An Efficient Scaling Algorithm for the Minimum Weight Bibranching Problem

Maxim A. Babenko*

Moscow State University
max@adde.math.msu.su

Abstract. Let $G = (VG, AG)$ be a digraph and let $S \sqcup T$ be a bipartition of VG. A *bibranching* is a subset $B \subseteq AG$ such that for each node $s \in S$ there exists a directed s–T path in B and, vice versa, for each node $t \in T$ there exists a directed S–t path in B.

Bibranchings generalize both branchings and bipartite edge covers. Keijsper and Pendavingh proposed a strongly polynomial primal-dual algorithm that finds a minimum weight bibranching in $O(n'(m+n\log n))$ time (where $n := |VG|$, $m := |AG|$, $n' := \min(|S|, |T|)$).

In this paper we develop a weight-scaling $O(m\sqrt{n}\ \log n \log(nW))$-time algorithm for the minimum weight bibranching problem (where W denotes the maximum magnitude of arc weights).

1 Introduction

In a directed graph G, the sets of nodes and arcs are denoted by VG and AG, respectively. A similar notation is used for paths, cycles, and etc.

Consider a digraph G and a fixed bipartition $S \sqcup T$ of VG. A subset $B \subseteq AG$ is called a *bibranching* if the following conditions are met:

- for each $s \in S$, set B contains a directed path from s to some node in T;
- for each $t \in T$, set B contains a directed path from some node in S to t.

Bibranchings were introduced in [Sch82]. In the present paper we study the *minimum weight bibranching problem*, which is formulated as follows:

(BB) *Given G, S, T, and* arc weights $w\colon AG \to \mathbb{R}_+$, *find a bibranching B whose weight $w(B)$ is minimum.*

Hereinafter we assume that every real-valued function $f\colon U \to \mathbb{R}$ is additively extended to the family of subsets of U (denoted by 2^U) by $f(A) := \sum_{a\in A} f(a)$. In particular, $w(B)$ denotes the sum of weights of arcs in B.

Minimum weight bibranchings provide a common generalization to a pair of well-known combinatorial problems. Firstly, if $S = \{s\}$ then (BB) asks for a minimum weight *s-branching* (a directed tree rooted at s that covers all nodes of G). An $O(m + n\log n)$ algorithm for the latter task is known [GGSR86]

* Supported by RFBR grants 05-01-02803 and 06-01-00122.

S.-H. Hong, H. Nagamochi, and T. Fukunaga (Eds.): ISAAC 2008, LNCS 5369, pp. 232–245, 2008.

(here $n := |VG|$ and $m := |AG|$; we are assuming throughout the paper that $n \leq m \leq n^2$). Another special case arises when graph G only contains S–T arcs (but no T–S, S–S, or T–T arcs). Then the definition of a bibranching reduces to that of a *bipartite edge cover*. The minimum weight bipartite edge cover problem (call it (EC) for brevity) seems to be harder: no strongly polynomial $o(mn)$-algorithm is known so far. Problem (EC) can be solved by finding a maximum weight bipartite matching in $O(n'(m + n \log n))$ time [FT87] (where $n' := \min(|S|, |T|)$). Keijsper and Pendavingh [KP98] generalized the latter shortest-path augmentation method to solve (BB) in the same time bound.

On the other hand, many optimization problems may be solved faster if weights are known to be integral. The corresponding algorithms are based on scaling technique and achieve time bounds that are, in a sense, better than of their strongly-polynomial counterparts. A typical example is a scaling algorithm for bipartite matching problems [GT89], which runs in $O(m\sqrt{n}\ \log(nW))$ time (hereinafter W denotes the maximum magnitude of weights). The latter approach can also be adopted to solve (EC) and leads to an algorithm with the same time bound.

Similar ideas are also applicable to general min-cost flow problems [GT87]. However, when the structure of dual solutions becomes non-trivial (i.e. when one needs exponentially many dual variables) the algorithm and its complexity analysis may become much more involved. A good example is the minimum weight perfect matching problem in general (non necessarily bipartite) graphs, which is solved by Gabow and Tarjan in $O(m\sqrt{n\alpha(m,n)\log n}\ \log(nW))$ time [GT91].

Since (BB) involves solving a linear program with inequalities corresponding to all possible subsets of S and T, our approach is of no exception. We present an $O(m\sqrt{n}\ \log n \log(nW))$-time weight scaling algorithm for (BB). It is based on the general notion of ε-optimality (see, e.g., [GT87]), the augmentation procedure from [KP98], and attracts some additional combinatorial ideas to deal with dual solutions during scaling steps. Also, a variant of the blocking augmentation technique [Din80] is employed.

Note that the complexity of our algorithm coincides (up to a logarithmic factor) with that of the best known scaling algorithm [GT89] for solving a special case of (BB), namely (EC).

The rest of the paper is organized as follows. Section 2 gives the needed formal background and introduces the linear programming formulation of (BB). Section 3 and Section 4 outline a high-level description of the algorithm. Section 5 bounds the number of primal and dual steps performed by the algorithm. Due to lack of space some technical proofs and implementation details are omitted and will appear in the full version of the paper.

2 Preliminaries

First, we need some additional definitions and notation. Let G be a digraph and X be a subset of nodes. Then $\delta^{\text{in}}_G(X)$ (resp. $\delta^{\text{out}}_G(X)$ and $\gamma_G(X)$) denotes the set of arcs entering X (resp. leaving X and having both endpoints in X). When it is

clear from the context which graph G is meant, it is omitted from the notation. Also, when $X = \{v\}$ we drop curly braces and write just $\delta^{\text{in}}(v)$ and $\delta^{\text{out}}(v)$.

For a graph G and a subset $X \subseteq VG$ let $G[X]$ denote the subgraph of G induced by X (i.e. the graph obtained from G by removing all nodes in $VG - X$).

Consider a bipartition $S \sqcup T$ of VG. For a subset $X \subseteq S$ we put $\delta(X) := \delta^{\text{out}}(X)$, similarly for $X \subseteq T$ we put $\delta(X) := \delta^{\text{in}}(X)$. If $a \in \delta(X)$ then arc a is said to *cover* X. For a set $A \subseteq AG$, the *set of nodes covered by* A is defined as the union of the sets of nodes covered by the individual elements of A. Clearly, (BB) prompts for a minimum weight subset $B \subseteq AG$ that covers every subset in $2^S \cup 2^T$.

Let us introduce an important notion of *graph contraction*. To *contract* a set $U \subseteq VH$ in a graph H means to replace nodes in U by a single *complex node* (also denoted by U). Arcs in $\gamma(VH - U)$ are not affected, arcs in $\gamma(U)$ are dropped, and arcs in $\delta^{\text{in}}(U)$ (resp. $\delta^{\text{out}}(U)$) are redirected so as to enter (resp. leave) the complex node U. The resulting graph is denoted by H/U. Note that this graph may contain multiple parallel arcs. We identify arcs in H/U with their pre-images in H. If H' is obtained from H by an arbitrary sequence of contractions, then $H' = H/U_1/\ldots/U_k$ for a certain family of disjoint subsets $U_1, \ldots, U_k \subseteq VH$ (called the *maximal contracted sets*). Each node in H' corresponds to a subset of nodes in H: nodes $v \in VH - (U_1 \cup \ldots \cup U_k)$ correspond to singletons $\{v\}$, complex nodes correspond to sets U_i.

For a set $A \subseteq AG$, we shall write A_{SS}, A_{ST}, and A_{TT} to denote the sets of S–S, S–T, and T–T arcs in A, respectively. Note that any minimum weight bibranching need not contain T–S arcs (as their removal preserves the required connectivity and can only decrease the total weight). Hence, we shall assume that graph G contains no T–S arcs.

We call a bibranching B *canonical* if every its proper subset $B' \subset B$ is not a bibranching. The following observations are easy:

Fact 1. *For each bibranching B there exists and can be found in $O(m)$ time a canonical bibranching $B' \subseteq B$.*

Fact 2. *Any canonical bibranching contains at most n arcs.*

Consider the following linear program:

$$\text{(P)} \qquad \begin{array}{ll} \textbf{minimize} & \sum (w(a)x(a) : a \in AG) \\ \textbf{subject to} & x\colon AG \to \mathbb{R}_+, \\ & x(\delta(X)) \geq 1 \quad \text{for each } X \in 2^S \cup 2^T. \end{array}$$

The program dual of (P) is:

$$\text{(D)} \qquad \begin{array}{ll} \textbf{maximize} & \sum \left(\pi(X) : X \in 2^S \cup 2^T\right) \\ \textbf{subject to} & \pi\colon 2^S \cup 2^T \to \mathbb{R}_+, \\ & w_\pi(a) \geq 0 \quad \text{for each } a \in AG. \end{array}$$

Here $w_\pi := w - \vartheta\pi$ are the *reduced weights* of arcs w.r.t. π, and the function $\vartheta\pi\colon AG \to \mathbb{R}_+$ is defined by

$$\vartheta\pi(a) := \sum (\pi(X) : a \text{ covers } X)\,.$$

It is known [Sch03] that (P) describes the upper convex hull of the incidence vectors of all bibranchings in G. Hence, finding a bibranching of the minimum weight (under the assumption $w \geq 0$) amounts to finding a 0,1-solution to (P).

Weak duality for (P) and (D) implies that $\sum_a w(a)x(a) \geq \sum_X \pi(X)$ holds for every pair of admissible solutions x and π. By strong duality, the latter turns into equality if x and π are optimal. Moreover, (P) is known [Sch03] to be *totally dual integral*, that is, if all weights w are integers then there exists an optimal dual solution π that is integer. Hence, the polyhedron determined by (P) is integral.

The complementary slackness conditions for (P) and (D) (giving an optimality criterion for solutions x and π to these programs) are viewed as:

(1) if $x(a) > 0$ for $a \in AG$ then $w_\pi(a) = 0$;

(2) if $\pi(X) > 0$ for $X \in 2^S \cup 2^T$ then $x(\delta(X)) = 1$.

For a set $B \subseteq AG$ and a function $\pi\colon 2^S \cup 2^T \to \mathbb{R}_+$ we say that B is π-*consistent* if $\pi(X) > 0$ implies $|B \cap \delta(X)| \leq 1$ for each $X \in 2^S \cup 2^T$. Consistency is closely related to the complementary slackness conditions, in particular, if B is a bibranching then its π-consistency is just (2).

Fact 3. *For an arbitrary function $\pi\colon 2^S \cup 2^T \to \mathbb{R}_+$ and a set $B \subseteq AG$ one has*

$$\vartheta\pi(B) \geq \sum (\pi(X) : B \text{ covers } X)\,.$$

Additionally, if B is π-consistent then the above inequality turns into equality.

3 Algorithm

Recall that a family $\mathcal{F}$ of subsets is called *laminar* if for any $X, Y \in \mathcal{F}$ one either has $X \subseteq Y$, or $Y \subseteq X$, or $X \cap Y = \emptyset$. Also, let $\operatorname{supp}(f)$ denote the *support set* of a function f, i.e. $\{x \mid f(x) \neq 0\}$.

The algorithm maintains a laminar family $\mathcal{F} \subseteq 2^S \cup 2^T$ and a function $\pi\colon 2^S \cup 2^T \to \mathbb{Z}_+$. For $X \in \mathcal{F}$, the *shell* $S(X)$ of X is the graph obtained from $G[X]$ by contracting all proper maximal subsets $Y \subset X$, $Y \in \mathcal{F}$. Let $\overline{G}$ denote the graph obtained by contracting all maximal sets of $\mathcal{F}$ in G. Put $\overline{S}$ (resp. $\overline{T}$) to be the image of S (resp. T) in $\overline{G}$ under these contractions. Let $S^0_\pi(X)$ denote the subgraph of $S(X)$ consisting of arcs a with $w_\pi(a) = 0$.

For a set of nodes Y in $\overline{G}$ (or in $S(X)$ for $X \in \mathcal{F}$) we write $\widetilde{Y}$ to denote the corresponding pre-image subset in VG. When $Y = \{y\}$ we write just $\widetilde{y}$ instead of $\widetilde{\{y\}}$.

We introduce the following set of properties:

(D1) $\mathcal{F}$ is laminar and $\operatorname{supp}(\pi) \subseteq \mathcal{F}$;
(D2) $S^0_\pi(X)$ is strongly connected for each $X \in \mathcal{F}$;
(D3) $w_\pi(a) \geq 0$ for all $a \in AG$.

Property (D3) is just the feasibility of a dual solution π while (D1) and (D2) introduce some additional structural requirements for π.

Next, the algorithm maintains a subset $\overline{B} \subseteq A\overline{G}$, which will be referred to as a *partial bibranching*. Consider the following properties:

(P1) set $\overline{B}_{SS}$ (resp. $\overline{B}_{TT}$) forms a directed out- (resp. in-) forest in $G[\overline{S}]$ (resp. in $G[\overline{T}]$), each arc in $\overline{B}_{ST}$ covers a pair of roots of the said forests;
(P2) if $a \in \overline{B}_{SS} \cup \overline{B}_{TT}$ then $w_\pi(a) = 0$; if $a \in \overline{B}_{ST}$ then $w_\pi(a) \le 1$;
(P3) if $v \in V\overline{G}$ is covered by more than one arc in $\overline{B}$ then $\pi(\widetilde{v}) = 0$ (c.f. (2)).

Here by an *in-* (resp. *out-*) *forest* we mean an acyclic set of arcs X such that for each node v at most one arc in X enters (resp. leaves) v. If no arc in X enters (resp. leaves) node v then v is said to be a *root* of X.

Property (P1) is required mostly by technical reasons that will become evident later. Property (P2) may be regarded as a relaxation of (1) and (P3) directly corresponds to (2).

The algorithm employs *bit scaling* and works as follows. Let $w_0 \colon AG \to \mathbb{Z}_+$ denote the input weight function. The algorithm starts with $w := 0$, $\pi := 0$, $\mathcal{F} := \emptyset$ and puts $\overline{B}$ to be an arbitrary bibranching in G obeying (P1) (the latter can be found in $O(m)$ time by an obvious routine). In case $\overline{B}$ does not exist, the algorithm halts.

Each scaling step takes a weight function w from the previous iteration, a bibranching $\overline{B} \subseteq A\overline{G}$ in $\overline{G}$, a function π, and a collection $\mathcal{F}$ altogether obeying properties (D1)–(D3), (P1)–(P3). The weights $w(a)$ are doubled and some of them are increased by 1 (namely, those having 1 at the corresponding position of the binary representation of $w_0(a)$). Changing weights w may lead to violation of the above properties so the goal of the scaling step is to restore them. The necessary details will be given in Section 4.

Lemma 1. *Suppose that the current scaling step is complete, so properties (D1)–(D3), (P1)–(P3) hold for $\mathcal{F}$, π, and $\overline{B}$. Also, let $\overline{B}$ be a bibranching in $\overline{G}$. Then there exists a canonical π-consistent bibranching B in G such that: (i) $w(B) \le w(\overline{B})$, (ii) $w_\pi(a) = 0$ for each $a \in B_{SS} \cup B_{TT}$; and (iii) $w_\pi(a) \le 1$ for each $a \in B_{ST}$.*

Proof. Firstly, with the help of Fact 1 set $\overline{B}$ is turned into a canonical bibranching in $\overline{G}$ by removing some arcs. Next, let $X \in \mathcal{F}$ be one of the maximal contracted sets in $V\overline{G}$. We describe procedure EXPAND(X) that extends $\overline{B}$ into the shell $S(X)$. For simplicity's sake suppose $X \subseteq S$, the other case is symmetric. Remove X from $\mathcal{F}$ thus partly uncontracting graph $\overline{G}$. Let X' denote the set of nodes in new $\overline{G}$ arising from X during this uncontraction. Let R denote the set of nodes in X' that are covered by arcs in $\overline{B}$. One has $R \ne \emptyset$ since $\overline{B}$ was a bibranching before set X got removed from $\mathcal{F}$. We grow an out-forest F in the subgraph $\overline{G}[X']$ such that: (i) R is the set of roots of F; (ii) $VF = X'$; (iii) $w_\pi(a) = 0$ holds for each $a \in AF$. Property (D2) shows that this forest always exists. Now we add the arcs of F to $\overline{B}$. Clearly, the new set $\overline{B}$ is a canonical bibranching in $\overline{G}$.

Applying EXPAND to the elements of $\mathcal{F}$ (in an appropriate order) one gets a bibranching B in G that obeys the required properties.

Weights w are iteratively scaled, as explained above, until achieving the equality $w = w_0$. Totally it takes $\lceil \log W \rceil$ scaling steps. Next, we put $t := \lfloor \log n \rfloor + 1$ and perform t additional scaling steps, doubling w each time. Finally, the algorithm applies Lemma 1 to construct the final bibranching in G.

Let us prove that this general scheme is correct. Put $\Pi := \sum_X \pi(X)$ and estimate the weight of an optimal bibranching as follows.

Lemma 2. *Property (D3) implies $w(B) \geq \Pi$ for any bibranching B in G.*

Proof. By definition B covers each subset in $2^S \cup 2^T$. Hence, by Fact 3 and (D3) one has $w(B) = w_\pi(B) + \vartheta\pi(B) \geq \sum_X \pi(X) = \Pi$, as required.

Lemma 3. *If B is a canonical bibranching in G and properties (P2) and (P3) hold (for $\overline{G} := G$, $\overline{B} := B$) then $w(B) \leq \Pi + n$.*

Proof. By Fact 2, Fact 3, and property (P2) one has $w(B) = w_\pi(B) + \vartheta\pi(B) \leq n + \sum_X \pi(X) = n + \Pi$.

Theorem 4. *The algorithm constructs a minimum weight bibranching.*

Proof. Let $\overline{B}$, w, and π denote the corresponding objects after the last scaling step. Put B to be a canonical bibranching obtained from $\overline{B}$ by Lemma 1. Let B_{min} be a minimum weight bibranching (w.r.t. w or, equivalently, w_0). One may assume by Fact 1 that B_{min} is canonical. Lemma 2 and Lemma 3 imply that $w(B_{min}) \geq \Pi$ and $w(B) \leq \Pi + n$, so $w(B_{min}) \leq w(B) \leq w(B_{min}) + n$. Recall, each of the last t scaling steps doubles arc weights. Since all initial weights w_0 are integers, $w(a)$ is divisible by 2^t for each $a \in AG$. Hence, so is $w(B)$. The choice of t implies $n < 2^t$, therefore $w(B) = w(B_{min})$, so B is optimal.

4 Scaling Step

Each scaling step consists of the following four stages: doubling stage, shell stage, ST stage, and TS stage.

First, the **doubling stage** is executed: arc weights w are multiplied by 2 and some of them are increased by 1, as described in Section 3. Also, duals π are doubled. Put $\mathcal{F} := \operatorname{supp}(\pi)$ and $\overline{B} := \emptyset$ (hence, any previous bibranching is discarded). As earlier, let $\overline{G}$ denote the graph obtained from G by contracting all maximal sets in $\mathcal{F}$. Obviously, properties (D1), (D3), (P1)–(P3) now hold. One needs to solve the following two tasks:

- restore property (D2) by ensuring that graphs $S^0_\pi(X)$, $X \in \mathcal{F}$, are strongly connected;
- construct a bibranching $\overline{B}$ in $\overline{G}$ obeying properties (P1)–(P3).

The **shell stage** is executed to deal with (D2). The algorithm scans the sets in $\mathcal{F}$ choosing an inclusion-wise minimal unscanned set at each iteration. Let $X \in \mathcal{F}$ be the current set to be scanned. Procedure NORMALIZE-SHELL(X) is called to adjust duals π and ensure (D2) for X or remove X from $\operatorname{supp}(\pi)$ (and hence also from $\mathcal{F}$).

Suppose $X \subseteq S$, the case $X \subseteq T$ is analogous. NORMALIZE-SHELL performs a series of iterations similarly to the minimum weight branching algorithm [Edm67, GGSR86].

More precisely, it maintains a directed out-forest F_S containing all nodes of X and consisting of some arcs a with $w_\pi(a) = 0$. Initially $VF_S := VS(X)$ and $AF_S := \emptyset$. If $S(X)$ is a single node graph then property (D2) is restored for X, NORMALIZE-SHELL(X) terminates. Otherwise, an arbitrary tree W in F_S is picked. Let r be the root of W. Suppose that all arcs leaving r (in $S(X)$) have positive reduced weights. Put

$$\mu_1 := \min\left(w_\pi(a) : a \in \delta^{\text{out}}_{S(X)}(r)\right), \quad \mu_2 := \pi(X), \quad \mu := \min(\mu_1, \mu_2).$$

Adjust the duals as follows:

$$\begin{aligned} \pi(\widetilde{r}) &:= \pi(\widetilde{r}) + \mu, \\ \pi(X) &:= \pi(X) - \mu. \end{aligned}$$

These adjustments decrease the reduced weights of all arcs in $S(X)$ leaving r by μ. Also, $\widetilde{r}$ is added to $\operatorname{supp}(\pi)$. By the choice of unscanned sets for NORMALIZE-SHELL, property (D2) holds for $\widetilde{r}$ and all its subsets in $\mathcal{F}$. Set $\widetilde{r}$ is also marked as scanned, so NORMALIZE-SHELL is never called for it.

If $\pi(X) = 0$ holds after the adjustment then set X vanishes from $\operatorname{supp}(\pi)$, we remove X from $\mathcal{F}$ and halt NORMALIZE-SHELL(X). We also say that set X *dissolves* during the execution of the shell stage.

Now suppose that there is an arc $a \in \delta^{\text{out}}_{S(X)}(r)$ such that $w_\pi(a) = 0$. Two cases are possible. Firstly, a may connect W with another tree W' in F_S. Then, a is added to F_S thus linking W and W'. Secondly, a may connect r to a node in the very same tree W. In this case, a cycle of arcs with zero reduced weights is discovered. Let Y be the set of nodes of this cycle (in $S(X)$). Algorithm contracts Y in $S(X)$, adds $\widetilde{Y}$ to $\mathcal{F}$, and proceeds to the next iteration.

Once NORMALIZE-SHELL(X) is called for all sets $X \in \mathcal{F}$ (in an appropriate order), property (D2) gets restored. Normalization procedure for a subset $X \subseteq T$ is the same except for it considers sets δ^{out} rather than δ^{in}.

The remaining part of the scaling step builds a bibranching $\overline{B}$ in $\overline{G}$ that satisfies properties (P1)–(P3). Firstly, the **ST stage** is executed. It starts with $\overline{B} = 0$ and applies a certain augmenting path approach aiming to update $\overline{B}$ so that it covers all subsets in $2^{\overline{S}}$. Next, S and T parts are exchanged and a similar **TS stage** is executed, thus completing the scaling step. We shall only describe the ST stage since the TS stage is essentially symmetric.

Similarly to the shell stage, a directed out-forest F_S obeying $VF_S = \overline{S}$ is maintained in graph $\overline{G}$. The latter forest satisfies the following conditions:

(F1) $w_\pi(a) = 0$ holds for all $a \in AF_S$;
(F2) $w_\pi(a) > 0$ holds for each root node $r \in \overline{S}$ and an $\overline{S}$–$\overline{S}$ arc a leaving r;
(F3) $\overline{B}_{SS} \subseteq AF_S$;
(F4) if a node $v \in \overline{S}$ is not covered by $\overline{B}$ then v is a root of F_S.

Forest F_S is initially constructed by putting $VF_S := \overline{S}$ and $AF_S := \overline{B}_{SS}$ (the latter set forms a directed out-forest according to (P1)). Next, NORMALIZE-FOREST routine is applied to ensure (F2) and (F4). The latter works as follows. If (F2) fails for a root node r and an $\overline{S}$–$\overline{S}$ arc a leaving r then two cases are possible. Firstly, a may connect a tree W of F_S rooted at r with another tree W' in F_S. Then, a is added to F_S thus linking W and W'. Secondly, a may connect r to a node in the very same tree W. In this case, a cycle consisting of arcs with zero reduced weights is discovered. Let Y denote the set of nodes of this cycle (in $\overline{G}$). The algorithm puts $\overline{B} := \overline{B} \setminus \gamma(Y)$, $AF_S := AF_S \setminus \gamma(Y)$, contracts Y in $\overline{G}$, and adds $\widetilde{Y}$ to $\mathcal{F}$. Note that at this point $\pi(\widetilde{Y}) = 0$ holds.

Next, if (F4) fails for a node $v \in \overline{S}$, then v is not a root of F_S and v is not covered by $\overline{B}$. To fix this, an arc $a \in AF_S$ that covers v is fetched and added to $\overline{B}$. Property (F1) implies the validity of (P2). Also, v is covered by exactly one arc in $\overline{B}$ after the augmentation, so (P3) holds.

This completes the description of NORMALIZE-FOREST.

Once forest F_S obeying (F1)–(F4) is constructed, the algorithm builds an auxiliary digraph H. Put $VH := V\overline{G}$ and proceed as follows:

- if $a \notin \overline{B}$ is an $\overline{S}$–$\overline{T}$ arc with $w_\pi(a) = 0$ then add a to H (these arcs are called *forward*);
- if $a \in \overline{B}$ is an $\overline{S}$–$\overline{T}$ arc with $w_\pi(a) = 1$ then add a to H but change its direction to the opposite (these arcs are called *backward*).

A node $v \in \overline{S}$ is said to be *initial* if $\overline{B}$ does not cover v. By (F4) only root nodes of F_S may be initial. A node v is called *final* if any of the following cases applies:

1. $v \in \overline{T}$ and $\pi(\widetilde{v}) = 0$;
2. $v \in \overline{T}$ and v is covered by a $\overline{T}$–$\overline{T}$ arc of $\overline{B}$ (the latter is unique by (P1));
3. $v \in \overline{T}$ and v is not covered by $\overline{B}$;
4. $v \in \overline{S}$ and v is an inner node of F_S;
5. $v \in \overline{S}$ and v is covered by at least two $\overline{S}$–$\overline{T}$ arcs of $\overline{B}$.

A path in H connecting an initial node to a final node (with all intermediate nodes neither initial nor final) is called *augmenting*. Suppose for a moment that there exists an augmenting path P in H from an initial node s to a final node t. In this case, a **primal step** is possible. We construct a set $A(P) \subseteq AH$ as follows. First, we add arcs that correspond to arcs of P (both forward and backward). Second, we perform adjustments to account for the type of node t. In cases (1), (3), and (5) we do nothing. In case (2) we add to $A(P)$ the unique arc in $\overline{B}_{TT}$ that covers t. In case (4) we add to $A(P)$ the unique arc in F_S that leaves t.

The *augmentation* of $\overline{B}$ along P is performed by putting $\overline{B} := \overline{B} \triangle A(P)$ (here $\triangle$ denotes the symmetric difference).

Lemma 4. *The augmentation of $\overline{B}$ along P preserves properties (P1)–(P3), (D1)–(D3), and (F1)–(F4). The set of nodes covered in $V\overline{G}$ by $\overline{B}$ strictly increases. The set of initial nodes strictly decreases. The set of final nodes does not increase. The arc set of H decreases by AP.*

The proof is carried out by a straightforward case-splitting, so we leave details to the reader.

Note, that in order to achieve the desired time bound one cannot recompute path P from scratch each time. Taking into account the monotonicity of the sets of initial and final nodes, and the arc set of H, the *blocking augmentation* technique is applied (see, e.g., [Din80]). The latter computes, one by one, a sequence of augmenting paths and stops when there are no such paths left. The total running time of this routine is $O(m)$. Each of these paths is used for augmenting $\overline{B}$, as explained above.

A **dual step** is carried out when no more augmenting paths can be found. Let $\overline{S}_0$ (resp. $\overline{T}_0$) denote the set of nodes in $\overline{S}$ (resp. $\overline{T}$) that are reachable from initial nodes. Calculate the value of the *adjustment parameter* μ as follows:

$$\begin{aligned}
\mu_1 &:= \min\left(\pi(\widetilde{v}) : v \in \overline{T}_0\right), \\
\mu_2 &:= \min\left(w_\pi(a) : a = (u,v) \in A\overline{G},\ u \in \overline{S}_0,\ v \in \overline{S}\right), \\
\mu_3 &:= \min\left(w_\pi(a) : a = (u,v) \in A\overline{G},\ u \in \overline{S}_0,\ v \in \overline{T} - \overline{T}_0\right), \\
\mu_4 &:= \min\left(1 - w_\pi(a) : a = (u,v) \in \overline{B},\ u \in \overline{S} - \overline{S}_0,\ v \in \overline{T}_0\right), \\
\mu &:= \min(\mu_1, \mu_2, \mu_3, \mu_4).
\end{aligned}$$

Clearly, (P2) implies that $\mu \geq 0$. Also, $\mu_4 \in \{0, 1, \infty\}$ (moreover, case $\mu_4 = 0$ is not possible, see Lemma 5 below).

A *dual update* is performed as follows:

$$\begin{aligned}
\pi(\widetilde{v}) &:= \pi(\widetilde{v}) + \mu \quad \text{for each } v \in \overline{S}_0, \\
\pi(\widetilde{v}) &:= \pi(\widetilde{v}) - \mu \quad \text{for each } v \in \overline{T}_0.
\end{aligned}$$

If the dual $\pi(\widetilde{v})$ of some complex node $v \in \overline{T}_0$ drops to zero then the algorithm calls EXPAND($\widetilde{v}$) to remove $\widetilde{v}$ from $\mathcal{F}$, uncontract (partly) graph $\overline{G}$, and extend $\overline{B}$ into the shell of $\widetilde{v}$.

Also, suppose $w_\pi(a) = 0$ holds for some $a = (u,v) \in A\overline{G}$, $u \in \overline{S}_0$, $v \in \overline{S}$ after the update. Then, node u must be a root of F_S and property (F2) fails. Procedure NORMALIZE-FOREST is called to restore it.

One can see the following:

Lemma 5. *If no augmenting path exists in H then $0 < \mu < \infty$. The dual step preserves properties (P1)–(P3), (D1)–(D3), and (F1)–(F4), does not change the set of initial nodes, and does not decrease the set of reachable nodes.*

If an augmenting path arises after these changes, the dual step completes and a primal step is executed. Otherwise, the next value of μ is calculated and the process of changing π proceeds.

The algorithm changes $\overline{B}$, π, and $\mathcal{F}$ by executing primal and dual steps alternatively. It stops when $\overline{B}$ covers all nodes in $\overline{S}$. Then, parts $\overline{S}$ and $\overline{T}$ are exchanged and the TS stage runs until $\overline{B}$ covers both $\overline{S}$ and $\overline{T}$. This way, the requested bibranching $\overline{B}$ is constructed, the scaling step completes.

5 Complexity Analysis

In this section we present a sketch of the efficiency analysis. Our immediate goal is to prove an $O(\sqrt{n})$ bound for the number of primal and dual steps during each scaling step. Hereinafter w, π_0, $\mathcal{F}_0$, and $\overline{G}_0$ denote the corresponding objects after the doubling stage. Similarly, we use notation π_1, $\mathcal{F}_1$, and $\overline{G}_1$ when referring to the state immediately after the shell stage.

Lemma 6. *There exists a canonical π_1-consistent bibranching $B_1 \subseteq AG$ obeying $w_{\pi_1}(B_1) \leq 6n$.*

Proof. Let $\overline{B}_0$ be a bibranching in $\overline{G}_0$ that was constructed by the previous scaling step (or just an arbitrary bibranching in G for the first invocation of the scaling step). By removing an appropriate set of arcs from $\overline{B}_0$, one may assume that $\overline{B}_0$ is canonical, see Fact 1. Property (P2) and the structure of the doubling stage imply that $w_{\pi_0}(a) \leq 3$ holds for each $a \in \overline{B}_0$.

We now gradually transform graph $\overline{G}_0$ into $\overline{G}_1$ and, simultaneously, $\overline{B}_0$ into a bibranching $\overline{B}_1$ in $\overline{G}_1$ such that $w_{\pi_1}(\overline{B}_1)$ is small and (P3) holds for $\overline{B} := \overline{B}_1$, $\pi := \pi_1$. We denote the current graph by $\overline{G}$ and the current bibranching by $\overline{B}$. Initially, put $\overline{G} := \overline{G}_0$ and $\overline{B} := \overline{B}_0$. The difference between $\overline{G}_0$ and $\overline{G}_1$ is that some maximal sets in $\operatorname{supp}(\pi_0)$ could have dissolved during the shell stage. The corresponding dissolved nodes, therefore, are replaced by certain subgraphs.

We enumerate the nodes of $\overline{G}_0$, let v be the current one. If $\widetilde{v} \notin \operatorname{supp}(\pi_0)$ then v is a simple node ($\widetilde{v} = \{v\}$), it remains simple in $\overline{G}_1$. No change is applied to $\overline{G}$ and $\overline{B}$. Note that the reduced weights of arcs covering v are not changed during the shell stage and, thus, do not exceed 3.

Next, suppose $\widetilde{v} \in \operatorname{supp}(\pi_0)$. We assume that $\widetilde{v} \subseteq S$ (the other case is symmetric). Property (P3) implies that in $\overline{G}_0$ node v is covered by a unique arc, say $a \in \overline{B}_0$. Let the tail of arc a in G be w. Applying (D2) iteratively, we construct an out-tree W in G such that: (i) W is rooted at w; (ii) $VW = \widetilde{v}$; (iii) if $X \in \operatorname{supp}(\pi_0)$ and $X \subseteq \widetilde{v}$ then X is covered by exactly one arc in $A := AW \cup \{a\}$; (iv) every arc $a \in AW$ was of zero reduced weight prior to the doubling stage. Clearly, $w_{\pi_1}(a) \leq 3$ holds for every $a \in A$.

Update $\overline{G}$ and $\overline{B}$ as follows. First, uncontract $\widetilde{v}$ completely and add AW to $\overline{B}$. Since node w is reachable from every node in $\widetilde{v}$ by arcs in AW, it follows that $\overline{B}$ remains a bibranching. Next, contract the maximal sets $X \in \operatorname{supp}(\pi_1)$ such that $X \subseteq \widetilde{v}$ and update $\overline{B}$ accordingly. Let $\overline{v}$ denote the image of $\widetilde{v}$ under these contractions. (It is possible that the whole set $\widetilde{v}$ gets contracted again, this happens when $\widetilde{v} \in \operatorname{supp}(\pi_1)$; in this case $\widetilde{v}$ did not dissolve during the call NORMALIZE-SHELL($\widetilde{v}$).)

The above contractions may remove some arcs from $\overline{B}$ (more precisely, exactly those arcs whose head and tail nodes are simultaneously contained in the same maximal contracted set). However, $\overline{B}$ remains a bibranching since contraction of an arbitrary subset of S- or T-part of the graph preserves the required connectivity. Finally, we apply Fact 1 and remove all redundant arcs from $\overline{B}$ (in an arbitrary way) turning it into a canonical bibranching.

Recall that initially v was covered by the unique arc a. Now v is expanded into some set of nodes, namely $\overline{v}$, and some arcs from $\gamma(\widetilde{v})$ are added to $\overline{B}$. Canonicity implies that every node in $\overline{v}$ is covered by a unique arc from $\overline{B}$. This way, (P3) follows for $\overline{B}$.

Let us estimate the total reduced weight of all newly added arcs in $\overline{B}$. To this aim, we bound $w_{\pi_1}(A)$ (since $\overline{B}$ receives some subset of arcs from A and reduced weights of the omitted arcs are non-negative). We consider the following two subfamilies of $\operatorname{supp}(\pi_0)$ and $\operatorname{supp}(\pi_1)$:

$$\mathcal{F}_0 := \{X \in \operatorname{supp}(\pi_0) \mid X \subseteq \widetilde{v}\}, \qquad \mathcal{F}_1 := \{X \in \operatorname{supp}(\pi_1) \mid X \subseteq \widetilde{v}\}.$$

During NORMALIZE-SHELL, each time the dual variable corresponding to a set $X \in 2^S$ (resp. $X \in 2^T$) is increased by μ, the dual variable corresponding to some other set $Y \in 2^S$ (resp. $Y \in 2^T$) is decreased by the same value μ. Hence,

$$\sum(\pi_0(X) : X \in \mathcal{F}_0) = \sum(\pi_1(X) : X \in \mathcal{F}_1).$$

Also, by Fact 3 it follows that

$$\begin{aligned} w_{\pi_0}(A) &= w(A) - \vartheta\pi_0(A) = w(A) - \textstyle\sum(\pi_0(X) : X \in \mathcal{F}_0), \\ w_{\pi_1}(A) &= w(A) - \vartheta\pi_1(A) \le w(A) - \textstyle\sum(\pi_1(X) : X \in \mathcal{F}_1). \end{aligned}$$

Therefore,

$$w_{\pi_1}(A) \le w_{\pi_0}(A) \le 3\,|A| = 3\,|\widetilde{v}|.$$

The above procedure is applied to each node $v \in V\overline{G}_0$ and eventually stops with $\overline{G} = \overline{G}_1$. The final set $\overline{B}$ is denoted by $\overline{B}_1$. Let us estimate its reduced weight $w_{\pi_1}(\overline{B}_1)$. First, $\overline{B}_1$ gets at most n arcs that cover simple nodes in $\overline{G}_0$; each of those arcs has a reduced weight not exceeding 3. Second, each complex node $v \in V\overline{G}_0$ generates a set of arcs with total reduced weight not exceeding $3\,|\widetilde{v}|$. Summing these bounds, one gets:

$$w_{\pi_1}(\overline{B}_1) \le 3n + 3n = 6n.$$

Finally, to get the desired bibranching B_1 in G we apply EXPAND routine and extend $\overline{B}_1$ into the maximal contracted sets of $\overline{G}_1$. This step only adds arcs of zero reduced weight w_{π_1}. Hence, $w_{\pi_1}(B_1) = w_{\pi_1}(\overline{B}_1) \le 6n$ holds.

Lemma 7. *Let $\pi\colon 2^S \cup 2^T \to \mathbb{R}_+$ be an arbitrary function and $B \subseteq AG$ be an arbitrary π-consistent arc set satisfying properties (D3) and (P2) (for $\overline{G} := G$, $\overline{B} := B$). Then*

$$\Delta(\pi, \pi_1, B) := \sum\big(\pi(X) - \pi_1(X) : X \text{ is not covered by } B\big) \le 6n + w_\pi(B).$$

Proof. Consider the duals π_1 (at the moment just before the ST stage) and the canonical bibranching B_1 constructed in Lemma 6. Put

$$Q := \sum_{a \in AG} \big(\chi^{B_1}(a) - \chi^{B}(a)\big)\big(w_{\pi_1}(a) - w_\pi(a)\big).$$

(Here χ^U denotes the *incidence vector* of an arc set U, i.e. the function that equals 1 on U and 0 on $AG - U$.)

Taking into account equalities $w_\pi = w - \vartheta\pi$ and $w_{\pi_1} = w - \vartheta\pi_1$ one gets

$$\begin{aligned} Q &= \sum_{a \in AG} \big(\chi^{B_1}(a) - \chi^B(a)\big)\big(\vartheta\pi(a) - \vartheta\pi_1(a)\big) = \\ &= \vartheta\pi(B_1) + \vartheta\pi_1(B) - \vartheta\pi_1(B_1) - \vartheta\pi(B). \end{aligned}$$

Since B is π-consistent and B_1 is π_1-consistent and covers each set in $2^S \cup 2^T$, Fact 3 implies

$$\begin{aligned} Q \geq & \sum \big(\pi(X) : X \text{ is covered by } B_1\big) + \sum \big(\pi_1(X) : X \text{ is covered by } B\big) - \\ & - \sum \big(\pi_1(X) : X \text{ is covered by } B_1\big) - \sum \big(\pi(X) : X \text{ is covered by } B\big) = \\ & = \sum \big(\pi(X) : X \text{ is not covered by } B\big) - \\ & - \sum \big(\pi_1(X) : X \text{ is not covered by } B\big) = \\ & = \sum \big(\pi(X) - \pi_1(X) : X \text{ is not covered by } B\big). \end{aligned}$$

On the other hand,

$$\begin{aligned} Q &= \sum_{a \in AG} \big(\chi^{B_1}(a) - \chi^B(a)\big)\big(w_{\pi_1}(a) - w_\pi(a)\big) \leq \\ &\leq \sum_{a \in AG} \chi^{B_1}(a) w_{\pi_1}(a) + \chi^B(a) w_\pi(a) \leq \\ &\leq w_{\pi_1}(B_1) + w_\pi(B) \leq 6n + w_\pi(B). \end{aligned}$$

Now the claim follows by transitivity.

Lemma 8. *Each scaling step executes $O(\sqrt{n})$ primal and dual steps.*

Proof. Let $\overline{B} \subseteq A\overline{G}$ denote the current partial bibranching in the current graph $\overline{G}$ at some intermediate moment during the ST or the TS stage.

Firstly, we prove that $w_\pi(\overline{B}) \leq n$. Indeed $w_\pi(\overline{B})$ does not exceed the number of $\overline{S}$–$\overline{T}$ arcs in $\overline{B}$ (by property (P2)). The latter can only increase by 1 on each primal step. The total number of primal steps does not exceed n (since each of these steps increases the set of covered nodes).

Next, we proceed similarly to Lemma 1, apply EXPAND routine, and extend $\overline{B}$ to a π-consistent set $B \subseteq AG$. This set obeys $w_\pi(B) = w_\pi(\overline{B}) \leq n$.

Consider the sum $\Delta(\pi, \pi_1, B)$. Let X be a set such that $\pi(X) < \pi_1(X)$. It follows from the structure of the algorithm that B covers X. Indeed, when the dual $\pi(\widetilde{v})$ decreases (for $v \in V\overline{G}$), $\overline{B}$ covers v for otherwise v is a reachable final node of type (3) and the dual adjustment is not possible. Also, if $\overline{B}$ covers v then B covers $\widetilde{v}$ and all its subsets. Therefore, all terms in $\Delta(\pi, \pi_1, B)$ are non-negative.

Let us deal with the positive terms. Consider the ST stage. Initially, all nodes in $\overline{S}$ are not covered by $\overline{B}$. Then, during the course of the algorithm the number

of uncovered nodes in $\overline{S}$ decreases. Let k dual steps be performed so far. For each i $(1 \le i \le k)$ let $D^i_1, \ldots, D^i_{l_i}$ be the subsets of S that correspond to all uncovered nodes $d^i_1, \ldots, d^i_{l_i}$ in $\overline{S}$ during the i-th dual step. For each fixed i, the sets $D^i_1, \ldots, D^i_{l_i}$ are pairwise disjoint. Altogether, these sets form a laminar family

$$\mathcal{D} := \left\{ D^i_j \mid 1 \le i \le k, 1 \le j \le l_i \right\}.$$

Each uncovered node d^i_j was initial (at the corresponding moment of time) and, hence, reachable. Therefore, $\pi(D^i_j)$ was increased during the corresponding dual step.

Moreover, for each set D^i_j, $i < k$, $1 \le j \le l_i$, there are exactly two possibilities: (i) node d^i_j gets covered during the upcoming primal step and, hence, $D^{i'}_{j'} \cap D^i_j = \emptyset$ for each $i' > i$ and $1 \le j' \le l_{i'}$; or (ii) node d^i_j gets incorporated into some contracted set at the next dual step, hence, $D^i_j \subseteq D^{i+1}_{j'}$ for some $1 \le j' \le l_{i+1}$.

This way, sets D^i_j form a forest F and for each its tree H with root D^i_j the leafs of H (these are some sets of the form D^1_j, $1 \le j \le l_1$) are on the same depth i, which is equal to the height of H. Let $H_1, \ldots, H_{l_k}$ denote the set of trees in F rooted at $D^k_1, \ldots D^k_{l_k}$. All these trees are of height k. Also, for each $1 \le i \le k$ and $1 \le j \le l_k$ there exists a set D^i_j in tree H_j such that B does not cover D^i_j. Therefore, each tree H_j adds a positive term of value $\sum \mu$ to $\Delta(\pi, \pi_1, B)$ (where $\sum \mu$ denotes the sum of the dual adjustments μ performed by the algorithm). Now summing over all trees $H_1, \ldots, H_{l_k}$ one gets

$$\Delta(\pi, \pi_1, B) \ge \sum \mu \cdot l_k, \tag{3}$$

On the other hand, by Lemma 7 it follows that

$$\Delta(\pi, \pi_1, B) \le 6n + w_\pi(B) \le 7n \tag{4}$$

Since each dual step changes duals by at least 1, we conclude that $k \cdot l_k \le 7n$. Hence, after $\lceil \sqrt{n} \rceil$ dual steps at most $O(\sqrt{n})$ nodes in $\overline{S}$ may remain uncovered. To cover these remaining nodes $O(\sqrt{n})$ primal steps are sufficient (as each such step decreases the number of uncovered nodes by 1).

Next, consider the state after k dual steps in the TS stage and let, as earlier, π and $\overline{B} \subseteq A\overline{G}$ denote the current duals the current partial bibranching, respectively. Put $B \subseteq AG$ to be the result of expanding $\overline{B}$. We have shown earlier that there are no negative terms in $\Delta(\pi, \pi_1, B)$. Moreover, the very same argument (with $\overline{S}$ and $\overline{T}$ exchanged) applies, so one can construct subsets $D^i_j \subseteq T$ similarly to the ST stage. Since each of these sets D^i_j corresponds to an uncovered node at some intermediate moment and the algorithm can only decrease duals of sets that are covered, it follows that none of $\pi(D^i_j)$ is changed during the ST stage. Thus, (3) and (4) hold for the TS stage as well, and the latter completes after executing $O(\sqrt{n})$ primal and dual steps.

Taking the above Lemma 8 into account and applying the ideas from [Edm67, GGSR86, FT87, KP98] one can implement the shell stage to run in $O(m \log n)$

time and both ST and TS stages to take $O(m\sqrt{n}\log n)$ time (the $\log n$ factor in the above estimates comes from the complexity of priority queue operations, maintenance of $\mathcal{F}$, and computation of reduced arc weights; we employ binary heaps and dynamic trees [ST83] for these purposes). Recall (see Section 3) that the total number of scaling steps is $\lceil \log W \rceil + \lfloor \log n \rfloor + 1 = O(\log(nW))$. Hence, we conclude as follows:

Theorem 5. *The running time of the algorithm is* $O(m\sqrt{n}\ \log n \log(nW))$.

Acknowledgements

The author is thankful to Petr Mitrichev (Faculty of Mechanics and Mathematics, Moscow State University) and also to the anonymous referees for helpful comments and suggestions.

References

[Din80] Dinic, E.: Algorithm for solution of a problem of maximum flow in networks with power estimation. Soviet Math. Dokl. 11, 1277–1280 (1980)

[Edm67] Edmonds, J.: Optimum branchings. J. Res. Nat. Bur. Standards 71B, 233–240 (1967)

[FT87] Fredman, M., Tarjan, R.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34(3), 596–615 (1987)

[GGSR86] Gabow, H., Galil, Z., Spencer, T., Tarjan, R.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica 6(2), 109–122 (1986)

[GT87] Goldberg, A., Tarjan, R.: Solving minimum-cost flow problems by successive approximation. In: STOC 1987: Proceedings of the nineteenth annual ACM conference on Theory of computing, pp. 7–18 (1987)

[GT89] Gabow, H., Tarjan, R.: Faster scaling algorithms for network problems. SIAM J. Comput. 18(5), 1013–1036 (1989)

[GT91] Gablow, H., Tarjan, R.: Faster scaling algorithms for general graph matching problems. J. ACM 38(4), 815–853 (1991)

[KP98] Keijsper, J., Pendavingh, R.: An efficient algorithm for minimum-weight bibranching. J. Comb. Theory, Ser. B 73(2), 130–145 (1998)

[Sch82] Schrijver, A.: Min-max relations for directed graphs. Ann. Discrete Math. 16, 261–280 (1982)

[Sch03] Schrijver, A.: Combinatorial Optimization. Springer, Berlin (2003)

[ST83] Sleator, D., Tarjan, R.: A data structure for dynamic trees. Journal of Computer and System Sciences 26(3), 362–391 (1983)