A Linear-Time Algorithm for Integral Multiterminal Flows in Trees*

Mingyu Xiao[1] and Hiroshi Nagamochi[2]

1 **School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China**
myxiao@gmail.com

2 **Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan**
nag@amp.i.kyoto-u.ac.jp

Abstract

In this paper, we study the problem of finding an integral multiflow which maximizes the sum of flow values between every two terminals in an undirected tree with a nonnegative integer edge capacity and a set of terminals. In general, it is known that the flow value of an integral multiflow is bounded by the cut value of a cut-system which consists of disjoint subsets each of which contains exactly one terminal or has an odd cut value, and there exists a pair of an integral multiflow and a cut-system whose flow value and cut value are equal; i.e., a pair of a maximum integral multiflow and a minimum cut. In this paper, we propose an $O(n)$-time algorithm that finds such a pair of an integral multiflow and a cut-system in a given tree instance with n vertices. This improves the best previous results by a factor of $\Omega(n)$. Regarding a given tree in an instance as a rooted tree, we define $O(n)$ rooted tree instances taking each vertex as a root, and establish a recursive formula on maximum integral multiflow values of these instances to design a dynamic programming that computes the maximum integral multiflow values of all $O(n)$ rooted instances in linear time. We can prove that the algorithm implicitly maintains a cut-system so that not only a maximum integral multiflow but also a minimum cut-system can be constructed in linear time for any rooted instance whenever it is necessary. The resulting algorithm is rather compact and succinct.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Multiterminal flow, Maximum flow, Minimum Cut, Trees, Linear-time algorithms

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.62

1 Introduction

The min-cut max-flow theorem by Ford and Fulkerson [5] is one of the most important theorems in graph theory. It catches a min-max relation between two fundamental graph problems. This theorem leads to many effective algorithms and much theory for flow problems as well as graph cut problems. Due to the great applications of it, researchers have interests to seek more similar min-max formulas in various kinds of flow and cut problems. In this paper, we consider the *maximum multiterminal flow problem*, a generalization of the basic maximum flow problem.

* A full version of the paper is available at https://arxiv.org/abs/1611.08803.

27th International Symposium on Algorithms and Computation (ISAAC 2016).
Editor: Seok-Hee Hong; Article No. 62; pp. 62:1–62:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the maximum flow problem, we are given two terminals (source and sink) and asked to find a maximum flow between the two terminals. A natural generalization of the maximum flow problem is the famous *maximum multicommodity flow problem*, in which, a list of pairs of source and sink for the commodities is given and the objective is to maximize the sum of the simultaneous flows in all the source-sink pairs subject to the standard capacity and flow conservation requirements. The maximum multiterminal flow problem is one of the most important special cases of the maximum multicommodity flow problem. In it, a set T of more than one terminal is given and the list of source-sink pairs is given by all pairs of terminals in T. The extensions of the maximum flow problem have been extensively studied in the history. Readers are referred to a survey [2].

A dual problem of the maximum multiterminal flow problem is the *minimum multiterminal cut problem*, in which we are asked to find a minimum set of edges whose removal disconnects each pair of terminals in the graph. The minimum multiterminal cut problem is a generalization of the minimum cut problem. When there are only two terminals, the min-cut max-flow theorem shows that the value of the maximum flow equals to the value of the minimum cut in the graph. However, when there are more than two terminals, the equivalence may not hold. Consider a star with three leaves. Each leaf is a terminal and each of the three edges has capacity 1. The flow value of a maximum multiterminal flow is 1.5 (a flow of size 0.5 routed between every pair of the three terminal pairs), whereas the size of a minimum multiterminal cut is 2. In fact, Cunningham [4] has proved a min-max theory for the pair of problems: The size of a minimum multiterminal cut is at most $(2 - 2/|T|)$ times of the flow value of a maximum multiterminal flow. A similar min-max theory for the maximum multicommodity flow problem and its dual problem is presented in [6].

In the maximum multiterminal flow problem, each edge is assigned a nonnegative capacity and a flow routed between a terminal pair is allowed to take any feasible fraction, whereas in the *integral multiterminal flow problem*, a flow is allowed to take a nonnegative integer and we are asked to find a maximum flow under this restriction. Clearly, we can simply assume that all edge capacities of the integral multiterminal flow problem are nonnegative integers. The integral multiterminal flow problem is different from the maximum multiterminal flow problem. We can see in the above example, the flow value of a maximum integral multiterminal flow is 1. The special case of the integral multiterminal flow problem where all edges have unit capacities is also known as the *T-path problem*, in which we are asked to find the maximum number of edge-disjointed paths between different terminal pairs.

In this paper, we study the maximum multiterminal flow problem in trees and give linear-time algorithms for both fractional and integer versions, which improve the best previous algorithms by a factor of $\Omega(n)$ [3]. Note that the maximum (integral) multicommodity flow problem in trees is NP-hard and there is a $\frac{1}{2}$-approximation algorithm for it [7].

The rest of the paper is organized as follows. Section 2 introduces basic notations on flows and cuts, and reviews important min-max theorems for fractional and integer versions of maximum multiterminal flow problem. Section 3 discusses instances with rooted trees, and introduces notations necessary to build a dynamic programming method over the set of $O(n)$ instances of rooted subtrees of a given instance. Informally "a blocking flow" in a rooted tree instance is defined to be a flow in the tree currently pushing maximal flows among terminals except for the terminal designated as the root. Section 4 shows several properties of blocking flows, and presents a representation of flow values of blocking flows. Section 5 provides a main technical lemma that tells how to compute the representation of flow values of blocking flows and how to construct a maximum flow from the representations. Based on the lemma, Section 6 gives a description of a linear-time algorithm for computing

the representations of flow values of blocking flows and constructing a maximum flow from the representations. Finally Section 7 makes some concluding remarks. The proofs of some lemmas are omitted due to the space limitation.

2 Preliminaries

This section introduces basic notations on flows and cuts, and reviews important min-max theorems for fractional and integer versions of maximum multiterminal flow problem. Let $\Re^+$ denote the set of nonnegative reals, and $\mathbb{Z}^+$ denote the set of nonnegative integers.

Graphs and Instances

We may denote by $V(G)$ and $E(G)$ the sets of vertices and edges of an undirected graph G, respectively. Let $G = (V, E)$ denote a simple undirected graph with a vertex set V and an edge set E, and let n and m denote the number of vertices and edges in a given graph. Let $X \subseteq V$ be a subset of vertices in G. Let $E(X)$ denote the set of edges with one end-vertex in X and the other in $V - X$, where $E(\{v\})$ for a vertex $v \in V$ is denoted by $E(v)$. Let $G - X$ denote the graph obtained from G by removing the vertices in X together with the edges in $\cup_{v \in X} E(v)$. For a vertex subset T, let $\mathcal{P}(T)$ be the set of all paths $P_{t,t'}$ with end-vertices $t, t' \in T$ with $t \neq t'$.

An instance I of a maximum flow problem consists of a graph G, a set T of vertices called terminals, and a capacity function $c : E \to \Re^+$.

Flows

For a function $h : E \to \Re^+$, $\sum_{e \in E(X)} h(e)$ for a subset $X \subseteq V$ is denoted by $h(X)$. A function $f : E \to \mathbb{Z}^+$ is called a *flow* in an instance (G, T, c) if there is a function $g : \mathcal{P}(T) \to \mathbb{Z}^+$ such that

$$f(e) = \sum \{g(P) \mid e \in E(P),\ P \in \mathcal{P}(T)\} \quad \text{for all edges } e \in E,$$

where $g(P)$ is the flow value sent along path P, and such a function g is called a *decomposition* of a flow f. A flow f is called *integer* if it admits a decomposition g such that $g(P) \in \mathbb{Z}^+$ for all paths $P \in \mathcal{P}(T)$ (note that f may not be integer even if $f(e) \in \mathbb{Z}^+$ for all edges $e \in E$).

A flow f is called *feasible* if $f(e) \leq c(e)$ for all edges $e \in E$. The *flow value* $\alpha(f)$ is defined to be $\frac{1}{2} \sum_{t \in T} f(\{t\})$, and a feasible flow f that maximizes $\alpha(f)$ is called *maximum*.

Cut-Systems

A subset X of vertices is called a *terminal set* (or a *t-set*) if $X \cap T = \{t\}$ and X induces a connected subgraph from G. A *cut-system* of T is defined to be a collection $\mathcal{X}$ of disjoint $|T|$ terminal sets X_t, $t \in T$, where $\mathcal{X}$ is not required to be a partition of V. For a cut-system $\mathcal{X}$ of T, let $\gamma(\mathcal{X}) = \sum_{X \in \mathcal{X}} c(X)$. For any pair of a feasible flow f and a cut-system $\mathcal{X}$ of T in (G, T, c), it holds

$$\alpha(f) \leq \frac{1}{2}\gamma(\mathcal{X}). \tag{1}$$

Cherkasskii [1] proved the next result.

▶ **Theorem 1.** *A feasible flow f in (G, T, c) is maximum if and only if there is a cut-system $\mathcal{X}$ such that $\alpha(f) = \frac{1}{2}\gamma(\mathcal{X})$.*

Ibaraki *et al.* [9] proposed an $O(nm \log n)$-time algorithm for computing a maximum flow f in a graph G with n vertices and m edges. Hagerup *et al.* [8] proved a characterization of the maximum multiterminal flow problem and gave an $O(\mathrm{ex}(|T|)n)$-time algorithm for the maximum multiterminal flow problem in bounded treewidth graphs, where $\mathrm{ex}(|T|)$ is an exponential function of the number $|T|$ of terminals. This algorithm runs in linear time only when $|T|$ is restricted to a constant.

An integer version of the multiterminal flow problem is defined as follows. Let $I = (G = (V, E), T, c)$ have integer capacities $c(e) \in \mathbb{Z}^+$, $e \in E$. Recall that an integral flow f is a flow which can be decomposed into integer individual flows g, i.e., $g : \mathcal{P}(T) \to \mathbb{Z}^+$. An instance (G, T, c) is called *inner-eulerian* if all edge capacities $c(e)$, $e \in E$ are integers and $c(E(v))$ is an even integer for each non-terminal vertex $v \in V - T$. It is known that any inner-eulerian instance admits a pair of a maximum integral flow f and a cut-system $\mathcal{X}$ with $\alpha(f) = \frac{1}{2}\gamma(\mathcal{X})$ [1]. In general, there is no pair of an integral flow f and a cut-system $\mathcal{X}$ with $\alpha(f) = \frac{1}{2}\gamma(\mathcal{X})$ even for trees. We review a min-max theorem on the integer version as follows.

Assume that $c(e) \in \mathbb{Z}^+$, $e \in E$. A component $W \subseteq V$ in the graph $G - \cup_{X \in \mathcal{X}} X$ is called an *odd set* in $\mathcal{X}$ if $c(W)$ is odd. Let $\kappa(\mathcal{X})$ denote the number of odd sets in $G - \cup_{X \in \mathcal{X}} X$. For each odd set W, at least one unit of capacity from $c(W)$ cannot be used by any feasible integral flow $f : E \to \mathbb{Z}^+$. Hence since each path in $\mathcal{P}(T)$ goes through edges in $E(X_t)$ of a t-set for exactly two terminals $t \in T$, we see that, for any decomposition g of f,

$$2\alpha(f) = \sum_{P \in \mathcal{P}(T)} g(P) \leq \sum_{X \in \mathcal{X}} c(X) - \kappa(\mathcal{X}) = \gamma(\mathcal{X}) - \kappa(\mathcal{X}). \tag{2}$$

Mader [10] proved the next result.

▶ **Theorem 2.** *A feasible integral flow f in (G, T, c) is maximum if and only if there is a cut-system $\mathcal{X}$ such that $\alpha(f) = \frac{1}{2}[\gamma(\mathcal{X}) - \kappa(\mathcal{X})]$.*

For trees with n vertices, an $O(n^2)$-time algorithm for computing a maximum integral flow f is proposed [3], while no strongly-polynomial time algorithm is known to general graphs (e.g., see [2]).

3 Tree Instances

In the rest of this paper, we assume that a given instance $I = (G, T, c)$ consists of a tree $G = (V, E)$, a terminal set T and an integer capacity $c(e) \in \mathbb{Z}^+$ for each $e \in E$. We simply call an integral flow a *flow*.

This section discusses instances with rooted trees, and introduces notations necessary to build a dynamic programming method over the set of $O(n)$ instances of rooted subtrees of a given instance.

If a vertex $v \in T$ is not a leaf of G, i.e., v is of degree $d \geq 2$, then we can split the instance at the cut-vertex v into d instances, and it suffices to find a maximum flow in each of these instances. Also we can split a vertex $v \in V - T$ of degree $d \geq 4$ into $d - 2$ vertices that induce a tree with edges of capacity sufficiently larger without losing the feasibility and optimality of the instance. In the rest of paper, we assume that T is the set of leaves of G, and the degree of each non-leaf is 3, and $c(e) \geq 1$ for all edges $e \in E$, as shown in Fig. 1.

For a leaf $v \in V$ in G, let e_v denote the edge incidenet to v. For two vertices $u, v \in V$, let $P_{u,v}$ denote the path connecting u and v in the tree G. For a subset $S \subseteq V$ of vertices, let $\mathcal{P}(S)$ denote the set of all paths $P_{s,s'}$ with $s, s' \in S$.

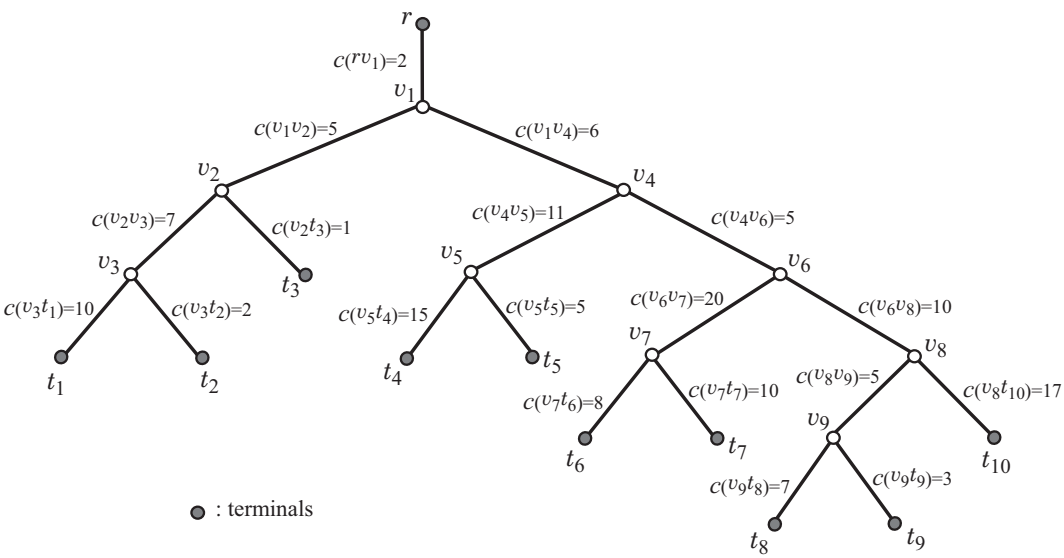

Figure 1 An example of a tree instance $I = (G, T, c)$ such that the degree of each internal vertex is 3 and all capacities are positive integers, where terminal r is chosen as the root.

In a tree instance (G, T, c), a *flow* admits a function $g : \binom{T}{2} \to \mathbb{Z}^+$ such that

$$f(e) = \sum \{g(t, t') \mid e \in E(P_{t,t'}),\ t, t' \in T\} \quad \text{for all edges } e \in E,$$

where $g(t, t')$ is the flow value sent along path $P_{t,t'}$. For a flow f, a path $P \in \mathcal{P}(T)$ is called a *positive-path* if f admits a decomposition g such that $g(t, t') > 0$.

For a path P in G, and an integer $\delta \geq -\min_{e' \in E} h(e')$ (possibly $\delta < 0$), the function $h' : E \to \mathbb{Z}^+$ obtained from h by setting $h'(e) = h(e) + \delta$ for all edges $e \in E(P)$ and $h'(e) = h(e)$ for all edges $e \in E - E(P)$ is denoted by $h + (P, \delta)$.

Rooted Tree

Choose a terminal $r \in T$, and regard G as a tree rooted at r, which defines a parent-child relationship among the vertices in G. In a rooted tree G, we write an edge $e = uv$ such that u is the parent of v by an ordered pair (u, v). For an edge $e = (u, v)$, any edge $e' = (v, w)$ is called a *child-edge* of e, and e is called the *parent-edge* of e'.

Let Y be a subset of vertices in $V - \{r\}$ such that Y induces a connected subgraph from G. Then there is exactly one edge $(u, v) \in E(Y)$ such that $v \in Y$ and u is the parent of v, and we call the edge uv the *parent-edge* of Y while any other edge in $E(Y)$ is called a *child-edge* of Y.

For an edge $e = (u, v) \in E$, let $V_e \subseteq V$ denote the set of vertex u and all the descendants of v including v itself, $G_e = (V_e, E_e)$ denote the graph induced from G by V_e, and let $T_e = (T \cap V_e) - \{u\}$, where we remark that $u \notin T_e$. Let $I(e)$ denote an instance $(G_e, T_e \cup \{u\}, c)$ induced from (G, T, c) by the vertex subset V_e, where we remark that u is included as a terminal in the instance $I(e)$.

Blocking Flows

Informally "a blocking flow" in a rooted tree instance is defined to be a flow in the tree currently pushing maximal flows among terminals except for the terminal designated as

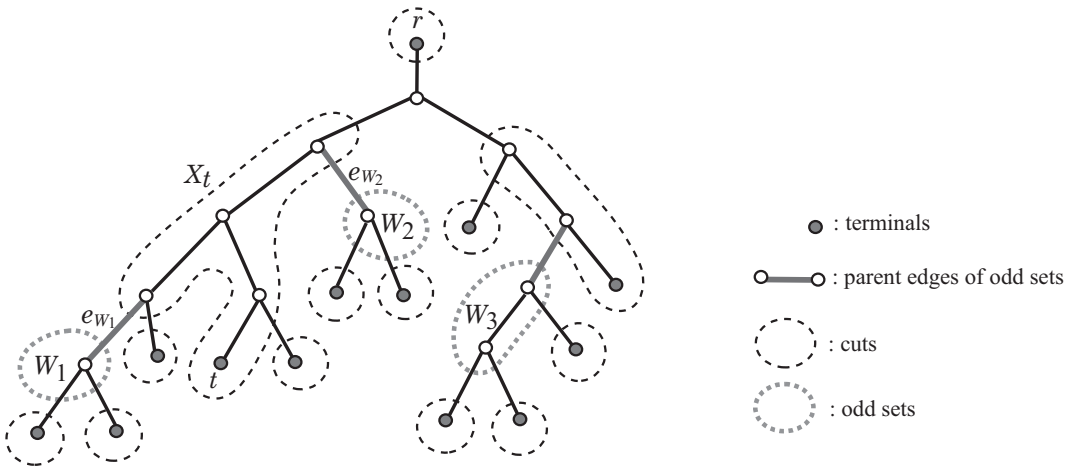

Figure 2 Illustration of a cut-system $\mathcal{X}$ and the family $\mathrm{odd}(X_t) = \{W_1, W_2\}$ for a terminal set $X_t \in \mathcal{X}$.

the root. Let $\mathcal{X}$ be a cut-system of T_e in $I(e)$ for some edge $e = (u, v)$. An odd set W in $G_e - \cup_{X \in \mathcal{X}} X$ is called an *odd set* of a terminal set $X \in \mathcal{X}$ if the parent-edge of W is a child-edge of X, where $u \notin X$ implies $r, u \notin W$. For each terminal set $X \in \mathcal{X}$, let $\mathrm{odd}(X)$ denote the family of odd sets of X, i.e., W of $\mathcal{X}$ whose parent-edge e_W is a child-edge of X. Fig. 2 illustrates a cut-system $\mathcal{X}$ and the family $\mathrm{odd}(X_t) = \{W_1, W_2\}$.

For a function $h : E \to \Re_+$, let $E[h; k]$ denote the set of edges $e \in E$ such that $h(e) \geq k$.

Let f be a feasible flow of $I(e)$ for an edge $e = (u, v)$. We call a terminal set $X \in \mathcal{X}$ with $t \in X \cap T$ *blocked* (or *blocked by* f) if

$$f(e_t) = f(X) = c(X) - |\mathrm{odd}(X)|,$$

and call $\mathcal{X}$ *blocked* (or *blocked by* f) if all terminal sets in it are blocked by f.

For each vertex $s \in V_e$, we define $V_f(s)$ to be the set of vertices $w \in V_e$ reachable from s by a path $P_{s,w'}$ from s to the common ancestor w' of s and w using edges in $E[c - f; 1]$ and by a path $P_{w',w}$ from w' to w using edges in $E[c - f; 2]$. In other words, we travel an edge e' upward if $c(e') - f(e') \geq 1$ and downward if $c(e') - f(e') \geq 2$ from s to w. By the definition of $V_f(s)$, we can see that $V_f(s)$ induces a connected subgraph, the parent-edge e' of $V_f(s)$ satisfies $f(e') = c(e')$, and any child-edge e' of $V_f(s)$ satisfies $f(e') \in \{c(e') - 1, c(e')\}$.

We call f *blocking* if $\{V_f(t) \mid t \in T_e\}$ is a cut-system of T_e blocked by f. Let $\Psi(e)$ denote the set of integers x such that $I(e)$ has a blocking flow $f(e) = x$.

Interval Computation

Our dynamic programming approach to compute the maximum flow value updates the set of flow values of blocking flows recursively. As it will be shown in Section 4, such a set of flow values always is given by an interval that consists of consecutive odd or even integers, and we here introduce a special operation on such types of intervals.

For two reals a, b with $a \leq b$, let $[a, b]$ denote the set of reals s with $a \leq s \leq b$.

For two integers $k, a \in \mathbb{Z}^+$, the set $\{a + 2i \mid i = 0, 1, \ldots, k\}$ of consecutive odd or even integers is denoted by $\langle a, b \rangle$, where $b = 2k + a$. For two sets $A, B \subseteq \mathbb{Z}^+$ of nonnegative integers, let $A \otimes B$ denote the set of nonnegative integers $\{a + b - 2i \mid i = 0, 1, \ldots, \min\{a, b\}\}$ over all $a \in A$ and $b \in B$. In particular, for sets $A_1 = \langle a_1, b_1 \rangle$ and $A_2 = \langle a_2, b_2 \rangle$, we observe

that

$$A_1 \otimes A_2 = \begin{cases} \langle 0, b_1 + b_2 \rangle & \text{if } A_1 \cap A_2 \neq \emptyset \\ \langle 1, b_1 + b_2 \rangle & \text{if } a_2 \leq b_1,\ a_1 \leq b_2 \text{ and } A_1 \cap A_2 = \emptyset \\ \langle a_1 - b_2, b_1 + b_2 \rangle & \text{if } b_2 < a_1 \\ \langle a_2 - b_1, b_1 + b_2 \rangle & \text{if } b_1 < a_2. \end{cases}$$

Given an integer $x \in A_1 \otimes A_2$, we can find in $O(1)$ time three integers $x_i \in \langle a_i, b_i \rangle$, $i = 1, 2$ and $y \in [0, \min\{x_1, x_2\}]$ such that $x = x_1 + x_2 - 2y$. To see this, assume that $b_1 \leq b_2$ without loss of generality, and let a'_2 be the minimum element in $\langle a_2, b_2 \rangle$ with $b_1 \leq a'_2$, where $a'_2 \in \{b_1, b_1 - 1, a_2\}$. Observe that $\{x \in A_1 \otimes A_2 \mid x \leq b_2 - b_1\} = \{b_1 + x_2 - 2b_1 \mid x_2 \in \langle a'_2, b_2 \rangle\}$ and $\{x \in A_1 \otimes A_2 \mid x > b_2 - b_1\} = \{b_1 + b_2 - 2y \mid y = 0, 1, \ldots, b_1 - 1\}$. Hence if $x \leq b_2 - b_1$ then let $x_1 = y = b_1$ and $x_2 = x + b_1$; otherwise $x_1 = b_1$, $x_2 = b_2$ and $y = (x - b_1 - b_2)/2$.

4 Basic Properties on Blocking Flows

This section shows several properties of blocking flows, and presents a representation of flow values of blocking flows. We first observe two lemmas on some properties of blocking flows.

▶ **Lemma 3.** *Let f be a feasible flow in $I(e)$ for an edge $e \in E$.*

(i) *For a terminal $t \in T_e$, let X_t be a t-cut such that $f(X_t) = f(e_t)$ and $P_{s,s'}$ be a positive-path of f with $s, s' \in T_e \cup \{u\}$. If $t \in \{s, s'\}$ then $P_{s,s'}$ contains exactly one edge in $E(X_t)$, and otherwise $P_{s,s'}$ is disjoint with X_t.*

(ii) *Assume that $V_f(t) \cap V_f(t') = \emptyset$ for any two $t, t' \in T_e$. Then $V_f(u)$ is disjoint with $V_f(t)$ of any terminal $t \in T_e$, and the following holds:*

(1) *For each edge $e' \in E(V_f(t))$ with $t \in T_e \cup \{u\}$,*

$$f(e') = \begin{cases} c(e') - 1 & \textit{if } e' \textit{ is the parent-edge of an odd set } W \in \mathrm{odd}(V_f(t)) \\ c(e') & \textit{otherwise.} \end{cases}$$

(2) *$f(V_f(t)) = c(V_f(t)) - |\mathrm{odd}(V_f(t))|$ for each $t \in T_e \cup \{u\}$.*

(iii) *Flow f is blocking if $V_f(t) \cap V_f(t') = \emptyset$ for any two $t, t' \in T_e$, and $f(e_t) = f(V_f(t))$ for each $t \in T_e$.*

(iv) *When f is blocking, any edge $e' \in E_e$ with $f(e') = c(e')$ satisfies $c(e') \in \Psi(e')$.*

(v) *When f is blocking, the parent-edge e_W of any odd set $W \in \mathrm{odd}(V_f(t))$ for a terminal $t \in T_e$ satisfies $c(e_W) - 1 \in \Psi(e')$.*

A proof of this lemma can be found in the full version of this paper.

The next lemma tells how to obtain a maximum flow and a minimum cut-system in an instance $I(e)$.

▶ **Lemma 4.** *For an edge $e = (u, v) \in E$, let f be a blocking flow in $I(e)$ such that $f(e)$ is the maximum in $\Psi(e)$. Then $\mathcal{X} = \{V_f(t) \mid t \in T_e \cup \{u\}\}$ is a cut-system in $I(e)$ satisfying $2\alpha(f) = f(e) + \sum_{t \in T_e} f(e_t) = \gamma(\mathcal{X}) - \kappa(\mathcal{X})$ (hence f is a maximum flow in $I(e)$ by (2)).*

Proof. Since f is a blocking flow in $I(e)$, the family $\{V_f(t) \mid t \in T_e\}$ is a cut-system of T_e blocked by f by definition, and we know that $f(e_t) = f(V_f(t)) = c(V_f(t)) - |\mathrm{odd}(V_f(t))|$ for all terminals $t \in T_e$. First we see that $V_f(u)$ is disjoint with $V_f(t)$ of any terminal $t \in T_e$, since the vertices in $V_f(u)$ are spanned with edges in $E[c - f; 2]$ and the parent-edge of $V_f(t)$ is saturated by f. By Lemma 3(ii), we have $f(V_f(u)) = c(V_f(u)) - |\mathrm{odd}(V_f(u))|$.

We now show that $f(e) = f(V_f(u))$. If $f(e) \in \{c(e), c(e)-1\}$, then we have $V_f(u) = \{u\}$ and $f(e) = f(V_f(u))$. Consider the case where $c(e) - f(e) \geq 2$. We claim that any positive-path P_{t_1,t_2} for $t_1, t_2 \in T_e$ is disjoint with $V_f(u)$. Assume indirectly that a positive-path P_{t_1,t_2} contains a vertex in $V_f(u)$. Let w be the branch vertex of $P_{t_1,u}$ and $P_{t_2,u}$. The function $f' := f + (P_{t_1,t_2}, -1) + (P_{t_1,u}, 1) + (P_{t_2,u}, 1)$ is a feasible flow in $I(e)$, since $V_f(u)$ is spanned with edges in $E[c-f; 2]$. Since $f'(e') = f(e')$ for all edges $e' \in E - E(P_{u,w})$, the cut-system $\mathcal{X}$ is blocked also by the flow f', and thereby f' is a blocking flow in $I(e)$ with $f'(e) > f(e) = \max\{x \in \Psi(e)\}$, which contradicts the definition of $\Psi(e)$. Hence any positive-path P_{t_1,t_2} with $t_1, t_2 \in T_e$ is disjoint with $V_f(u)$. This proves that $f(e) = f(V_f(u))$ even if $c(e) - f(e) \geq 2$. It always holds that $f(e) = f(V_f(u)) = c(V_f(u)) - |\mathrm{odd}(V_f(u))|$. Therefore we have $2\alpha(f) = f(e) + \sum_{t \in T_e} f(e_t) = \sum_{t \in T_e}(c(V_f(t)) - |\mathrm{odd}(V_f(t))|) + c(V_f(u)) - |\mathrm{odd}(V_f(u))| = \gamma(\mathcal{X}) - \kappa(\mathcal{X})$, as required. ◀

We prove that all edges $e \in E$ satisfies the following conditions (a) and (b) by an induction of depth of edges.

(a) $\Psi(e)$ is given by $\langle a(e), b(e)\rangle$ with some integers $a(e)$ and $b(e)$ such that
- **(i)** For each leaf-edge e, it holds $\Psi(e) = \langle a(e) = c(e), b(e) = c(e)\rangle$;
- **(ii)** For each non-leaf-edge e with two child-edges e_1 and e_2, it holds

$$\Psi(e) = \langle a(e), b(e)\rangle = ((\Psi(e_1) \otimes \Psi(e_2)) \cap [0, c(e)]) \cup \{c(e)\}.$$

That is, for $\langle \tilde{a}(e), \tilde{b}(e)\rangle = \Psi(e_1) \otimes \Psi(e_2)$, where $\tilde{b}(e) = b(e_1) + b(e_2)$ and

$$\tilde{a}(e) = \begin{cases} 0 & \text{if “}a(e_2) < b(e_1) \text{ or } a(e_1) < b(e_2)\text{” and } a(e_1) + a(e_2) \text{ is even,} \\ 1 & \text{if “}a(e_2) < b(e_1) \text{ or } a(e_1) < b(e_2)\text{” and } a(e_1) + a(e_2) \text{ is odd,} \\ a(e_i) - b(e_j) & \text{if } b(e_j) + 2 \leq a(e_i) \text{ with } \{i, j\} = \{1, 2\}, \end{cases} \tag{3}$$

where edge e_1 (resp., e_2) is called *dominating* if $b(e_2) + 2 \leq a(e_1)$ (resp., $b(e_1) + 2 \leq a(e_2)$), it holds that

$$\langle a(e), b(e)\rangle = \begin{cases} \langle \tilde{a}(e), \tilde{b}(e)\rangle & \text{if } \tilde{b}(e) \leq c(e), \\ \langle \tilde{a}(e), c(e)\rangle & \text{if } \tilde{a}(e) \leq c(e) < \tilde{b}(e) \text{ and } \tilde{a}(e) + c(e) \text{ is even,} \\ \langle \tilde{a}(e), c(e) - 1\rangle & \text{if } \tilde{a}(e) \leq c(e) < \tilde{b}(e) \text{ and } \tilde{a}(e) + c(e) \text{ is odd,} \\ \langle c(e), c(e)\rangle & \text{if } c(e) < \tilde{a}(e). \end{cases} \tag{4}$$

(b) If $e = (u, v)$ has a dominating child-edge $e' = (v, w)$, then there is a terminal $t \in T_{e'}$ such that $g(u, t) \geq a(e)$ holds for any decomposition g of a blocking flow f to $I(e)$ and $P_{v,t}$ consists of dominating edges.

A path consisting of dominating edges is called a *dominating path*. Fig. 3 shows the pairs $\{\tilde{a}(e), \tilde{b}(e)\}$ and $\{a(e), b(e)\}$ for all edges $e \in E$ in the instance I in Fig. 1 computed according to (3) and (4).

Assuming that each edge with depth at least d satisfies conditions (a) and (b), we prove that any edge e with depth $d - 1$ satisfies the statements in the next lemma, which indicates not only conditions (a) and (b) for the edge e but also how to construct a blocking flow in $I(e)$ from blocking flows in $I(e_1)$ and $I(e_2)$ of the child-edges e_1 and e_2 of e.

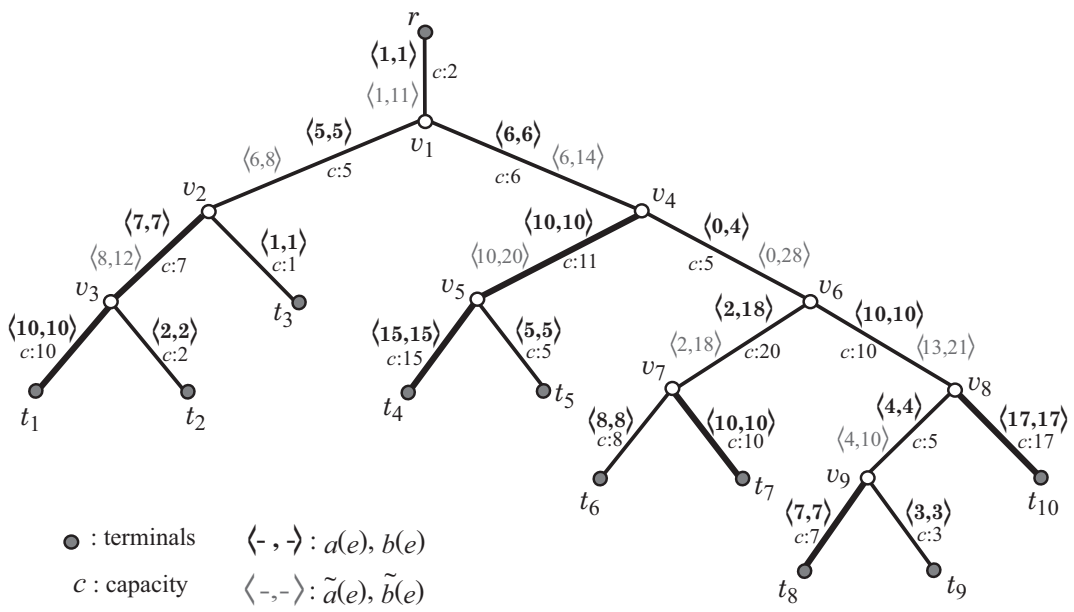

Figure 3 A pair of integers $\tilde{a}(e)$ and $\tilde{b}(e)$ in (3) and $\Psi(e) = \langle a(e), b(e)\rangle$ in (4) for each edge $e \in E$ in the instance I in Fig. 1, where each pair of $a(e)$ and $b(e)$ is depicted in bold while that of $\tilde{a}(e)$ and $\tilde{b}(e)$ in gray. The dominating edges are depicted in thick lines.

5 Main Lemma

This section provides a main technical lemma that tells how to compute the representation of flow values of blocking flows given by conditions (a) and (b), and how to construct a maximum flow from the representations.

▶ **Lemma 5.** *Let $e = (u, v)$ be a non-leaf-edge with depth $d - 1$ (≥ 1). Assume that all edges with depth at least d satisfy conditions* (a) *and* (b)*. For the two children w_1 and w_2 of v, let $\langle \tilde{a}, \tilde{b}\rangle = \Psi(vw_1) \otimes \Psi(vw_2) = \langle a(vw_1), b(vw_1)\rangle \otimes \langle a(vw_2), b(vw_2)\rangle$.*

- **(i)** *For a blocking flow of $I(e)$, if $e \in E(V_f(t))$ for some terminal $t \in T_e$, then the path $P_{v,t}$ from v to t is a dominating path, the path $P_{u,t}$ from u' to t satisfies $g(u, t) \geq c(e)$ for any decomposition g of a blocking flow of $I(e)$, and it holds $c(e) < \tilde{a}$.*
- **(ii)** *One of the child-edges of e is dominating if $c(e) < \tilde{a}$. Edge $e = (u, v)$ satisfies condition* (b)*; if vw_1 or vw_2, say vw_1 is dominating, then there is a terminal $t^* \in T_{vw_1}$ such that $g(u, t^*) \geq \min\{\tilde{a}, c(e)\}$ holds for any decomposition g of a blocking flow of $I(e)$ and P_{v,t^*} is a dominating path.*
- **(iii)** *For any integers x_1, x_2 and x such that $x_i \in \Psi(vw_i)$, $i = 1, 2$ and $x = x_1 + x_2 - 2y$ for some integer $y \in [0, \min\{x_1, x_2\}]$, let f_i, $i = 1, 2$ be a blocking flow of $I(vw_i)$ with $f_i(vw_i) = x_i$. Then $x \geq \tilde{a}$ holds. When $\tilde{a} \leq c(e)$, any function $f = (x, f_1, f_2)$ with $x \leq c(e)$ is a blocking flow of $I(e)$.*
- **(iv)** *If $I(e)$ admits a blocking flow f with $f(e) < c(e)$, then $f(e) \in \langle \tilde{a}, \tilde{b}\rangle$.*
- **(v)** *Assume that $c(e) < \tilde{a}$ and vw_1 is dominating. Let P_{v,t^*} be the dominating path in* (iii) *and let $\delta_e = \tilde{a} - c(e)$. There is a blocking flow f of $I(e)$ with $f(e) = c(e)$, which can be constructed as*

$$f = (c(e), f_1 + (P_{v,t^*}, -\delta_e), f_2)$$

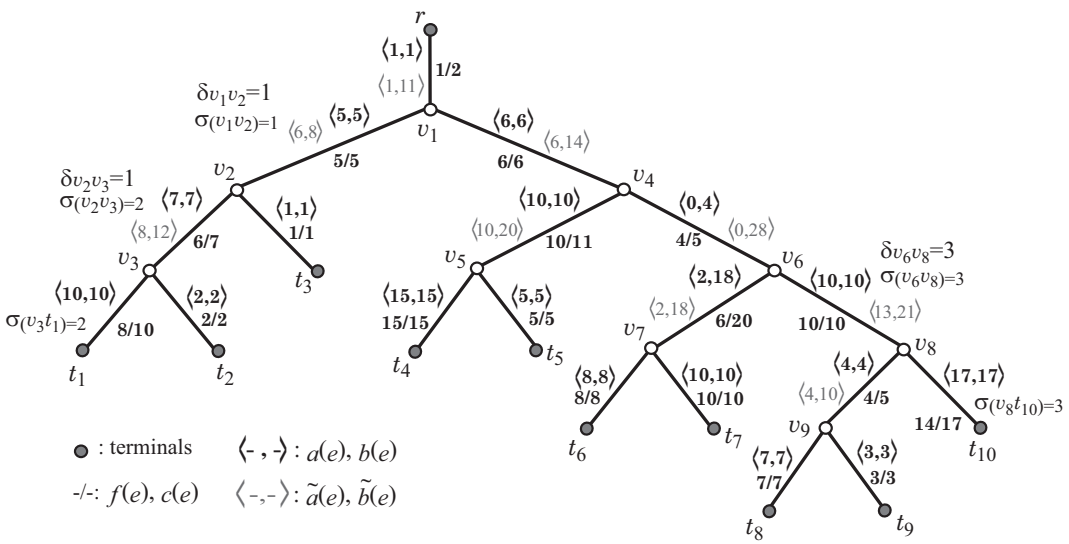

Figure 4 A blocking flow f with $f(e_r) = b(e_r)$ in the instance I in Fig. 1 such that $2\alpha(f) = \sum_{t \in T} f(e_t) = 1 + 8 + 2 + 1 + 15 + 5 + 8 + 10 + 7 + 3 + 14 = 74$, where the pair of flow value $f(e)$ and capacity $c(e)$ for each edge is indicated by f/c beside the line segment for edge e. The non-zero values for δ_e and $\sigma(e)$ are indicated beside the corresponding edge e.

by choosing a blocking flow f_1 of $I(vw_1)$ with $f_1(vw_1) = a(vw_1)$ and a blocking flow f_2 of $I(vw_2)$ with $f_2(vw_2) = b(vw_2)$.

(vi) *Edge $e = (u, v)$ satisfies condition* (a)*; i.e.,* $\Psi(e) = (\langle \tilde{a}, \tilde{b} \rangle \cap [0, c(e)]) \cup \{c(e)\}$.

A proof of this lemma can be found in the full version of this paper.

6 Algorithm Description

Based on Lemma 5, this section gives a description of a linear-time algorithm for computing the representations of flow values of blocking flows and constructing a maximum flow from the representations.

By Lemma 5(ii) and (iv), we see by induction that every edge in E satisfies conditions (a) and (b). By Lemma 5(iii) and (v), we know how to construct a blocking flow in $I(e)$ for some edge e from blocking flows in $I(e_1)$ and $I(e_2)$ of the child-edges e_1 and e_2 of e. By Lemma 4, it suffices to construct a blocking flow in $I = I(e_r)$ with $f(e_r) = b(e_r)$. For this, we first compute the integers $\tilde{a}(e)$, $\tilde{b}(e)$, $a(e)$ and $b(e)$ for each edge $e \in E$ according to (3) and (4) selecting edges in E in a non-increasing order of depth, and identify all the dominating edges in E. Next we apply Lemma 5(iii) and (v) repeatedly from edge e_r to descendants of the edge in a top-down manner to construct a blocking flow in $I = I(e_r)$ with $f(e_r) = b(e_r)$. To implement the algorithm to run in linear time, we avoid reducing flow values repeatedly along part of a dominating path. We let $\sigma(e)$ to store the total amount of decrements over each dominating edge e, i.e., $\sigma(e)$ is the summation of $\delta_{e'}$ in Lemma 5(v) over all dominating edges e' that are ancestors of e. An entire algorithm is given by the following compact and succinct description.

The algorithm runs in linear time, because it executes an $O(1)$-time procedure to each edge in E in constant time. Fig. 4 illustrates a result obtained from the instance I in Fig. 1 by applying the algorithm.

Algorithm 1 BlockFlow

Input: An instance $I = (G = (V, E), T, c)$ rooted at a terminal $r \in T$.
Output: A maximum flow f in I.

```
Compute the integers ã(e), b̃(e), a(e) and b(e) for each edge e ∈ E according to (3) and
(4) selecting edges in E in a non-increasing order of depth;
x(e_r) := b(e_r); σ(e_r) := 0;
for each edge e ∈ E selected in a non-decreasing order of depth do
  f(e) := x(e) − σ(e);
  if e is not a leaf edge then
    /* Denote by e_1 and e_2 the child-edges of e */
    if ã(e) ≤ c(e) then
      Choose integers x_1 ∈ ⟨a(e_1), b(e_1)⟩ and x_2 ∈ ⟨a(e_2), b(e_2)⟩ such that
      x(e) = x_1 + x_2 − 2y for some integer and y ∈ [0, min{x_1, x_2}];
      x(e_1) = x_1;  x(e_2) = x_2;
      if e_i is dominating for i = 1 or 2 then
        σ(e_i) := σ(e) and σ(e_j) := 0 for j ∈ {1, 2} − {i}
      else
        σ(e_1) := σ(e_2) := 0
      end if
    else
      /* c(e_0) < ã(e_0), where e_0 is dominating, and exactly one of e_1 and e_2 is
      dominating; assume that e_1 is dominating without loss of generality. */
      x(e_1) = a(e_1);  x(e_2) = b(e_2);  δ_{e_1} := a(e_1) − c(e);
      σ(e_1) := σ(e) + δ_{e_1};  σ(e_2) := 0
    end if
  end if
end for
```

After a maximum flow f is constructed, a minimum cut-system $\mathcal{X}$ to a given instance can be constructed in linear time by Lemma 4. Fig. 5 illustrates the cut-system $\mathcal{X} = \{V_f(t) \mid t \in T\}$ for the blocking flow f in Fig. 4, which indicates that the flow f is maximum because $2\alpha(f) = \sum_{t \in T} f(e_t) = 74 = \gamma(\mathcal{X}) - \kappa(\mathcal{X})$ holds.

From the above argument, the next theorem is established.

▶ **Theorem 6.** *Given a tree instance (G, T, c), a feasible integral multiflow f and a cut-system $\mathcal{X}$ with $\alpha(f) = (\gamma(\mathcal{X}) - \kappa(\mathcal{X}))/2$ can be found in $O(n)$ time and space, where f is a maximum integral multiflow.*

7 Concluding Remarks

In this paper, we revealed a recursive formula among flow values of blocking flows in rooted instances and designed a linear-time dynamic programming algorithm for computing a maximum integral flow in a tree instance. The optimality of flows is ensured by the property of the formula, by which we can always construct the corresponding dual object, i.e., a minimum cut-system that satisfies (2) by equality.

It would be interesting to characterize similar recursive properties and design fast algorithms for the maximum integral multiterminal flows in more general classes of graphs.

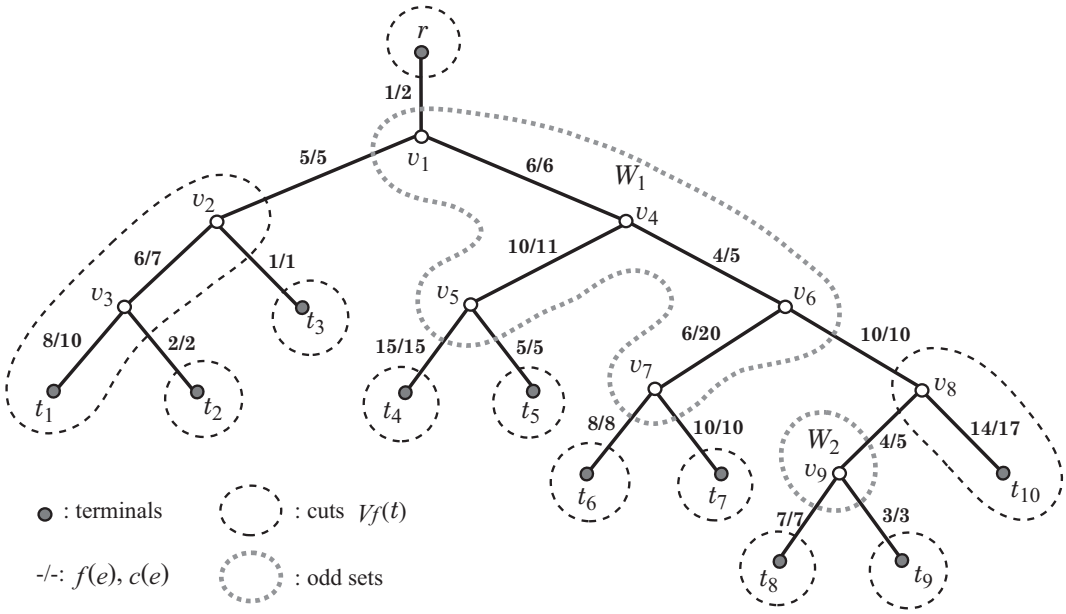

Figure 5 The cut-system $\mathcal{X} = \{V_f(t) \mid t \in T\}$ for the blocking flow f in Fig. 4, where the set $V - \cup_{X \in \mathcal{X}} X$ induces from G two odd sets $W_1 \in \mathrm{odd}(V_f(r))$ and $W_2 \in \mathrm{odd}(V_f(t_{10}))$, and it holds that $\gamma(\mathcal{X}) - \kappa(\mathcal{X}) = \sum_{t \in T} c(V_f(t)) - 2 = 2 + (2+1+5) + 2 + 1 + 15 + 5 + 8 + 10 + 7 + 3 + (5+10) - 2 = 74$.

References

1. B. V. Cherkasskii. Reshenie odnoi zadachi o mnogoproduktovykh potokakh v seti [russian; a solution of a problem of multicommodity flows in a network]. *Èkonomika i Matematicheskie Metody*, 13(1):143–151, 1977.
2. Marie-Christine Costa, Lucas Létocart, and Frédéric Roupin. Minimal multicut and maximal integer multiflow: a survey. *European Journal of Operational Research*, 162(1):55–69, 2005.
3. Mariechristine Costa and Alain Billionnet. Multiway cut and integer flow problems in trees. *Electronic Notes in Discrete Mathematics*, 17(20):105–109, 2004.
4. William H. Cunningham. The optimal multiterminal cut problem. *DIMACS series in discrete mathematics and theoretical computer science*, 5:105–120, 1991.
5. J. R. Ford and D. R. Fulkerson. *Flows in networks.* Princeton university press, 1962.
6. Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.
7. Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.
8. Torben Hagerup, Jyrki Katajainen, Naomi Nishimura, and Prabhakar Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *Journal of Computer and System Sciences*, 57(3):366–375, 1998.
9. Toshihide Ibaraki, Alexander V. Karzanov, and Hiroshi Nagamochi. A fast algorithm for finding a maximum free multiflow in an inner eulerian network and some generalizations. *Combinatorica*, 18(1):61–83, 1998.
10. Wolfgang Mader. Über die Maximalzahl kantendisjunkter A-Wege. *Archiv der Mathematik*, 30(1):325–336, 1978.