x))$ it follows that $\Delta + \Delta' + \Delta'' = 0$; so without loss of generality we may assume that

$$\Delta \geq -\Delta' \geq -\Delta'' \geq 0.$$

Let $\Delta > 0$. Note that $\mathrm{d}_g(x) = 0$ together with the integrality of g implies that Δ is even; similarly, Δ' and Δ'' are even.

First we modify g and g' to make g' regular at x and to reduce the discrepancy of g at x by $|\Delta'|$. Using the method to be described below, we extract from g and g' integer subflows $h \leq g$ and $h' \leq g'$ both from s_1 to x such that

$$(19) \qquad \mathrm{d}_h(s_1) = \mathrm{d}_{h'}(s_1) \quad \text{and} \quad \Delta_h(x) - \Delta_{h'}(x) = |\Delta'|.$$

Then we exchange the subflows h and h' in g and g'. That is, we update $g := g - h + h'$ and $g' := g' - h' + h$. By (19), g remains a flow from s_1 to s_2 and its value is preserved, while its discrepancy at x becomes $\Delta - \Delta_h(x) + \Delta_{h'}(x) = \Delta + \Delta'$, as required. Similarly, the new g' is a flow from s_1 to s_3 with the same value as before, and $\Delta_{g'}(x)$ becomes $\Delta' - \Delta_{h'}(x) + \Delta_h(x) = 0$, i.e., g' is now regular at x.