Primal-Dual Approach for Directed Vertex Connectivity Augmentation and Generalizations

LÁSZLÓ A. VÉGH

Eötvös Univeristy, Budapest

AND

ANDRÁS A. BENCZÚR

Computer and Automation Institute, Hungarian Academy of Sciences

Abstract. In their seminal paper, Frank and Jordán [1995] show that a large class of optimization problems, including certain directed graph augmentation, fall into the class of covering supermodular functions over pairs of sets. They also give an algorithm for such problems, however, it relies on the ellipsoid method. Prior to our result, combinatorial algorithms existed only for the 0–1 valued problem. Our key result is a combinatorial algorithm for the general problem that includes directed vertex or $S-T$ connectivity augmentation. The algorithm is based on Benczúr's previous algorithm for the 0–1 valued case [Benczúr 2003].

Our algorithm uses a primal-dual scheme for finding covers of partially ordered sets that satisfy natural abstract properties as in Frank and Jordán. For an initial (possibly greedy) cover, the algorithm searches for witnesses for the necessity of each element in the cover. If no two (weighted) witnesses have a common cover, the solution is optimal. As long as this is not the case, the witnesses are gradually exchanged for smaller ones. Each witness change defines an appropriate change in the solution; these changes are finally unwound in a shortest-path manner to obtain a solution of size one less.

Categories and Subject Descriptors: G.2.1 [**Discrete Mathematics**]: Combinatorics—*Combinatorial algorithms*

General Terms: Algorithms

Additional Key Words and Phrases: Vertex connectivity augmentation, combinatorial algorithm

ACM Reference Format:

Végh, L. A. and Benczúr, A. A. 2008. Primal-dual approach for directed vertex connectivity augmentation and generalizations. ACM Trans. Algor. 4, 2, Article 20 (May 2008), 21 pages. DOI = 10.1145/1361192.1361197 http://doi.acm.org/10.1145/1361192.1361197

This article is an extended version of the paper presented at SODA'05.

The authors are member of the Egerváry Research Group (EGRES) and are supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T037547, K60802, T042481 and T042706, the European MCRTN ADONET, Grant Number 504438 and by the Inter-University Center for Telecommunications and Informatics (ETIK).

Authors' addresses: L. A. Végh, e-mail: veghal@cs.elte.hu; A. A. Benczúr, e-mail: benczur@sztaki.hu

1. *Introduction*

Connectivity augmentation problems form a subclass of survivable network design [Goemans and Williamson 1997] where one is interested in the minimum number of edges needed to be added to a graph to satisfy certain connectivity prescriptions. Algorithms for various augmentation problems have a large and expanding literature [e.g., Benczúr 1999; Benczúr and Karger 2000; Cai and Sun 1989; Frank 1992; Frank and Jordán 1995; Gabow 1994; Jordán 1993, 1995].

In this article, we give a combinatorial algorithm for the general problem of *covering supermodular functions over pairs of sets* introduced by Frank and Jordán [1995] as a generalization for directed vertex and edge connectivity augmentation problems. They also give an algorithm for this problem using the ellipsoid method. Previously, combinatorial algorithms existed only for special problems [Frank 1999a, 1999b] and for increasing connectivity by one [Benczúr 2003; Frank 2003]. For the problem of increasing directed vertex connectivity to target value k, the best previous combinatorial algorithm has running time polynomial in n but exponential in k [Frank and Jordán 1999]. The running time of our algorithm can be bounded by $O(n^7)$.

The central example of covering supermodular functions over pairs of sets is finding the minimum number of directed edges that make a directed graph G k-vertex-connected. We consider all cuts of G with less than k vertices as set pairs (X^-, X^+) of the vertex set, where X^- is the sink and X^+ is the source side of the cut (recall the graph is directed). For a directed cut with sides X^- and X^+, let

$$p(X^-, X^+) = \max\{0, k - (|V| - |X^-| - |X^+|)\}$$

denote the number of vertices 'missing' for a k-connected graph; for all other pairs X^-, X^+ with $X^- \cap X^+ = \emptyset$, let $p(X^-, X^+) = 0$. The graph becomes k-connected if and only if for all X^-, X^+ with $X^- \cap X^+ = \emptyset$, we add at least $p(X^-, X^+)$ edges that lead from X^- to X^+.

Our algorithm works for the generalization of the a forementioned demand function p to arbitrary crossing supermodular functions; for the general covering problem, our algorithm is weakly polynomial in the sense that the running time depends on the maximal value of the function p. A demand function p is called *crossing supermodular* if for all $X^- \cap Y^- \neq \emptyset$, $X^+ \cap Y^+ \neq \emptyset$ and $p(X^-, X^+) > 0$, $p(Y^-, Y^+) > 0$,

$$p(X^- \cap Y^-, X^+ \cup Y^+) + p(X^- \cup Y^-, X^+ \cap Y^+) \geq p(X^-, X^+) + p(Y^-, Y^+).$$

Another problem that falls into the class of covering set pairs is increasing directed $S{-}T$ vertex or edge connectivity to target value k by adding a minimum number of edges between S and T [Frank and Jordán 1995]. For two possibly overlapping vertex sets S and T, the $S{-}T$ connectivity is the maximum number of directed vertex (or edge) disjoint paths that connect vertices in S to vertices in T. Yet another remarkable problem of this class is Győri's rectangle cover problem [Győri 1984; Franzblau and Kleitman 1986; Benczúr et al. 1999].

At the heart of most results related to covering problems over set pairs, we find Dilworth's theorem stating that the minimum number of chains that cover a partially ordered set is equal to the maximum number of pairwise incomparable elements of the set. Both the noncombinatorial algorithm [Frank and Jordán 1995] and certain combinatorial ones [Frank 1999a, 1999b, 2003; Benczúr et al. 1999] start with a reduction to chain covers as in Dilworth's theorem.

Similar to most results related to covering problems over set pairs [Frank and Jordán 1995; Frank 1999a, 1999b, 2003; Benczúr et al. 1999], we find Dilworth's chain cover theorem at the heart of our new combinatorial algorithm. However, we circumvent the reduction to Dilworth's theorem; instead we give a more general algorithm that resembles the folklore Dilworth algorithm as described in Frank [1976]. The relation between supermodular functions over set pairs and Dilworth's chain covers is based on the following observation. We say that a directed edge xy covers the pair (X^-, X^+), if $x \in X^-$ and $y \in X^+$. It is easy to see that there is a pair $(X^-_{\min}, X^+_{\min})$ covered by xy with $X^-_{\min}$ minimum and $X^+_{\min}$ maximum and another pair $(X^-_{\max}, X^+_{\max})$ covered by xy with $X^-_{\max}$ maximum and $X^+_{\max}$ minimum. A pair (X^-, X^+) is covered by xy if and only if $X^-_{\min} \subseteq X^- \subseteq X^-_{\max}$ and $X^+_{\min} \supseteq X^+ \supseteq X^+_{\max}$. For the partial ordered set of the set pairs with this 'skew' containment relation, we get the problem of covering a special partial ordered set by intervals similar to Dilworth's problem of covering a poset by chains.

Our algorithm is based on that of Benczúr [2003] for the simpler case where p may only take values 0 and 1. This algorithm directly generalizes a Dilworth algorithm; for a crossing supermodular p that may take arbitrary nonnegative integer values, we start out with a multichain version of Dilworth's problem where poset elements have weights, and the total number of chains containing an element must be at least its weight. We consider multiple copies of the same chain instead of weighted chains; our algorithm is pseudopolynomial in this sense.

We construct an optimum interval cover by starting with an arbitrary (possibly greedy) cover and gradually improve it in a primal-dual augmenting path manner that mimics standard Dilworth algorithms, as described in Frank [1976].

In the bulk of this article, we describe our algorithm that solves certain directed edge augmentation problems via a reduction to covering a poset by weighted intervals where poset elements are weighted by a supermodular function p. As shown in Section 2, this covering problem is equivalent to that considered by Frank and Jordán [1995]. Thus our algorithm applies to the task (among others) of increasing directed vertex connectivity or directed S–T edge connectivity to a target value.

The rest of the article, is organized as follows. In Section 2, we give the main definitions and state the equivalence of our theorem with that of Frank and Jordán [1995]. In Section 3, we first give an overview of the primal-dual procedure, then in separate sections show the key procedures PUSHDOWN and REDUCE and in separate sections show their correctness. Finally in Sections 4 and 5, we briefly elaborate on the running times for the augmentation problems.

2. *Poset Properties of the Frank–Jordán Set Pairs*

Frank and Jordán [1995] introduce systems of set pairs closed under a certain *skew intersection* operation defined next. Let two members (X^-, X^+) and (Y^-, Y^+) be called *dependent* if both $X^- \cap Y^-$ and $X^+ \cap Y^+$ are nonempty; otherwise they are *independent*. Observe that (X^-, X^+) and (Y^-, Y^+) are independent if and only if they cannot be covered by the same edge. Then for all dependent pairs,

$$(X^- \cap Y^-, X^+ \cup Y^+),\ (X^- \cup Y^-, X^+ \cap Y^+) \tag{1}$$

are also members of the set system. A function p over the system of set pairs satisfies the *crossing supermodular* property if for all dependent (X^-, X^+) and (Y^-, Y^+) with $p(X^-, X^+) > 0$ and $p(Y^-, Y^+) > 0$,

$$p(X^- \cap Y^-, X^+ \cup Y^+) + p(X^- \cup Y^-, X^+ \cap Y^+) \geq p(X^-, X^+) + p(Y^-, Y^+).$$

They prove the following theorem:

THEOREM 2.1. [Frank and Jordán 1995]. *Let p be a crossing supermodular function over a system of set pairs closed under the operations* (1). *The minimum cardinality of an edge multiset $\{e = (v_1, v_2)\}$, such that, for all (X^-, X^+), there exist $p(X^-, X^+)$ edges with $v_1 \in X^-$, $v_2 \in X^+$, is equal to the maximum sum of p-values for pairwise independent elements in the system of set pairs.*

We give an alternate proof of an equivalent form of this theorem stated as a poset covering problem. The proof is via a combinatorial algorithm.

Definition 2.2. Consider a poset $(\mathcal{P}, \leq)$. We say that, for a minimal element m and a maximal element M, the set $\{z : m \leq z \leq M\}$ is the *interval* $[m, M]$. Let $x, y \in \mathcal{P}$ be called *dependent* if there exists an interval $[m, M]$ with $x, y \in [m, M]$; otherwise they are called *independent*.

We say that $(\mathcal{P}, \leq)$ satisfies the *strong interval property*, if the following hold.

(1) For all dependent $x, y \in \mathcal{P}$, the operations $x \vee y = \min\{z : z \geq x, z \geq y\}$ and $x \wedge y = \max\{z : z \leq x, z \leq y\}$ are uniquely defined.
(2) For every interval $[m, M]$,

$$x \wedge y \in [m, M] \text{ implies } x \in [m, M] \text{ or } y \in [m, M],$$

and the same holds with $x \wedge y$ replaced by $x \vee y$.

The notion of a *crossing supermodular function p* over the poset follows, similar to set pairs: for all dependent x and y with $p(x) > 0$ and $p(y) > 0$, we require

$$p(x \vee y) + p(x \wedge y) \geq p(x) + p(y).$$

Consider a multiset of intervals $\mathcal{I}$. We say that $\mathcal{I}$ *covers* the function p or $\mathcal{I}$ is a *cover* of p if for every x, at least $p(x)$ intervals in $\mathcal{I}$ contain x. An element v is called *tight* if it is contained in exactly $p(x)$ intervals in $\mathcal{I}$.

Given the notion of the cover problem for a poset with the strong interval property, we next show its equivalence to the Frank and Jordán set pair cover problem. First we show the equivalence of the poset properties as seen in Figure 1.

THEOREM 2.3. *For two sets $\mathcal{X}^-, \mathcal{X}^+$, let $\mathcal{P} \subseteq \{(X^-, X^+) : X^- \subseteq \mathcal{X}^-, X^+ \subseteq \mathcal{X}^+\}$ such that for all dependent $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$,*

$$x \wedge y = (X^- \cap Y^-, X^+ \cup Y^+) \in \mathcal{P},$$
$$x \vee y = (X^- \cup Y^-, X^+ \cap Y^+) \in \mathcal{P}.$$

For any $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$, let $x \leq y$ if and only if $X^- \subseteq Y^-$ and $X^+ \supseteq Y^+$. Then $(\mathcal{P}, \leq)$ with operations $\vee$, $\wedge$ satisfies the strong interval property. Furthermore subfamilies

$$I_{v_1,v_2} = \{(X^-, X^+) : v_1 \in X^-, v_2 \in X^+\}$$

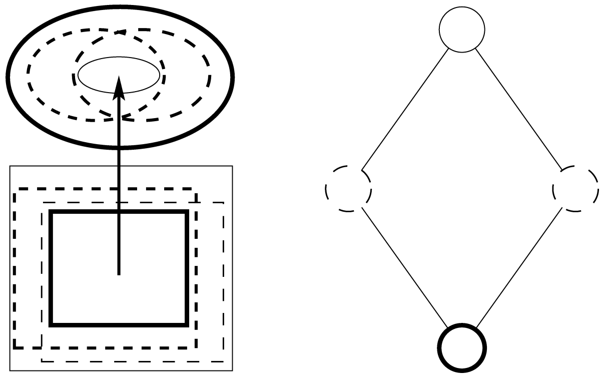

FIG. 1. The correspondence between set pairs and poset elements. The four pairs on the left side can be covered by one edge, and the corresponding four elements are contained in one interval.

for pairs $v_1 \in \mathcal{X}^-$, $v_2 \in \mathcal{X}^+$ are either intervals themselves or contained by some intervals of $\mathcal{P}$. Furthermore, for all intervals of $\mathcal{P}$, there exist v_1 and v_2 such that the interval can be given in such a form.

PROOF. Property (1) of Definition 2.2 follows directly from the properties of set union, intersection, and containment.

To show the relation of intervals and subfamilies defined by pairs of vertices, consider I_{v_1,v_2} first. Since all set pairs $\{(X^-, X^+) : v_1 \in X^-, v_2 \in X^+\}$ of I_{v_1,v_2} are dependent, we may define $u = \bigwedge I_{v_1,v_2}$ and $v = \bigvee I_{v_1,v_2}$ and choose a minimal $m \leq u$ and maximal $M \geq v$ to prove $I_{v_1,v_2} \subseteq [m, M]$.

Now we show that a pair v_1, v_2 exist for all intervals $[m, M] = [(m^-, m^+), (M^-, M^+)]$. We take an arbitrary pair $v_1 \in m^-$ and $v_2 \in M^+$, and we show that this is an appropriate selection. If $z = (Z^-, Z^+) \in [m, M]$, then $m^- \subseteq Z^-$ and $M^+ \subseteq Z^+$, hence $v_1 \in Z^-$ and $v_2 \in Z^+$. Assume now we have some z of this form with $z \notin [m, M]$, that is, either $m \not\leq z$ or $z \not\leq M$. z is dependent on both m and M since $v_1 \in z^- \cap m^- \cap M^-$ and $v_2 \in z^+ \cap m^+ \cap M^+$, thus $z \wedge m$ and $z \vee M$ exists. In the first case, $z \wedge m < m$, in the second case, $z \vee M > M$, both contradicting the extremity of m or M.

To show Property (2) of Definition 2.2, we take a pair v_1, v_2 as defined for the interval $[m, M]$. It suffices to show that the graph edge $v_1 v_2$ covers either $x = (X^-, X^+)$ or $y = (Y^-, Y^+)$. Notice $v_1 \in X^- \cap Y^-$ and $v_2 \in X^+ \cup Y^+$. The former implies $v_1 \in X^-$ and $v_1 \in Y^-$, while the latter implies $v_2 \in X^+$ or $v_2 \in Y^+$. The same holds with $x \wedge y$ replaced by $x \vee y$, hence the claim follows. $\square$

Before giving our algorithm, we state our main result as a min-max formula.

THEOREM 2.4. *For a poset $(\mathcal{P}, \leq)$ with the strong interval property and a crossing supermodular function p, the minimum number of intervals covering $\mathcal{P}$ is equal to the maximum of the sum of p values for pairwise independent elements of $\mathcal{P}$.*

Theorem 2.3 implies that Theorem 2.1 is a special case of this theorem. Now we show that Theorem 2.1 implies Theorem 2.4 as well, hence they are equivalent. Given a poset $\mathcal{P}$ with the strong interval property, let us define a representative element $\varphi(x)$ for every minimal or maximal element x. For $a \in \mathcal{P}$, let us define the pair $\delta(a) = (a^-, a^+)$ so that

$$a^- = \{\varphi(m) : m \le a,\ m \in \mathcal{P} \text{ minimal}\};\ a^+ = \{\varphi(M) : M \ge a, M \in \mathcal{P} \text{ maximal}\}.$$

It is easy to show that the function δ is a homomorphism for $\vee$, $\wedge$ and $\le$, and that the function defined by $p'(X^-, X^+) := \max\{p(a) : \delta(a) = (X^-, X^+)\}$ is crossing supermodular. Hence applying Theorem 2.1 for p' on the pairs of sets implies Theorem 2.4.

We conclude the section by showing some basic properties of the tight elements.

LEMMA 2.5. *If x and y are two dependent tight elements with $p(x) > 0$, $p(y) > 0$, then both $x \vee y$ and $x \wedge y$ are tight.*

PROOF. Let $g(x)$ denote the number of intervals covering element x. By the strong interval property, all intervals that cover $x \vee y$ or $x \wedge y$ also cover x or y, and, if they cover both, then they cover all four, hence $g(x)+g(y) \ge g(x \vee y)+g(x \wedge y)$. The proof is completed by

$$\begin{aligned} g(x \vee y) + g(x \wedge y) &\ge p(x \vee y) + p(x \wedge y) \\ &\ge p(x) + p(y) = g(x) + g(y) \ge g(x \vee y) + g(x \wedge y), \end{aligned} \tag{2}$$

implying equality everywhere. Here the first inequality follows since we have a cover; the second is the definition of crossing supermodularity; and the equality follows by the tightness of x and y. □

The following easy corollary will be used throughout the article.

COROLLARY 2.6. *For a cover $\mathcal{I}$, every $I \in \mathcal{I}$ has a unique minimal and a unique maximal tight element.*

LEMMA 2.7. *If x and y are two dependent tight elements with $p(x) > 0$, $p(y) > 0$ and the interval $[m, M] \in \mathcal{I}$ contains x, then it contains at least one of $x \vee y$ and $x \wedge y$ or, equivalently, $y \le M$ or $m \le y$.*

PROOF. Recall that by the proof of Lemma 2.5, we have equality everywhere in (2); the last inequality hence turns to $g(x) + g(y) = g(x \vee y) + g(x \wedge y)$. By the strong interval property all intervals that cover $x \vee y$ or $x \wedge y$ also cover x or y and, if they cover both, then they cover all four. Hence the equality implies the claim. □

3. *The Algorithm*

We give a brief overview of our algorithm for the 0–1 valued case first. The algorithm starts out with a (possible greedy) interval cover $\mathcal{I} = \{I_1, \ldots, I_k\}$. In Algorithm PUSHDOWN-REDUCE, we maintain a tight element $u_i \in I_i$ for each interval I_i as a witness for the necessity of I_i in the cover. A long as the set of witnesses are nonindependent or, in other words, they do not form a dual solution, in Procedure PUSHDOWN we replace certain u_i by smaller elements. By such steps, we aim to arrive at an independent system of witnesses. If witnesses are indeed pairwise independent, they form a dual solution with the same value as the primal cover solution, thus showing both primal and dual optimality. Otherwise in Procedure PUSHDOWN, the Procedure REDUCE, a procedure that exchanges interval endpoints so that we get an interval cover of size one less, is called.

In order to handle weighted posets, technically we need to consider multisets of intervals and witnesses in our algorithm. We assume $\mathcal{I} = \{I_1, \ldots, I_k\}$ may contain the same interval more than once and the same may happen to the set of witnesses. The next lemma shows that if the witnesses are pairwise independent as a weighted set instead of a multiset, then the solution is optimal.

LEMMA 3.1. *Consider a cover $\mathcal{I} = \{I_1, \ldots, I_k\}$ and a tight element $u_i \in I_i$ for every i. If u_i and u_j are either independent or $u_i = u_j$ for every i, j, then the elements $\{u_1, \ldots, u_k\}$ give a dual-optimal solution, and hence $\mathcal{I}$ is an optimal cover.*

PROOF. It suffices to show that, if for a poset element y, there exists an i with $y = u_i$, then there exist exactly $p(y)$ such intervals I_j with $y = u_j$. Since $y = u_i$ is tight, there are exactly $p(y)$ intervals I_j with $y \in I_j$. Consider such an u_j now: u_i and u_j are either independent or $u_i = u_j$, but the first case is impossible since both of them are covered by I_j. Hence $u_j = u_i$ for all $p(y)$ values of j. □

3.1. THE PUSHDOWN STEP. Our Algorithm PUSHDOWN-REDUCE tries to push witnesses down along their intervals in iterations $t = 1, 2, \ldots$ until they satisfy the requirements of Lemma 3.1, that is, witnesses are superscripted by the iteration value (t). Initial witnesses $u_j^{(1)}$ are maximum tight; their existence follows from Corollary 2.6.

```
Algorithm PUSHDOWN-REDUCE($\mathcal{I}$)
for $j = 1, \ldots, k$ do
    if $I_j$ has no tight elements then
        return reduced cover $\{I_i : i = 1, \ldots, j-1, j+1, \ldots, k\}$
$u_j^{(1)} \leftarrow$ maximal tight element of $I_j$
$t \leftarrow 1$
do
    for $j = 1, \ldots, k$ do
        $u_j^{(t+1)} \leftarrow$ PUSHDOWN($j, t, \mathcal{I}$)
    $t \leftarrow t + 1$
while exist $j$ such that $u_j^{(t)} < u_j^{(t-1)}$
return dual optimal solution $\{u_1^{(t)}, \ldots, u_k^{(t)}\}$

Procedure PUSHDOWN($j, t, \mathcal{I}$)
    $V \leftarrow \{x : m_j \le x \le u_j^{(t)}, x$ tight and $\forall i = 1, \ldots, k, u_i^{(t)}$ may not push $x$ down$\}$
    if $V = \emptyset$ then
        $t^* \leftarrow t$;
        return REDUCE($j, t^*, \mathcal{I}$)
    else return the maximal $x \in V$
```

Given two intervals $I_i = [m_i, M_i]$ and $I_j = [m_j, M_j]$ and two tight elements $u \in I_i$ and $v \in I_j$, we say that u may *push v down* with respect to I_i if u and v are dependent and $v \not\le M_i$. If the case set V of Procedure PUSHDOWN is nonempty, we will push v down, that is, replace it by the maximal element of V strictly below v. Notice that the definition depends on the choice of the interval I_i with $u \in I_i$; it is possible that v may push u down with respect to certain I_i and not with others. In the following, when it is clear from the context, we will omit mentioning I_i. Different scenarios when u may push v down are shown in Figure 2.

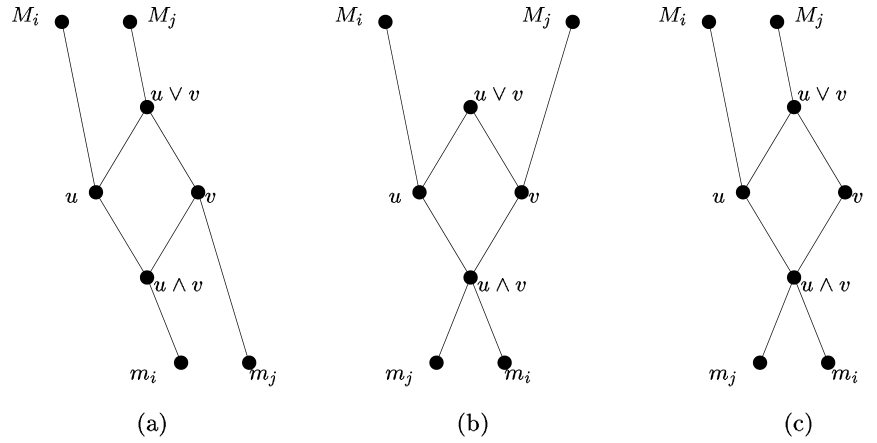

FIG. 2. Different cases when u may push v down. By Lemma 2.7, $m_i \leq v$, and there are three possible cases: (a) $m_j \not\leq u \leq M_j$, (b) $m_j \leq u \not\leq M_j$, and (c) $m_j \leq u \leq M_j$.

In what follows, we motivate which element replaces a given v when v gets pushed down. When selecting $u_j^{(t+1)}$, our aim is to replace $u_j^{(t)}$ by the maximal such tight element $x \in I_j$ which satisfies $x \leq u_j^{(t)}$, and no $u_i^{(t)}$ may push x down. As the motivation of pushing $u_j^{(t)}$ down by $u_i^{(t)}$, we give the following claim as a relatively easy consequence of Lemma 3.6; we omit the proof as it is not used elsewhere. If $u_i^{(t)}$ can push $u_j^{(t)}$ down, then for all subsequent $t' > t$ of the WHILE loop of Algorithm PUSHDOWN-REDUCE, if the witnesses $u_j^{(t')}$ and $u_i^{(t')}$ are dependent, then they must be equal. This will be the main reason why all nonequal dependent pairs of witnesses gradually disappear from the system.

While the this motivation considers the dual solution, namely, it shows that the set of witnesses will satisfy the optimality requirements, we may also give a primal motivation of pushing v down by u. If u is maximum tight in I_i, then we hope that, by replacing $[m_i, M_i]$ by $[m_i, M_j]$, we still get a cover. In the examples of Figure 2, this holds for cases (a) and (c). In this cover, v is contained in the new interval, while it was not contained in the old, thus it may be replaced by a smaller witness.

While in cases (a) and (c) of Figure 2 one could prove that, if u pushes v down, then $u \leq M_j$ (as in Benczúr [2003] for the case of increasing connectivity by one), in case (b), the argument fails since we may have $u \notin [m_i, M_j]$, and the actual proof of correctness will use a slightly more complicated argument. This is the main reason why the analysis is significantly harder than in the case of unweighted poset covers. While the argument for replacing $[m_i, M_i]$ by $[m_i, M_j]$ fails, we still push v down and proceed with the algorithm. Then we use a backward analysis as in Benczúr [2003]. In the weighted case it turns out that, while this fails to hold in general, if a particular interval exchange is performed corresponding to a pushdown step, then the exchange is valid, and, in particular, we have $u \leq M_j$. We prove this later in Lemma 3.9.

The next properties of elements where one pushes the other down are required both for the definition of the algorithm and later for the proof of correctness.

LEMMA 3.2. *If $u, u' \in I_i$ and $v \in I_j$ are tight with $u' \leq u$ and u may push v down, then u' may also push v down.*

PROOF. We only have to show that u' and v are dependent. $v \not\leq M_i$, since u may push v down. Now by Lemma 2.7, we have $m_i \leq v$. Hence the dependence of u' and v follows: a common lower bound is m_i, and a common upper bound is $u \vee v$. □

LEMMA 3.3. *Suppose $u \in I_i$, $v \in I_j$, $v' \in I_h$ are tight elements and v and v' are dependent. If u may push $v \vee v'$ down, then it may also push either v or v' down.*

PROOF. Since u may push $v \vee v'$ down, we have $v \vee v' \not\leq M_i$, hence by Lemma 2.7, we have $m_i \leq v \vee v'$. By the strong interval property, either $m_i \leq v$ or $m_i \leq v'$. By symmetry, let us consider the first case; in this case, v and u are also dependent since their common lower bound is m_i, and their common upper bound is $u \vee (v \vee v')$. If $v \not\leq M_i$, then u may push v down. Suppose now $m_i \leq v \leq M_i$. Since u may push $v \vee v'$ down, we have $v \vee v' \not\leq M_i$ and thus $v' \not\leq M_i$. Then by applying Lemma 2.7 for v, v' and $[m_i, M_i]$, it follows that $m_i \leq v'$, hence u and v' are dependent. Finally by $v' \not\leq M_i$ we get that u may push v' down. □

The actual change of a witness $u_j^{(t)}$ is performed in Procedure PUSHDOWN. We select all tight elements $x \in I_j$, $x \leq u_j^{(t)}$ into a set V that cannot be pushed down with elements $u_i^{(t)}$. If V is nonempty, we next show that it has a unique maximal element; we use this element as the new witness $u_j^{(t+1)}$.

LEMMA 3.4. *In Procedure* PUSHDOWN, *either $V = \emptyset$ or else it has a unique maximal element.*

PROOF. It suffices to show that if $x, x' \in V$, then so is $x \vee x' \in V$. Obviously, $x \vee x'$ is tight and $m_j \leq x \vee x' \leq u_j^{(t)}$. Suppose now that some $u_i^{(t)}$ may push $x \vee x'$ down. By Lemma 3.3, $u_i^{(t)}$ may push either x or x' down, contradicting $x, x' \in V$. □

If we find no dependent pair of witnesses such that one pushes the other down, then we will show that the witnesses are pairwise independent or equal and thus the solution is optimal. As long as we find pairs such that one pushes the other down, in the main loop of Algorithm PUSHDOWN-REDUCE we record a possible interval endpoint change by pushing one witness lower in its interval; these changes are then unwound to a smaller cover as shown in Section 3.3.

3.2. PROOF FOR TERMINATION WITHOUT REDUCE.

We turn to the first key step in proving the correctness. We show that, if the algorithm terminates without calling Procedure REDUCE, then $u_i^{(t)}$ are pairwise independent or equal. In other words, if none of them can be pushed down by another, then the solution is optimal.

THEOREM 3.5. *If the algorithm terminates without calling Procedure* REDUCE, *then $u_i^{(t)}$ and $u_j^{(t)}$ dependent implies $u_i^{(t)} = u_j^{(t)}$.*

The theorem is an immediate consequence of the next lemma. Notice that if the algorithm terminates without calling Procedure REDUCE, then in a last iteration the while condition of Algorithm PUSHDOWN-REDUCE fails. However, then there are no pairs i and j such that $u_i^{(t)}$ may push $u_j^{(t)}$ down.

LEMMA 3.6. *Assume that $t_1 \le t_2$ and $u_i^{(t_2)}$ and $u_j^{(t_1)}$ are dependent and $u_j^{(t_1)}$ may not push $u_i^{(t_2)}$ down. Then $u_i^{(t_2)} \le u_j^{(t_1)}$.*

This lemma is used not only for proving Theorem 3.5 but also in showing the correctness of Procedure REDUCE in Section 3.3 via the next immediate corollary.

COROLLARY 3.7. *If $u_j^{(t)}$ and $u_i^{(t+1)}$ are dependent, then $u_i^{(t+1)} \le u_j^{(t)}$.*

In the proof of Lemma 3.6, we need to characterize elements that cause witness u_j to move below a certain tight element y. Assume that, for some tight $y \in I_j$ and t, we have $y \not\le u_j^{(t)}$. Since $u_j^{(1)}$ is maximal tight, we may select the unique t_0 with $y \le u_j^{(t_0)}$ but $y \not\le u_j^{(t_0+1)}$. In step PUSHDOWN$(j, t_0, \mathcal{I})$, we must have an $u_d^{(t_0)}$ that pushes y down. We will use this in the following special case.

LEMMA 3.8. *Assume that z is tight and dependent with $u_j^{(t)}$. Assume furthermore that $z \not\le u_j^{(t)}$ and $z \le M_j$. Then there exists $t_0 < t$ and d such that $u_d^{(t_0)}$ may push $u_j^{(t)} \vee z$ down. In addition, $u_d^{(t_0)}$ may also push z down.*

PROOF. We apply these observations for $y = u_j^{(t)} \vee z \in I_j$. Since y is tight, $y \le u_j^{(1)}$. And since $z \not\le u_j^{(t)}$, we get $y = u_j^{(t)} \vee z \not\le u_j^{(t)}$. We select t_0 with $y \le u_j^{(t_0)}$ but $y \not\le u_j^{(t_0+1)}$; then in step PUSHDOWN$(j, t_0, \mathcal{I})$, we must have an $u_d^{(t_0)}$ that may push y down.

For the second part of the claim, observe that by Lemma 3.3, $u_d^{(t_0)}$ may push either $u_j^{(t)}$ or z down. The first choice is impossible since then $u_d^{(t-1)}$ could also push $u_j^{(t)}$ down by Lemma 3.2, and $t - 1 \ge t_0$. This latter contradicts the choice of $u_j^{(t)}$ as the maximum tight element that may not be pushed down in PUSHDOWN $(j, t-1, \mathcal{I})$. □

PROOF OF LEMMA 3.6. $u_i^{(t_2)} \le M_j$ since $u_j^{(t_1)}$ may not push $u_i^{(t_2)}$ down. If $u_i^{(t_2)} \not\le u_j^{(t_1)}$, then the conditions of Lemma 3.8 hold with $z = u_i^{(t_2)}$ and $t = t_1$. Thus we have some $t_0 < t_1$ and d such that $u_d^{(t_0)}$ may push $z = u_i^{(t_2)}$ down. But then $u_d^{(t_2-1)}$ may also push $u_i^{(t_2)}$ down by Lemma 3.2. This latter contradicts the choice of $u_i^{(t_2)}$ as the maximum tight element that may not be pushed down in PUSHDOWN$(i, t_2 - 1, \mathcal{I})$. □

3.3. THE REDUCE STEP.

So far, we have proved that if REDUCE is not called, then the initial primal solution is optimal, and the algorithm finds a dual-optimum proof of this fact. Now we turn to the second scenario when Procedure REDUCE is called. In this case, the solution is not optimal since Procedure REDUCE is called from Procedure PUSHDOWN when $V = \emptyset$. This means $u_j^{(t)} \notin V$, and thus there exists an i such that $u_i^{(t)}$ may push $u_j^{(t)}$ down.

Procedure REDUCE is called when one witness disappears from the dual solution. In this case, we unwind the steps to find a cover of size one less in Procedure REDUCE based on interval exchanges at certain pairs of tight poset elements.

```
Procedure REDUCE-ONESTEP($j, 1, \mathcal{I}$)
    $j_1 \leftarrow j$;
    $q \leftarrow$ minimal tight element in $[m_{j_1}, M_{j_1}]$
    $j_2 \leftarrow$ minimum value $\ell \ne j_1$ such that $u_\ell^{(1)}$ may push $q$ down
  return reduced cover $\{[m_i, M_i] : 1 \le i \le k, i \ne j_1, j_2\} \cup \{[m_{j_2}, M_{j_1}]\}$.
```

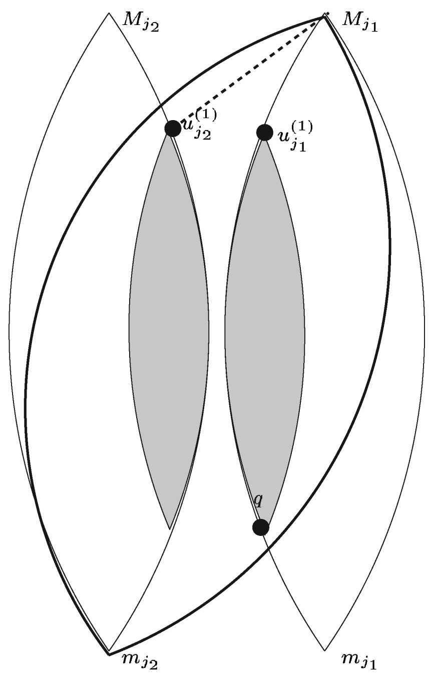

FIG. 3. Procedure REDUCE called with $t^* = 1$. The two upright intervals are the original ones with their tight elements shaded. These two intervals will be replaced by the single bold interval. The new interval contains all tight elements of the old ones since $u^{(1)}_{j_2} \leq M_{j_1}$ by Lemma 3.9. Remember that the intervals need not to be disjoint.

To illustrate the idea of Procedure REDUCE, first we discuss the simplest case $t^* = 1$; the general case will then be reduced to this case by a special induction. We summarize Procedure REDUCE-ONESTEP for this particular scenario with the steps shown in Figure 3. Since t^*=1, we have some $1 \leq j_1 \leq k$ such that Procedure REDUCE is called within Procedure PUSHDOWN($j_1, 1, \mathcal{I}$). This means that

$$V = \left\{x : m_{j_1} \leq x \leq u^{(1)}_{j_1}, x \text{ tight and } \forall \ell = 1, \ldots, k, u^{(1)}_{\ell} \text{ may not push } x \text{ down}\right\}$$

is empty. By Corollary 2.6, $[m_{j_1}, M_{j_1}]$ has a unique minimal tight element q. Since $q \notin V$, we must have some $\ell = j_2$ such that $u^{(1)}_{\ell}$ may push q down. Given an ordering over the intervals, the algorithm selects j_2 as the minimal such ℓ and returns a reduced interval system

$$\mathcal{I} - [m_{j_1}, M_{j_1}] - [m_{j_2}, M_{j_2}] + [m_{j_2}, M_{j_1}]. \tag{3}$$

In the proof of case t^*=1, we use the following general lemma for $h = j_1, \ell = j_2$, $u = u^{(1)}_{j_2}$.

LEMMA 3.9. *Let q be the minimal tight element of I_h. If $u \in I_\ell$ may push q down, then $u \leq M_h$. Furthermore, for all tight $v \in I_h$, we have that u may push v down with respect to I_ℓ.*

PROOF. Suppose by contradiction that $u \not\leq M_h$. Since u and q are dependent, by Lemma 2.7, $u \wedge q \in I_h$. Since q is the minimal tight in I_h, we have $q \leq u \wedge q$, hence $q \leq u \leq M_\ell$, contradicting that u may push q down. For the second part of the claim, consider a tight element $v \in I_h$. Elements u and v are dependent since common lower and upper bounds are $u \wedge q$ and M_h, respectively. By $q \leq v$ and $q \not\leq M_\ell$, the required $v \not\leq M_\ell$ follows. □

LEMMA 3.10. *If $t^* = 1$, Procedure* REDUCE$(j_1, t^*, \mathcal{I})$ *returns an interval cover.*

PROOF. It suffices to show that $[m_{j_2}, M_{j_1}]$ contains all tight elements of both $[m_{j_1}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$; furthermore, there is no common tight element in $[m_{j_1}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$. In this case, we may replace the intervals $[m_{j_1}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$ by $[m_{j_2}, M_{j_1}]$ since, if a tight element is contained by exactly one of $[m_{j_1}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$, then it is contained by the new interval and containment by both is excluded.

To prove this, first let $x \in [m_{j_2}, M_{j_2}]$ be tight; $x \leq u^{(1)}_{j_2}$ by maximality. When applying Lemma 3.9 for $h = j_1$, $\ell = j_2$, $u = u^{(1)}_{j_2}$, we get $u^{(1)}_{j_2} \leq M_{j_1}$. This implies $m_{j_2} \leq x \leq u^{(1)}_{j_2} \leq M_{j_1}$ as required.

Next let $x \in [m_{j_1}, M_{j_1}]$ be tight; $q \leq x$ for the minimal tight q of $[m_{j_1}, M_{j_1}]$. By the second part of Lemma 3.9 applied with $\ell = j_2$, we have that $u^{(1)}_{j_2}$ may push q down. By Lemma 2.7, $m_{j_2} \leq q$, thus we get $m_{j_2} \leq q \leq x \leq M_{j_1}$ as required.

Finally assume that a common tight element $x \in [m_{j_1}, M_{j_1}] \cap [m_{j_2}, M_{j_2}]$ exists; now $q \leq x \leq M_{j_2}$, contradicting the fact that $u^{(1)}_{j_2}$ may push q down. □

```
Procedure REDUCE($j, t^*, \mathcal{I}$)
  $j_1 \leftarrow j$;
  **for** $t = t^*, \ldots, 1$ **do**
    $s \leftarrow t^* + 1 - t$
    $q \leftarrow$ minimal tight element in $[m_{j_s}, M_{j_s}]$
    $j_{s+1} \leftarrow$ **minimum** value $\ell \neq j_s$ such that $u^{(t)}_\ell$ may push $q$ down
    $m_{j_s} \leftarrow m_{j_{s+1}}$
**return** reduced cover $\{[m_i, M_i] : 1 \leq i \leq k, i \neq j_{t^*+1}\}$.
```

Our aim in Procedure REDUCE is to repeatedly pick an interval $[m_{j_s}, M_{j_s}]$ and try to find another interval $[m_{j_{s+1}}, M_{j_{s+1}}]$ such that if we replace $[m_{j_s}, M_{j_s}]$ by $[m_{j_{s+1}}, M_{j_s}]$, then, after the switch, the minimum tight element of $[m_{j_{s+1}}, M_{j_{s+1}}]$ increases. We ensure this by defining

$$j_{s+1} \leftarrow \text{ minimum value } \ell \neq j_s \text{ such that } u^{(t)}_\ell \text{ may push } q \text{ down},$$

where q is the minimum tight element of $[m_{j_s}, M_{j_s}]$ after the intervals change and $t = t^* + 1 - s$. Applying Lemma 3.9 for $h = j_s$, $\ell = j_{s+1}$, $u = u^{(t)}_{j_{s+1}}$, we get $u^{(t)}_{j_{s+1}} \leq M_{j_s}$. Thus when replacing $[m_{j_s}, M_{j_s}]$ by $[m_{j_{s+1}}, M_{j_s}]$, the tight elements x

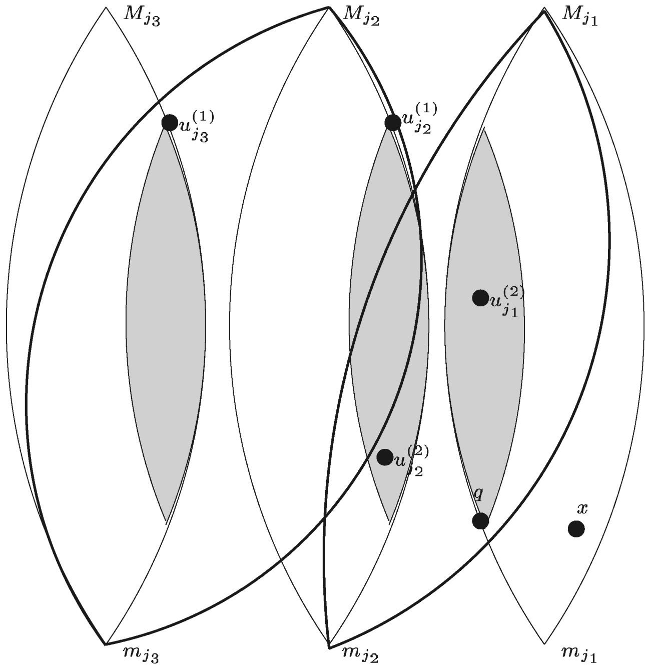

FIG. 4. Procedure REDUCE called with $t^* = 2$. The three upright intervals are the original ones with their tight elements shaded. The original three intervals will be replaced by the two bold intervals using the marked witnesses. Note that the two new intervals contain all tight elements of the old ones. While the number of intervals covering certain nontight elements (x in the example) may decrease, we prove that they remain covered. Note that the original intervals may not be disjoint.

in $[m_{j_{s+1}}, M_{j_{s+1}}]$ with $x \leq u_{j_{s+1}}^{(t)}$ will no longer be tight after the switch. The overall idea is seen in Figure 4.

While the first step of the procedure is well-defined since we call Procedure REDUCE exactly when the minimal tight $q \in I_j$ for $j = j_1$ is pushed down by certain other $u_\ell^{(t^*)}$, the existence of such an ℓ is by no means obvious for all the other iterations of the main loop as switches among the intervals could completely rearrange the set of the tight elements.

The existence of all further ℓ in Procedure REDUCE as well as the correctness of the algorithm is proved by 'rewinding' the algorithm after the first iteration of Procedure REDUCE and showing that each step is repeated identically up to iteration $t^* - 1$. The intuition behind rewinding is based on the resemblance of Procedure REDUCE to an augmenting path algorithm. In this terminology, instead of directly proving augmenting path properties, we use a special induction by executing the main loop of the procedure step-by-step and, after each iteration, rewinding the main algorithm. In the analogy of network flow algorithms, this may correspond to analyzing an augmenting path algorithm by choosing path edges starting at the source, changing the flow along this edge to a preflow, and, at each step, proving that the remaining path augments the flow.

The key Theorem that follows will show, by induction on the value t^* of t at the termination of the main loop of Algorithm PUSHDOWN-REDUCE, that the intermediate modified interval sets are covers for $t^*, t^* - 1, \ldots, 1$. Finally, when applied for $t^* = 1$, we get that Procedure REDUCE finds an interval cover of size one less than before by Lemma 3.10. This completes the correctness analysis of Procedure REDUCE. Before stating the Theorem, we define the intermediate modified interval set $\mathcal{I}'$ and show it is a cover.

LEMMA 3.11. *Let*

$$\mathcal{I}' = \mathcal{I} - [m_{j_1}, M_{j_1}] + [m_{j_2}, M_{j_1}] \tag{4}$$

be the set of intervals after the first iteration of Procedure REDUCE. *Then $\mathcal{I}'$ is a cover.*

PROOF. Since $u^{(1)}_{j_2}$ may push q down, $q \not\leq M_{j_2}$, thus, by Claim 2.7, $m_{j_2} \leq q$ and so $[m_{j_2}, M_{j_1}]$ contains all tight elements of $[m_{j_1}, M_{j_1}]$. □

THEOREM 3.12. *For $t^* > 1$, Algorithm* PUSHDOWN-REDUCE *performs the exact same steps with inputs $\mathcal{I}$ and $\mathcal{I}'$ of Lemma 3.11 until iteration $t^* - 1$ when* REDUCE$(j_2, t^* - 1, \mathcal{I}')$ *is called. Hence compared to $\mathcal{I}$, the main loop of Algorithm* PUSHDOWN-REDUCE *terminates one step earlier with $t = t^* - 1$ when run with $\mathcal{I}'$.*

To prove Theorem 3.12, we now define elements that are no longer tight and elements that become tight in the new cover:

LEMMA 3.13. *Let*

$$\begin{aligned} Z_1 &= \{x \text{ tight in } \mathcal{I} \text{ and } x \text{ not tight in } \mathcal{I}'\}, \\ Z_2 &= \{x \text{ not tight in } \mathcal{I} \text{ and } x \text{ tight in } \mathcal{I}'\}. \end{aligned}$$

Then

$$Z_1 \subseteq \{x : x \in [m_{j_2}, M_{j_1}], x \not\geq m_{j_1}\} \tag{5}$$

$$Z_2 \subseteq \{x : x \in [m_{j_1}, M_{j_1}], x \not\geq m_{j_2}\}. \tag{6}$$

Hence the same elements are tight in I_{j_1} for $\mathcal{I}$ as in $[m_{j_2}, M_{j_1}]$ for $\mathcal{I}'$.

PROOF. We get $\mathcal{I}'$ from $\mathcal{I}$ by removing $[m_{j_1}, M_{j_1}]$ and adding $[m_{j_2}, M_{j_1}]$ instead. Hence the elements of Z_1 should be contained in the latter but not in the former, and similarly the elements of Z_2 should be in the former but not in the latter interval. □

Next we show that the algorithm proceeds identically for $\mathcal{I}$ and $\mathcal{I}'$ for $t < t^*$. The proof is based on the fact that the key elements used in defining $u^{(t)}_i$ do not belong to $Z_1 \cup Z_2$.

LEMMA 3.14. *Let $u'^{(t)}_i$ denote elements selected by Algorithm* PUSHDOWN-REDUCE *with input $\mathcal{I}'$ with the convention that $u'^{(t)}_{j_1}$ belongs to the modified interval $I'_{j_1} = [m_{j_2}, M_{j_1}]$. Then for all $t < t^*$, we have $u^{(t)}_i = u'^{(t)}_i$.*

PROOF. By induction on $t \leq t^* - 1$, we will show $u_i{'}^{(t)} = u^{(t)}_i$. We prove the inductive hypothesis in three steps: we show for $i = 1, \ldots, k$ that

(i) $u^{(t)}_i \notin Z_1$;

(ii) $u'^{(t)}_i$ exists; and

(iii) $u'^{(t)}_i \notin Z_2$.

These three statements imply $u_i'^{(t)} = u_i^{(t)}$ as follows. For $t = 1$, the maximal tight elements are identical for $i \neq j_1$ by (i) and (iii), since $u_i'^{(1)}$ tight in $\mathcal{I}$ implies $u_i'^{(1)} \leq u_i^{(1)}$, and we have the opposite inequality when exchanging the role of the two elements. Also $u_{j_1}'^{(1)} = u_{j_1}^{(1)}$ since, by Lemma 3.13, the tight elements of I_{j_1} in $\mathcal{I}$ are the same as the tight elements of I'_{j_1} in $\mathcal{I}'$. For general t by induction on the step of defining $u_i'^{(t)}$, one can observe that element $u_i^{(t)}$ belongs to the set V of Procedure PUSHDOWN$(i-1, t, \mathcal{I}')$, and the same holds when exchanging the role of $u_i'^{(t)}$ and $u_i^{(t)}$. Thus the two elements must be equal.

Now we prove (i–iii). First of all, for $i = j_1$, the tight elements of I_{j_1} in $\mathcal{I}$ are the same as those of I'_{j_1} in $\mathcal{I}'$ by Lemma 3.13, yielding (i–iii). Hence we assume $i \neq j_1$ next.

PROOF OF (I). Assume $u_i^{(t)} \in Z_1$. By Lemma 3.13, $m_{j_2} \leq u_i^{(t)} \leq M_{j_1}$ and $m_{j_1} \not\leq u_i^{(t)}$. Furthermore, since $m_{j_2} \leq u_{j_1}^{(t^*)} \leq u_{j_1}^{(t+1)} \leq M_{j_1}$, we have $u_i^{(t)}$ and $u_{j_1}^{(t+1)}$ dependent. Using Corollary 3.7, $u_{j_1}^{(t+1)} \leq u_i^{(t)}$, thus $m_{j_1} \leq u_i^{(t)}$, a contradiction. □

PROOF OF (II). We show that $u_i'^{(t)}$ exists and $m_i \leq u_i^{(t)} \leq u_i'^{(t)}$. We proved that $u_i^{(t)} \notin Z_1$ and hence $u_i^{(t)}$ remains tight in $\mathcal{I}'$. This immediately gives the result for $t = 1$. And for $t > 1$, we use the consequence of the inductive hypothesis that $u_h^{(t-1)} = u_h'^{(t-1)}$ for all h. This yields $u_i^{(t)} \in V$ for PUSHDOWN$(i, t-1, \mathcal{I}')$ that, in turn, implies that $u_i'^{(t)}$ exists and $u_i^{(t)} \leq u_i'^{(t)}$. □

PROOF OF (III). Assume $u_i'^{(t)} \in Z_2$. By Lemma 3.13, $m_{j_1} \leq u_i'^{(t)} \leq M_{j_1}$, thus $u_i'^{(t)}$ and $u_{j_1}^{(t+1)}$ are dependent. Observe furthermore that $u_{j_1}^{(t+1)} \notin Z_1$, thus also tight in $\mathcal{I}'$. Hence by applying Lemma 2.7 for $\mathcal{I}'$, we get that either $u_{j_1}^{(t+1)} \leq M_i$ or $m_i \leq u_{j_1}^{(t+1)}$. In both cases, we derive a contradiction with the definition of $u_{j_1}^{(t+1)}$ in Procedure PUSHDOWN$(j_1, t, \mathcal{I})$ by showing that certain $u_d^{(t)}$ may push $u_{j_1}^{(t+1)}$ down.

Case 1. $u_{j_1}^{(t+1)} \leq M_i$. By Lemma 3.13, we also get $m_{j_2} \not\leq u_i'^{(t)}$, which, in turn, implies $u_{j_1}^{(t+1)} \not\leq u_i'^{(t)}$ since $m_{j_2} \leq u_{j_1}^{(t+1)}$. Because $u_{j_1}^{(t+1)}$ is tight in $\mathcal{I}'$ and $u_{j_1}^{(t+1)} \leq M_i$, we apply Lemma 3.8 for $\mathcal{I}'$, $u_i'^{(t)}$ and $z = u_{j_1}^{(t+1)}$. By the Lemma, there exists $t_0 < t$ and $1 \leq d \leq k$ such that the element $u_d'^{(t_0)}$ may push $u_{j_1}^{(t+1)}$ down. By induction $u_d^{(t_0)} = u_d'^{(t_0)}$ and by Lemma 3.2, $u_d^{(t)}$ may also push $u_{j_1}^{(t+1)}$ down.

Case 2. $m_i \leq u_{j_1}^{(t+1)}$ and $u_{j_1}^{(t+1)} \not\leq M_i$. As we have shown, $m_i \leq u_i^{(t)} \leq u_i'^{(t)}$. Thus $u_{j_1}^{(t+1)}$ and $u_i^{(t)}$ are dependent since their common lower and upper bounds are m_i and M_{j_1}, respectively. Hence, in this case, we have $d = i$: element $u_i^{(t)}$ may push $u_{j_1}^{(t+1)}$ down. The proof is complete. □

We complete the proof of Theorem 3.12 by the following lemma.

LEMMA 3.15. *When run with input $\mathcal{I}'$, Procedure* REDUCE *is called in iteration $t^* - 1$ with $j = j_2$.*

PROOF. By Lemma 3.14, Procedure REDUCE cannot be called for $\mathcal{I}'$ before iteration $t^* - 1$. Two things are left to prove: (i) in iteration $t^* - 1$, REDUCE$(h, t^* - 1, \mathcal{I}')$ is not called for any $h < j_2$ and (ii) REDUCE$(j_2, t^* - 1, \mathcal{I}')$ is called.

To prove (i), assume by contradiction that REDUCE$(h, t^* - 1, \mathcal{I}')$ is called for some $h < j_2$, or equivalently, $V = \emptyset$ in PROCEDURE$(h, t^* - 1, \mathcal{I}')$. We show that $u_h^{(t^*)} \in Z_1$. Indeed, by Lemma 3.14, $u_h^{(t^*-1)} = u_h'^{(t^*-1)}$ for all h. Since no $u_h^{(t^*-1)}$ may push $u_{j_2}^{(t^*)}$ down, this yields that, if $u_h^{(t^*)} \notin Z_1$, then $u_h^{(t^*)} \in V$, contradicting the assumption $V = \emptyset$.

By Lemma 3.13, $m_{j_2} \leq u_h^{(t^*)} \leq M_{j_1}$, thus q and $u_h^{(t^*)}$ are dependent. Element $u_h^{(t^*)}$ may not push q down because it would contradict the fact that $\ell = j_2$ is minimal in a fixed ordering of the intervals so that $u_\ell^{(t^*)}$ may push q down. This means that $q \leq M_h$. In addition, $q \not\leq u_h^{(t^*)}$ since $m_{j_1} \leq q$ and $m_{j_1} \not\leq u_h^{(t^*)}$ by $u_h^{(t^*)} \in Z_1$. We can apply Lemma 3.8 for $u_h^{(t^*)}$ and $z = q$, which implies the existence of some $t_0 < t^*$ and $1 \leq d \leq k$ so that $u_d^{(t_0)}$ may push q down. By the second part of Lemma 3.9, $u_d^{(t_0)}$ may also push $u_{j_1}^{(t_0+1)}$ down, a contradiction.

For (ii), suppose by contradiction that $u_{j_2}'^{(t^*)}$ exists. Since $u_{j_2}'^{(t^*)} \geq m_{j_2}$, by Lemma 3.13, $u_{j_2}'^{(t^*)} \notin Z_2$, hence $u_{j_2}'^{(t^*)}$ is also tight in $\mathcal{I}$. We use again that by Lemma 3.14, $u_h^{(t^*-1)} = u_h'^{(t^*-1)}$ for all h. This yields $u_{j_2}'^{(t^*)} \in V$ for PUSHDOWN$(j_2, t^* - 1, \mathcal{I})$, implying $u_{j_2}'^{(t^*)} \leq u_{j_2}^{(t^*)}$. By making use of Lemma 3.9, $u_{j_2}'^{(t^*)} \leq u_{j_2}^{(t^*)} \leq M_{j_1}$.

We claim that $u_{j_2}'^{(t^*)} \in Z_1$, contradicting the fact that $u_{j_2}'^{(t^*)}$ is tight in $\mathcal{I}'$. As $m_{j_2} \leq u_{j_2}'^{(t^*)} \leq M_{j_1}$ and $u_{j_2}'^{(t^*)}$ is tight in $\mathcal{I}$, all we need to show is $m_{j_1} \not\leq u_{j_2}'^{(t^*)}$. Assume $m_{j_1} \leq u_{j_2}'^{(t^*)}$. This implies $m_{j_1} \leq u_{j_2}'^{(t^*)} \leq M_{j_1}$, thus $q \leq u_{j_2}^{(t^*)}$ as q is the minimal tight element of $[m_{j_1}, M_{j_1}]$ in $\mathcal{I}$. In this case, $u_{j_2}^{(t^*)}$ may not push q down, contradicting the selection of j_2 in Procedure REDUCE$(j, t^*, \mathcal{I})$. $\square$

4. *Connectivity Augmentation*

In this section, we give a reformulation of the previous general algorithm which is applicable for the problem of directed vertex connectivity augmentation. The main difficulty is that we typically have an exponential size poset implicitly given as a set of (directed) cuts. We may either select an appropriate poset representation or implement the steps of the algorithm with direct reference to the underlying graph problem. We follow the second approach. We will show how all nontrival steps of the algorithm can be reduced to determining maximal tight elements in certain interval covers, which can be implemented as a sequence of BFS computations using some initial flow computations.

The key step in implementing Procedure PUSHDOWN for the underlying graph problems is the following reformulation of the main algorithm. We replace Procedure PUSHDOWN by an iterative method Procedure ALTERNATE-PUSHDOWN that selects a strictly descending sequence of tight elements $y_0 > y_1 > \cdots > y_\ell$ with $y_0 = u_j^{(t)}$ and $y_\ell = u_j^{(t+1)}$ or terminates by Procedure REDUCE$(j, t^*, \mathcal{I})$. In the implementation for graph augmentation problems, it is key to notice that, in a single iteration of Procedure ALTERNATE-PUSHDOWN, we only consider elements that may be pushed down by $u_i^{(t)}$ for a single value of i.

```
Procedure ALTERNATE-PUSHDOWN($j, t, \mathcal{I}$)
    $y_0 \leftarrow u_j^{(t)}$; $h \leftarrow 0$;
    **while** exists $i$ such that $u_i^{(t)}$ may push $y_h$ down **do**
        $V_h \leftarrow \{x : m_j \leq x \leq y_h, x$ tight and $u_i^{(t)}$ may not push $x$ down$\}$
        **if** $V_h = \emptyset$ **then**
            $t^* \leftarrow t$;
            **return** REDUCE($j, t^*, \mathcal{I}$)
        **else**
            $y_{h+1} \leftarrow$ maximal $x \in V_h$;
            $h \leftarrow h + 1$
    **return** $y_h$
```

LEMMA 4.1. *Procedure* ALTERNATE-PUSHDOWN *returns the same output as Procedure* PUSHDOWN.

PROOF. It follows straightforward from Lemma 3.3 that if $V_h \neq \emptyset$, then it has a unique maximal element, hence y_h for $h \geq 1$ is well defined.

If Procedure ALTERNATE-PUSHDOWN terminates by returning y_h for some h, then $y_h \in V$ for V as in Procedure PUSHDOWN. Thus $y_h \leq u_j^{(t+1)}$. This shows that if Procedure PUSHDOWN terminates by calling Procedure REDUCE, then so does Procedure ALTERNATE-PUSHDOWN.

Consider now the case when $V \neq \emptyset$ in Procedure PUSHDOWN. We show that $y_h \geq u_j^{(t+1)}$ for each $h \geq 0$. By contradiction, choose the smallest h with $y_h \not\geq u_j^{(t+1)}$; thus $y_{h-1} \geq y_h \vee u_j^{(t+1)} > y_h$. By the definition of V_{h-1}, $u_i^{(t)}$ may push $y_h \vee u_j^{(t+1)}$ down for some i. Using Lemma 3.3 again, it may push either y_h or $u_j^{(t+1)}$ down, both leading to contradiction. Now we can conclude that if Procedure ALTERNATE-PUSHDOWN terminates by returning y_h, then both $y_h \leq u_j^{(t+1)}$ and $y_h \geq u_j^{(t+1)}$ hold, thus they are equal. □

To compute y_h consider the set of intervals $\mathcal{J}_{j,i} = \mathcal{I} - [m_i, M_i] + [m_i, M_j]$ with i as in Procedure ALTERNATE-PUSHDOWN. While $\mathcal{J}_{j,i}$ is not necessarily a cover of the entire poset, the following lemmas still hold.

LEMMA 4.2. *All $x \in V_h$ are tight in $\mathcal{J}_{j,i}$.*

PROOF. Notice x is either contained in both intervals $[m_i, M_i]$ and $[m_i, M_j]$ or in neither of them: if $m_i \leq x$, then x and $u_i^{(t)}$ are dependent because m_i is a common lower and $u_i^{(t)} \vee y_h$ a common upper bound. Hence $x \leq M_i$, since $u_i^{(t)}$ may not push x down. □

LEMMA 4.3. *Suppose $u_i^{(t)}$ may push y_h down. The set of intervals $\mathcal{J}_{j,i}$ covers all elements of I_j, furthermore, $y_{h+1} = y_h \wedge Q$, where Q is the maximal tight element of I_j in $\mathcal{J}_{j,i}$.*

PROOF. For all $x \in I_j$, we have $x \in [m_i, M_j]$ if $x \in [m_i, M_i]$, hence the number of intervals covering x cannot be less in $\mathcal{J}_{j,i}$ than in $\mathcal{I}$, thus $\mathcal{J}_{j,i}$ covers all elements of I_j.

For the second part, we first show that if I_j has any tight elements for $\mathcal{J}_{j,i}$, then there is a unique maximal among them. We cannot apply Lemma 2.5 directly since $\mathcal{J}_{j,i}$ is not a cover, but the claim holds for any $x, y \in I_i$ since x, y, $x \vee y$, and $x \wedge y$ are all covered by $\mathcal{J}_{j,i}$. Hence the existence of the unique maximal tight element follows. Since any element of I_i is covered in $\mathcal{J}_{j,i}$ by at least as many intervals as in $\mathcal{I}$, Q is also tight in $\mathcal{I}$.

Finally we let $z = y_h \wedge Q$ and show $z = y_{h+1}$. Notice that z is tight in $\mathcal{I}$ as it is an intersection of two tight elements in $\mathcal{I}$. As $y_{h+1} \leq y_h$ and y_h is tight in $\mathcal{J}_{j,i}$ by Lemma 4.2, we get $y_{h+1} \leq Q$, and thus $y_{h+1} \leq y_h \wedge Q = z$. For $z \leq y_{h+1}$, we have to prove that $u_i^{(t)}$ may not push z down. Indeed, suppose that $u_i^{(t)}$ may push z down. Then $m_i \leq z \not\leq M_i$, hence by $z \leq Q$ follows $Q \in [m_i, M_j]$. As Q is tight in $\mathcal{J}_{j,i}$, this implies that $Q \in [m_i, M_i]$, thus $z \leq Q \leq M_i$, a contradiction. □

By the lemma, the basic step of Procedure ALTERNATE-PUSHDOWN consists of computing the maximum tight element of an interval for a certain set of covering intervals. Furthermore, at the beginning of the algorithm, $u_j^{(1)}$ is the maximum tight element of I_j. Now we turn our attention to the implementation of the steps of the algorithm for connectivity augmentation. We use the reduction of vertex connectivity augmentation to poset covering as in Theorem 2.3. The minimal elements correspond to set pairs having a singleton tail and all the other vertices as head; maximal elements are found by exchanging the role of tails and heads. For each interval $I = [m_i, M_i] \in \mathcal{I}$, we augment the graph by an edge $s_i t_i$ with s_i corresponding to m_i and t_i corresponding to M_i as in the previous reduction. If $\mathcal{I}$ covers all poset elements in $[m_i, M_i]$, then the minimum s_i–t_i cut in the augmented graph has value at least k.

Algorithm PUSHDOWN-REDUCE($\mathcal{I}$) will first be applied for a greedy cover $\mathcal{I}$ (e.g., including all possible intervals), and then subsequently for covers of decreasing cardinality, until we finally reach an optimal cover. We initialize PUSHDOWN-REDUCE($\mathcal{I}$) by computing $|\mathcal{I}|$ maximum flows, one corresponding to each interval in $\mathcal{I}$. For interval $[m_j, M_j]$, we compute a maximum s_j–t_j flow. Since $\mathcal{I}$ is a cover, the maximum flow value is at least k. If the s_j–t_j flow value is more than k, then $[m_j, M_j]$ contains no tight elements, and thus can be removed from the cover, and the iteration PUSHDOWN-REDUCE($\mathcal{I}$) is finished. Otherwise $u_j^{(1)}$ is the set pair corresponding to the value k cut with maximal tail that can be obtained by a breadth-first search from t_j on the graph obtainded from the standard auxiliary graph in the Ford-Fulkerson algorithm by reverting the edges.

LEMMA 4.4. *Consider the task of finding the maximum tight element of an interval $I_j = [m_j, M_j]$ for a certain set of intervals $\mathcal{J}_{j,i}$ (e.g., in Procedure* ALTERNATE-PUSHDOWN) *that cover I_j. Using the maximum s_j–t_j flow computed at the initialization for I_j, this step requires $O(1)$ breadth-first search (BFS) computations.*

PROOF. Consider the maximum s_j–t_j flow computed at the initialization. We add an edge $s_i t_j$ to the graph and remove the edge $s_i t_i$. If the flow contains the removed edge, then we remove the single flow path containing it. We augment the resulting flow to a maximum flow by a single BFS computation. By another BFS starting from t_j, we either obtain the maximum tight element or deduce that there are no tight elements and Procedure REDUCE can be called. □

For implementing REDUCE, we need to find minimal tight elements of certain intervals and a sequence of changes in the interval cover by adding an interval and removing an other. The first step can be preformed by a BFS computation from the corresponding s_i; for the second step, we need to update the flows corresponding to the intervals $[m_j, M_j] \in \mathcal{I}$. For each $[m_j, M_j]$ in iteration s, we consider the maximum s_j–t_j flow, add an edge $s_{j_{s+1}} t_{j_s}$ to the graph and remove the edge $s_{j_s} t_{j_s}$. Again, if the flow contains the removed edge, then we remove the single flow path containing it and augment the flow by a BFS computation.

5. *Running Times*

To estimate the running time, we need bounds for the number of intervals j and the length of a longest chain ℓ in the poset. At the initialization of PUSHDOWN-REDUCE, we preform j max-flow computations, then the dominating steps are finding elements y_h in Procedure ALTERNATE-PUSHDOWN. Since computing meets $\vee$ and intersections $\wedge$ of elements as well as checking whether $u \in I_i$ may push v down can be done in $O(1)$ time, this step is dominated by $O(1)$ BFS computations by Lemma 4.4.

Between two calls to Procedure REDUCE the total number of iterations in all calls to ALTERNATE-PUSHDOWN that compute certain y_h can be bounded by $j \cdot \ell$ since in each step we find a strictly smaller element of certain interval. This totals to $O(j \cdot \ell)$ BFS computations. For an iteration of REDUCE, we also have to do $O(j \cdot \ell)$ BFS computations. The total number of calls to Algorithm PUSHDOWN-REDUCE is bounded by j since the number of intervals decreases in each iteration. Hence we have $O(j^2)$ maximum flow and $O(j^2 \cdot \ell)$ BFS computations.

For vertex connectivity augmentation problems $\ell = O(n)$ and $j = O(n^2)$ since adding a complete digraph surely gives an $(n-1)$-connected digraph. Thus by the previous estimations, the running time is dominated by $O(n^5)$ BFS computations and $O(n^4)$ Max Flow Computations. As a BFS can be computed in time $O(n^2)$ and a Max Flow in time $O(n^3)$, the total running time can be bounded by $O(n^7)$.

6. *Conclusion*

We have given a combinatorial algorithm for covering posets satisfying a special property by the minimal number of intervals of the poset. As noted by Frank and Jordán [1995], the result can be applied for certain directed edge augmentation problems. While we have given a detailed algorithm only for the directed vertex connectivity augmentation, the other known applications (e.g., $S{-}T$-edge or vertex connectivity augmentation) can be implemented similarly. The existence of a strongly polynomial combinatorial algorithm, however, remains open. Another major open problem regards the complexity of undirected augmentation; the best known result is a polynomial algorithm which finds an optimal solution for every fixed k [Jackson and Jordán 2005].

One may wonder how strong the generalizational power of the interval covering problem is. Two algorithmically equivalent problems, Dilworth's chain cover and bipartite matching, are special cases of interval covers; our algorithm generalizes the standard augmenting path-matching algorithm. One may ask whether the network flow problem as different algorithmic generalization of matchings could also fit into our framework. Or, extending the question of Karger and Levine [1998], can we at least tell the hierarchy of hardness of the interval cover, Dilworth, (bipartite) matching and maximum flow problems? We might also hope that ideas such as capacity scaling, distance labeling, and preflows [R.K. Ahuja and Orlin 1993] that give polynomial algorithms for network flows can be used in the construction of a strongly polynomial algorithm for the interval covering problem.

Finally one may be interested in the efficiency of our algorithm for the particular problems that can be handled. Here particular implementations and good oracle choices are needed. We may want to reduce the number of mincut computations needed by polynomial size poset representations. One might also be able to give improvements in the sense of the Hopcroft–Karp matching algorithm [1973].

ACKNOWLEDGMENT. We are grateful to the anonymous referees for their thorough work in finding omissions in earlier versions as well as their suggestions to improve presentation.

REFERENCES

BENCZÚR, A. A. 1999. Parallel and fast sequential algorithms for undirected edge connectivity augmentation. *Math. Prog. B 84,* 3, 595–640.

BENCZÚR, A. A. 2003. Pushdown-reduce: An algorithm for connectivity augmentation and poset covering problems. *Discrete Appl. Math. 129,* 2–3, 233–262.

BENCZÚR, A. A., FÖRSTER, J., AND KIRÁLY, Z. 1999. Dilworth's theorem and its application for path systems of a cycle—implementation and analysis. In *Proceedings of the 7th Annual European Symposium on Algorithms (ESA'99)*. Springer-Verlag, Berlin, Germany, 498–509.

BENCZÚR, A. A. AND KARGER, D. R. 2000. Augmenting undirected edge connectivity in $\tilde{o}(n^2)$ time. *J. Algo. 37,* 1, 2–36.

CAI, G.-P. AND SUN, Y.-G. 1989. The minimum augmentation of any graph to a k-edge-connected graph. *Netw. 19*, 151–172.

FRANK, A. 1976. Combinatorial algorithms, algorithmic proofs. Ph.D. thesis, Eötvös University, Budapest.

FRANK, A. 1992. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discret. Math. 5,* 1, 25–53.

FRANK, A. 1999a. Finding minimum generators of path systems. *J. Comb. Theory Ser. B 75,* 2, 237–244.

FRANK, A. 1999b. Finding minimum weighted generators of a path system. In *Contemporary Trends in Discrete Mathematics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 49*, 129–138.

FRANK, A. 2003. An algorithm to increase the node-connectivity of a digraph. In *Proceedings of the 3rd Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*. Tokyo, Japan, 378–387.

FRANK, A. AND JORDÁN, T. 1995. Minimal edge-coverings of pairs of sets. *J. Comb. Theory Ser. B 65,* 1, 73–110.

FRANK, A. AND JORDÁN, T. 1999. Directed vertex-connectivity augmentation. *Math. Prog. 84*, 537–553.

FRANZBLAU, D. S. AND KLEITMAN, D. J. 1986. An algorithm for covering polygons with rectangles. *Inform. Control 63,* 3, 164–189.

GABOW, H. N. 1994. Efficient splitting off algorithms for graphs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC'94)*. ACM Press, New York, NY, 696–705.

GOEMANS, M. X. AND WILLIAMSON, D. P. 1997. *The Primal-Dual Method for Approximation Algorithms and Its Application to Network Design Problems*. PWS Publishing Co., Boston, MA.

GYŐRI, E. 1984. A min-max theorem on intervals. *J. Comb. Theory Ser. B 37*, 1–9.

HOPCROFT, J. E. AND KARP, R. M. 1973. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Comp. 2*, 225–231.

JACKSON, B. AND JORDÁN, T. 2005. Independence free graphs and vertex-connectivity augmentation. *J. Comb. Theory Ser. B 94*, 31–77.

JORDÁN, T. 1993. Increasing the vertex-connectivity in directed graphs. In *Proceedings of the 1st Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 726, 236–247.

JORDÁN, T. 1995. On the optimal vertex-connectivity augmentation. *J. Comb. Theory Ser. B 63,* 1, 8–20.

KARGER, D. R. AND LEVINE, M. S. 1998. Finding maximum flows in undirected graphs seems easier than bipartite matching. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*. ACM Press, New York, NY, 69–78.

AHUJA, R. K. T. M. AND ORLIN, J. 1993. *Network Flows. Theory, Algorithms and Applications*. Prentice-Hall, New York.

RECEIVED MAY 2005; REVISED JUNE 2006; ACCEPTED JULY 2006