Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2005-06. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

Primal-dual approach for directed vertex connectivity augmentation and generalizations

László A. Végh and András A. Benczúr

May 2005

Primal-dual approach for directed vertex connectivity augmentation and generalizations

László A. Végh* and András A. Benczúr**

Abstract

In their seminal paper, Frank and Jordán show that a large class of optimization problems including certain directed graph augmentation ones fall into the class of *covering supermodular functions over pairs of sets*. They also give an algorithm for such problems, however that 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 the second author's previous algorithm for the 0–1 valued case.

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 by 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.

1 Introduction

Edge connectivity augmentation problems form a subclass of survivable network design [17] 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 [3, 4, 6, 8, 12, 16, 21, 22, and many others].

*Department of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117. e-mail: veghal@cs.elte.hu. The author is member of the Egerváry Research Group (EGRES) Supported by the Hungarian National Foundation for Scientific Research Grant, OTKA T037547 and by European MCRTN ADONET, Grant Number 504438.

**Computer and Automation Institute, Hungarian Academy of Sciences, Lágymányosi u. 11., Budapest, Hungary H-1111, and Department of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117 e-mail: benczur@sztaki.hu. The author is member of the Egerváry Research Group (EGRES) Supported by OTKA T042481 and T042706.

In this paper we give a combinatorial algorithm for directed vertex and edge connectivity augmentation problems introduced by Frank and Jordán [12] as *covering supermodular functions over pairs of sets*. They give an algorithm that uses the ellipsoid method. Previously, combinatorial algorithms existed only for special problems [9, 10] and for increasing connectivity by one [2, 11]. 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 [13].

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 may consider all cuts of G with less than k vertices as set pairs (X, Y) of the vertex set where X is the sink and Y is the source side of the cut (recall the graph is directed). For a directed cut with sides X and Y, let

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

denote the number of vertices "missing" for a k-connected graph; for all other pairs X, Y let $p(X, Y) = 0$. The graph becomes k-connected iff for all X and Y we add at least $p(X, Y)$ edges that lead from X to Y. The running time of our algorithm is $O(n^5 \cdot \min\{k^4, n^2\})$ for this problem where k is the connectivity target. If we consider the special case when the starting graph G is already $k - 1$ connected, then we can give an even better running time bound of $O(n^3 m)$.

The above demand function p satisfies the following crossing supermodular property: whenever $X \cap X' \neq \emptyset$, $Y \cap Y' \neq \emptyset$ and $p(X, Y) > 0$, $p(X', Y') > 0$,

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

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 [12]. 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 [19, 14, 5].

In 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 non-combinatorial algorithm [12] and certain combinatorial ones [9, 10, 11, 5] start with a reduction to chain covers as in Dilworth's theorem.

Similar to most results related to covering problems over set pairs [12, 9, 10, 11, 5] we find Dilworth's chain cover theorem in 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 [2]. The relation between supermodular functions over set pairs and Dilworth's chain covers is based on the following observation. It is easy to show that for each directed edge (x, y) there is a unique set pair (X, Y) with X minimum and Y maximum and another (X', Y') with Y' minimum and X' maximum with $x \in X, X'$ and $y \in Y, Y'$. All other (X'', Y'') satisfy this property iff $X \subseteq X'' \subseteq X'$ and $Y \supseteq Y'' \supseteq Y'$. Thus

if we define a partially order with the above "skew" containment relation, we get the problem of covering a partially ordered set by intervals, a direct generalization of Dilworth's problem.

Our algorithm is based on the unweighted one of [2] that directly generalizes a Dilworth algorithm. In the weighted case 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 pseudo-polynomial 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. Algorithms for Dilworth's theorem are based on a reduction to the bipartite matching problem [15]. When we unfold the reduction of Dilworth's theorem to bipartite matchings, we find that the classical alternating path matching algorithm translates into an algorithm that (i) maintains one element for each chain as a candidate dual optimum; (ii) terminates with optimum if no element occurs more than its weight and they are pairwise incomparable when multiple copies are ignored; finally (iii) otherwise uses these elements to guard exchanges in chain parts such that one of the chains eventually becomes unnecessary for the cover. Such a direct Dilworth algorithm is described by Frank [7]. We remark that the current best bipartite matching algorithm is given in [20] and for Dilworth's problem in [5].

In the bulk of this paper 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 with that considered by Frank and Jordán [12]. Thus our algorithm applies among others to the task of increasing directed vertex connectivity or directed S–T edge connectivity to a target value.

The rest of the paper 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 [12]. In Section 3 we first give an overview of the primal-dual procedure, then in separate subsections show the key Procedures PUSHDOWN and REDUCE and in separate subsections show their correctness. Finally in Section 4 we briefly elaborate on the running times for the augmentation problems.

2 Poset properties of the Frank–Jordán set pairs

Frank and Jordán [12] 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 [12]). *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 $(\mathcal{P}, \leq)$; let $u, v \in \mathcal{P}$ be called **dependent** if $\exists\, m, M$ with $m \leq u \leq M$ and $m \leq v \leq M$; otherwise they are **independent**. For all dependent u and $v \in \mathcal{P}$ two operations $\vee$ and $\wedge$ are uniquely defined as

$$\begin{aligned} s \vee t &= \min\{x : x \geq s, x \geq t\}; \\ s \wedge t &= \max\{x : x \leq s, x \leq t\}. \end{aligned} \tag{2}$$

We say that for a minimal element m and a maximal element M, the set $\{x : m \leq x \leq M\}$ is the **interval** $[m, M]$. Let $\mathcal{P}$ satisfy furthermore the **strong interval property**: for every interval $[m, M]$,

$$u \wedge v \in [m, M] \text{ implies } u \in [m, M] \textbf{ or } v \in [m, M], \tag{3}$$

and the same holds with $u \wedge v$ replaced by $u \vee v$.

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

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

We say that $\mathcal{I}$ **covers** the function p if for every x at least $p(x)$ intervals contain x. An element v is called **tight** if we have equality.

Lemma 2.3. *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 $g(x) + g(y) \geq g(x \vee y) + g(x \wedge y)$. The proof is complete by

$$\begin{aligned} g(x \vee y) + g(x \wedge y) \geq p(x \vee y) + p(x \wedge y) \geq \\ \geq p(x) + p(y) = g(x) + g(y) \geq g(x \vee y) + g(x \wedge y) \end{aligned} \tag{4}$$

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. □

Lemma 2.4. *If x and y are two dependent tight elements with $p(x) > 0$, $p(y) > 0$, and the interval $[m, M]$ contains x, then it contains at least one of $x \vee y$ and $x \wedge y$; in addition either $y \leq M$ or $m \leq y$.*

Proof. Recall that by the proof of Lemma 2.3 we have equality everywhere in (4); 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 above equality implies the claim. □

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

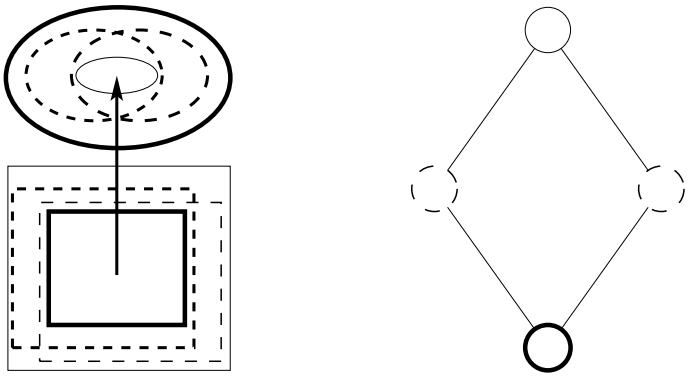

Figure 1: The correspondence between set pairs and poset elements. The four pairs on the right side can be covered by one edge, and the corresponding four elements are contained in one interval.

Theorem 2.5. *Let $\mathcal{P} \subseteq \{(X^-, X^+) : X^- \subseteq \mathcal{X}^-, X^+ \subseteq \mathcal{X}^+\}$ such that for all dependent $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$,*

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

For any $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$ let $x \leq y$ iff $X^- \subseteq Y^-$ and $X^+ \supseteq Y^+$. Then $\mathcal{P}$ with operations $\vee$, $\wedge$ and $\leq$ over $\mathcal{P}$ satisfies Definition 2.2. Furthermore subfamilies

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

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 (2) of Definition 2.2 follows directly by the properties of set union, intersection and containment.

To show the relation of intervals and subfamilies defined by pairs of vertices, consider vertices v_1 and v_2 first. Since all set pairs $\{(X^-, X^+) : v_1 \in X^-, v_2 \in X^+\}$

are dependent, we may take intersections and unions to find the unique minimal and maximal pairs. The interval consisting of all pairs (X^-, X^+) between the minimal and maximal satisfy $v_1 \in X^-, v_2 \in X^+$, hence v_1 and v_2 define a subset of an interval.

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 of both m and M since $v_1 \in Z^- \cap m^- \cap M^-$ and $v_2 \in Z^+ \cap m^+ \cap M^+$, thus $v \wedge m$ and $v \vee M$ exists. In the first case $v \wedge m < m$, in the second case $v \vee M > M$, both contradicting the extremity of m or M.

To show (3) of Definition 2.2, we take a pair v_1, v_2 defined as above for the interval $[m, M]$. It suffices to show this edge covers either $u = (X^-, X^-)$ or $v = (Y^-, Y^+)$. 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 $u \wedge v$ replaced by $u \vee v$, hence the claim follows. □

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

Theorem 2.6. *For a poset $\mathcal{P}$ as in Definition 2.2 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 2.5 implies that Theorem 2.1 is a special case of this theorem. Now we show that Theorem 2.1 implies Theorem 2.6, hence they are equivalent. Given a poset $\mathcal{P}$ as in Definition 2.2, let us define for all $x \in \mathcal{P}$ a representative element $\varphi(x)$. For $a \in \mathcal{P}$ let us define the pair $\delta(a) = (a^-, a^+)$ so that

$$a^- = \{\varphi(m) : m \leq a,\ m \in \mathcal{P} \text{ minimal}\} \qquad a^+ = \{\varphi(M) : M \geq a,\ M \in \mathcal{P} \text{ maximal}\}$$

It is easy to show that the function δ is a homomorphism for $\vee$, $\wedge$ and $\leq$, 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 2.6.

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 $I_1, \ldots, I_k$. 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. As long as the set of witnesses are non-independent 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 in 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 the algorithm calls Procedure REDUCE, a procedure that exchanges interval endpoints so that we get an interval cover of size one less.

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

In order to handle weighted posets, technically we need to consider multisets of intervals and witnesses in our algorithm. We assume $I_1, \ldots, I_k$ 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. *If for every i, j u_i and u_j are either independent or $u_i = u_j$, then the elements $\{u_1, ..., u_k\}$ give a dual optimal solution.*

Proof. It suffices to show that if for some poset element y there exists an i with $y = u_i$, then there exist exactly $p(y)$ such intervals I_j with $u_j = y$. 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 (see box) tries to push witnesses down along their intervals in iterations $t = 1, 2, \ldots$ until they satisfy the requirements of Lemma 3.1; witnesses are superscripted by the iteration value (t). 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 regard to I_i) if u and v are dependent and $v \not\leq M_i$. Different scenarios when u may push v down are shown in Fig. 2.

As the motivation of pushing $u_j^{(t)}$ down by $u_i^{(t)}$ we give the following claim as a relative easy consequence of Lemma 3.6; we omit the proof as it is not used elsewhere. After pushing $u_j^{(t)}$ down by $u_i^{(t)}$, for all subsequent $t' > t$ of the WHILE loop of Algorithm PUSHDOWN-REDUCE if the witnesses $u_j^{(t')}$ and $u_i^{(t')}$ for intervals i and j are dependent then they must be equal. Hence all non-equal dependent pairs of witnesses gradually disappear from the system.

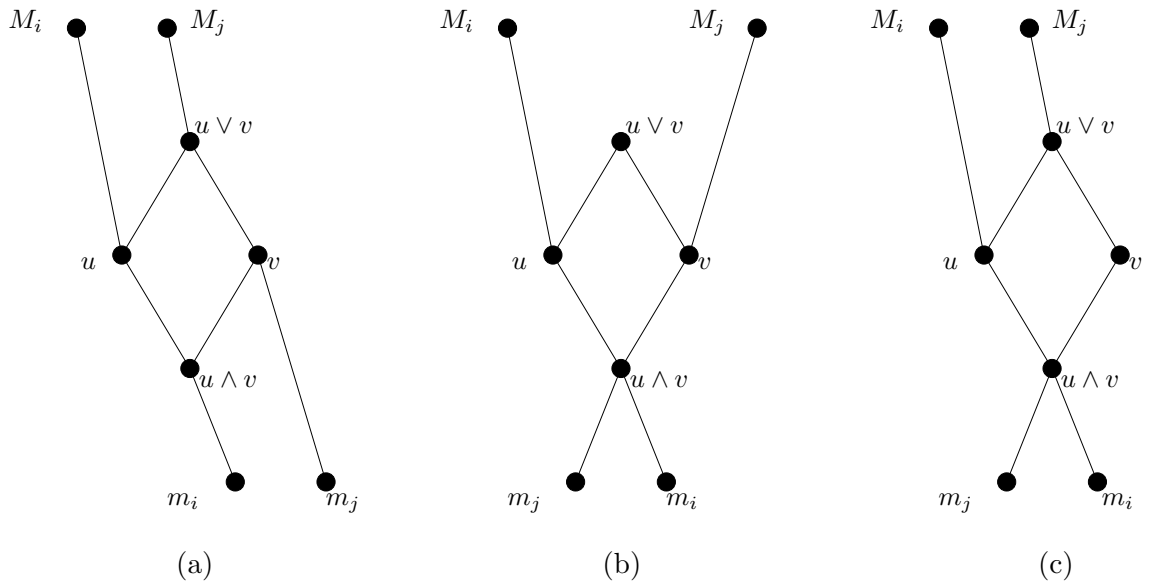

Figure 2: Different cases when u may push v down. By Lemma 2.4 $m_i \leq v$, and there are three possible cases: (a) $m_j \not\leq u \leq M_j$, (b) $m_j \leq u \not\leq M_j$, and (c) $m_j \leq u \leq M_j$

While the above motivation considers the dual solution, namely it shows that the set of witnesses becomes independent, we may also give a primal motivation of pushing v down by u. If u is maximum tight in I_i, then we may 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) one could prove that if u may push v down, then $u \leq M_j$ (as in [2] 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 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 [2]; 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 that one may push 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 2.4 we have that $m_i \leq v$. 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. Let v and v' be 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 2.4 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 2.4 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. □

Procedure PUSHDOWN$(j, t, \mathcal{I})$
  $V \leftarrow \{x : m_j \leq x \leq u_j^{(t)}$, 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 $x \in V$

The actual change of a witness $u_j^{(t)}$ is performed in Procedure PUSHDOWN (see box). 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 may push 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 may push 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 may 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. To see, 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 may be no pairs i and j such that $u_i^{(t)}$ may push $u_j^{(t)}$ down.

Lemma 3.6. *Assume that* $t_1 \leq t_2$*, 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* $u_i^{(t_2)} \leq u_j^{(t_1)}$.

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* $u_i^{(t+1)} \leq u_j^{(t)}$.

In the proof of Lemma 3.6 we need to characterize elements that cause witness u_j move below a certain tight element y. Assume that for some tight $y \in I_j$ and t we have $y \not\leq u_j^{(t)}$. Since $u_j^{(1)}$ is maximal tight, we may select the unique t_0 with $y \leq u_j^{(t_0)}$ but $y \not\leq u_j^{(t_0+1)}$. In step PUSHDOWN$(j, t_0, \mathcal{I})$ we must have an $u_d^{(t_0)}$ that may push 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* $z \not\leq u_j^{(t)}$ *and* $z \leq M_j$*. 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 the above observations for $y = u_j^{(t)} \vee z \in I_j$. Since y is tight, $y \leq u_j^{(1)}$. And since $z \not\leq u_j^{(t)}$, we get $y = u_j^{(t)} \vee z \not\leq u_j^{(t)}$. We select t_0 with $y \leq u_j^{(t_0)}$ but $y \not\leq u_j^{(t_0+1)}$; 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 $t - 1 \geq t_0$. 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. $u_i^{(t_2)} \leq M_j$, since $u_j^{(t_1)}$ may not push $u_i^{(t_2)}$ down. If $u_i^{(t_2)} \not\leq u_j^{(t_1)}$, 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$(i, t-1, \mathcal{I})$. □

3.3 The Reduce step

So far we have proved that if the initial primal solution is optimal, then the algorithm finds a dual optimum proof of this fact. Now we turn to the second scenario when

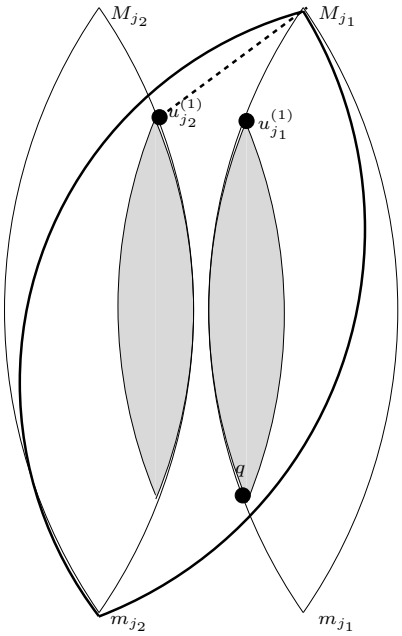

Figure 3: Procedure REDUCE called with $t^* = 1$. The two upright intervals are the original ones with their tight elements shaded. These two intervals will be replaced by the single bold interval. The new interval contains all tight elements of the old ones since $u^{(1)}_{j_2} \le M_{j_1}$ by Lemma 3.9.

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

---

one witness eventually 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.

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 steps shown in Fig. 3. Since $t^* = 1$, we have some $j = j_1 \le k$ such that Procedure REDUCE is called within Procedure PUSHDOWN$(j, 1, \mathcal{I})$. This means that

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

is empty. Let q be minimum tight in $[m_{j_1}, M_{j_1}]$; since $q \notin V$, we must have some $\ell = j_2$ such that $u^{(1)}_\ell$ may push q down. The algorithm selects such a j_2 and returns a reduced interval system

$$\mathcal{I} - [m_{j_1}, M_{j_1}] - [m_{j_2}, M_{j_2}] + [m_{j_2}, M_{j_1}]. \tag{5}$$

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

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

Lemma 3.9. *Let q be the minimal tight element of I_h. If $u \in I_\ell$ may push q down, then $u \le M_h$.*

Proof. Suppose by contradiction that $u \not\le M_h$. Since u and q are dependent, by Lemma 2.4 $u \wedge q \in I_h$. Since q is the minimal tight in I_h, we have $q \le u \wedge q$, hence $q \le u \le M_\ell$, contradicting that u may push q down. □

Lemma 3.10. *If $t^* = 1$, Procedure* REDUCE*$(j, t^*, \mathcal{I})$ 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 $[m_{j_1}, M_{j_1}] \cap [m_{j_2}, M_{j_2}]$. In this case we may replace the intervals $[m_{j_1}, M_{j_1}]$ and $[m_{j_2}, M_{j_2}]$ by $[m_{j_2}, M_{j_1}]$

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

Next let $x \in [m_{j_1}, M_{j_1}]$ be tight. $q \le x$ for the minimal tight q of $[m_{j_1}, M_{j_1}]$, and since $u_{j_1}^{(1)}$ may push q down, by Lemma 2.4 $m_{j_2} \le q$. Thus we get $m_{j_2} \le q \le x \le M_{j_1}$, 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 $q \le x \le M_{j_2}$, contradicting the fact that $u_{j_2}^{(1)}$ may push q down. □

Our aim in Procedure REDUCE (see box) 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 the minimum tight element of $[m_{j_{s+1}}, M_{j_s}]$ increases. We ensure this by defining

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

where q is the minimum tight element of $[m_{j_s}, M_{j_s}]$, and $t = t^* + 1 - s$. Applying Lemma 3.9 for $h = j_s$, $\ell = j_{s+1}$, $u = u_{j_{s+1}}^{(t)}$ we get $u_{j_{s+1}}^{(t)} \le M_{j_s}$. Thus when replacing $[m_{j_s}, M_{j_s}]$ by $[m_{j_{s+1}}, M_{j_s}]$, the tight elements in $[m_{j_{s+1}}, M_{j_{s+1}}]$ with $x \le u_{j_{s+1}}^{(t)}$ will no longer be tight. The overall idea is seen in Fig. 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

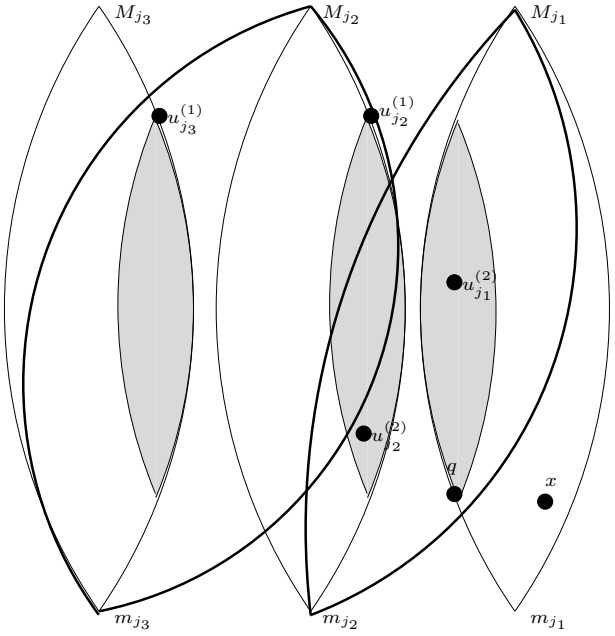

Figure 4: Procedure REDUCE called with $t^* = 2$. The three upright intervals are the original ones with their tight elements shaded. The original three intervals will be replaced by the two bold intervals using the marked witnesses. Note that the two new intervals contain all tight elements of the old ones. While the number of intervals covering certain non-tight elements (x in the example) may decrease, we prove that they remain covered.

$u_\ell^{(t^*)}$, the existence of such an ℓ is by no means obvious for all the other iterations of the main loop.

The existence of all further j_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 identical 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 backward from the sink, changing the flow along this edge to a preflow, and at each step proving that the remaining path augments the flow.

Let

$$\mathcal{I}' = \mathcal{I} - [m_{j_1}, M_{j_1}] + [m_{j_2}, M_{j_1}]. \tag{6}$$

be the set of intervals after the first iteration of Procedure REDUCE.

Theorem 3.11. *For $t^* > 1$ with input $\mathcal{I}'$ Algorithm* PUSHDOWN-REDUCE *performs the exact same steps as with input $\mathcal{I}$ until iteration t^*-1 when* REDUCE*$(j_2, t^*-1, \mathcal{I}')$ is called.*

By using the above theorem inductively for $t^*, t^*-1, \ldots, 1$ we prove that REDUCE finds an interval cover of size one less than before. This completes the correctness analysis of Procedure REDUCE.

To prove Theorem 3.11 now we define elements that are no longer tight and elements that become tight in the new cover:

Lemma 3.12. *Let*

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

Then

$$Z_1 \subseteq \{x : x \in [m_{j_2}, M_{j_1}], x \not\geq m_{j_1}\} \tag{7}$$
$$Z_2 \subseteq \{x : x \in [m_{j_1}, M_{j_1}], x \not\geq m_{j_2}\}. \tag{8}$$

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 identical for $\mathcal{I}$ and $\mathcal{I}'$ for $t < t^*$. The proof is based on the fact that the key elements used in defining $u_i^{(t)}$ do not belong to $Z_1 \cup Z_2$.

Lemma 3.13. *Let $u_i^{\prime(t)}$ denote elements selected by Algorithm* PUSHDOWN-REDUCE *with input $\mathcal{I}'$ with the convention that $u_{j_1}^{\prime(t)}$ belongs to the modified interval $[m_{j_2}, M_{j_1}]$. Then for all $t < t^*$ we have $u_i^{(t)} = u_i^{\prime(t)}$.*

Proof. By induction on $t \leq t^* - 1$ we will show $u_i^{\prime(t)} = u_i^{(t)}$. We show the inductive hypothesis in three steps: we show for $i = 1, \ldots, k$ that

1. $u_i^{\prime(t)}$ exists;
2. $u_i^{(t)} \notin Z_1$; and
3. $u_i^{\prime(t)} \notin Z_2$

The above three statements imply $u_i^{\prime(t)} = u_i^{(t)}$ as follows. For $t = 1$ the maximal tight elements are identical by 2 and 3 since $u_i^{\prime(1)}$ tight in $\mathcal{I}$ implies $u_i^{\prime(1)} \leq u_i^{(1)}$ and we have the opposite inequality when exchanging the role of the two elements. For general t by induction on the step of defining $u_i^{\prime(t)}$, one can observe that element $u_i^{(t)}$ belongs to the set V of Procedure REDUCE and the same holds when exchanging the role of $u_i^{\prime(t)}$ and $u_i^{(t)}$. Thus the two elements must be equal.

Assume first that $u_i^{(t)} \in Z_1$. Then by Lemma 3.12 $m_{j_2} \leq u_i^{(t)} \leq M_{j_1}$ and $m_{j_1} \not\leq u_i^{(t)}$. Since $m_{j_2} \leq u_{j_1}^{(t+1)} \leq M_{j_1}$ 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.

Next we show that $u_i^{\prime(t)}$ exists and $m_i \leq u_i^{(t)} \leq u_i^{\prime(t)}$. We proved above 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 $u_h^{(t-1)} = u_h^{\prime(t-1)}$ for all h. This yields $u_i^{(t)} \in V$ for PUSHDOWN$(i, t-1, \mathcal{I}')$ that in turn implies that $u_i^{\prime(t)}$ exists and $u_i^{(t)} \leq u_i^{\prime(t)}$.

Now we turn to the last part of the inductive hypothesis. Assume now that $u_i^{\prime(t)} \in Z_2$. By Lemma 3.12 $i \neq j_1$ and $m_{j_1} \leq u_i^{\prime(t)} \leq M_{j_1}$. Hence by applying Lemma 2.4 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 I: $u_{j_1}^{(t+1)} \leq M_i$. By Lemma 3.12 we also get $m_{j_2} \not\leq u_i^{\prime(t)}$, which in turn implies $u_{j_1}^{(t+1)} \not\leq u_i^{\prime(t)}$. Observe furthermore that $u_{j_1}^{(t+1)} \notin Z_1$, thus also tight in $\mathcal{I}'$. Combining with $u_j^{(t+1)} \leq M_i$ we may apply Lemma 3.8 for $\mathcal{I}'$, $u_i^{\prime(t)}$ and $z = u_{j_1}^{(t+1)}$. By the lemma there exists $t_0 < t$ such that an element $u_d^{\prime(t_0)}$ may push $u_{j_1}^{(t+1)}$ down. By induction $u_d^{(t_0)} = u_d^{\prime(t_0)}$, and by Lemma 3.2 $u_d^{(t)}$ may also push $u_{j_1}^{(t+1)}$ down.

Case II: $m_i \leq u_{j_1}^{(t+1)}$ and $u_{j_1}^{(t+1)} \not\leq M_i$. As we have seen above, $m_i \leq u_i^{(t)} \leq u_i^{\prime(t)}$. 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 3.11 by the following lemma.

Lemma 3.14. *When run with input $\mathcal{I}'$, Procedure* REDUCE *is called in iteration t^*-1 with $j = j_2$.*

Proof. By Lemma 3.13 Procedure REDUCE cannot be called before iteration $t^* - 1$. Suppose by contradiction that $u'^{(t^*)}_{j_2}$ exists. Since $u'^{(t^*)}_{j_2} \geq m_{j_2}$, by Lemma 3.12 $u'^{(t^*)}_{j_2} \notin Z_2$, hence $u'^{(t^*)}_{j_2} \leq u^{(t^*)}_{j_2} \leq M_{j_1}$ as in Lemma 3.13. We claim that $u'^{(t^*)}_{j_2} \in Z_1$, contradicting the fact that $u'^{(t^*)}_{j_2}$ is tight in $\mathcal{I}'$. For this we need to show $m_{j_1} \not\leq u'^{(t^*)}_{j_2}$.

Assume $m_{j_1} \leq u'^{(t^*)}_{j_2}$. This implies $m_{j_1} \leq u^{(t^*)}_{j_2} \leq M_{j_1}$, thus $q \leq u^{(t^*)}_{j_2}$ as q is the minimal tight element of $[m_{j_1}, M_{j_1}]$. In this case $u^{(t^*)}_{j_2}$ may not push q down, contradicting the selection of j_2 in Procedure REDUCE. □

4 Running times

In the application of our general algorithm to connectivity augmentation problems we have to be careful since 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.

The steps of our algorithm can be summarized as follows. Procedure PUSHDOWN takes $\vee$ and $\wedge$ of poset elements; compares poset elements (follows from taking $\vee$ and $\wedge$, provided we are able to tell the identity of poset elements); and computes a sequence of maximal tight elements. While the first two steps are trivial, we may, in connectivity augmentation applications, implement the third one as a sequence of BFS 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 (see box) that selects a strictly descending sequence of tight elements $y_0 > y_1 > \ldots > y_\ell$ with $y_0 = u^{(t)}_j$ and $y_\ell = u^{(t+1)}_j$ 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 $u^{(t)}_i$ for a single value of i.

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 $y_h \leq u^{(t+1)}_j$. This shows that if Procedure PUSHDOWN terminates by calling Procedure REDUCE, then so does Procedure ALTERNATE PUSHDOWN.

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**
    $V_h \leftarrow \{x : m_j \leq x \leq y_h, x \text{ tight and } u_i^{(t)} \text{ may not push } x \text{ down}\}$
    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

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 $u_i^{(t)}$ may push $y_h \vee u_j^{(t+1)}$ down. 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 still holds:

Lemma 4.2. *The set of intervals $\mathcal{J}_{j,i}$ covers all the elements of I_j; furthermore in $\mathcal{J}_{j,i}$ an element $x \in I_j$ with $x \leq y_h$ is tight if and only if $x \in V_h$.*

Proof. For all $x \in I_j$ we clearly have $x \in [m_i, M_j]$ if $x \in I_i$, hence the number of intervals covering x cannot decrease in $\mathcal{J}_{j,i}$. If $x \leq y_h$ is tight and $x \notin V_h$, then $m_i \leq x \not\leq M_i$, thus the number of intervals containing x will in fact increase by adding $[m_i, M_j]$ and x becomes no longer tight. On the other hand, if $x \in V_h$, then x is either contained in both intervals $[m_i, M_i]$ and $[m_j, M_j]$ or in neither of them. □

By the lemma our basic step consists of computing the maximum tight element of an interval for certain set of covering intervals. To see, notice that the lemma implies that y_{h+1} is the intersection of y_h and the maximum tight element Q of I_j in $\mathcal{J}_{j,i}$. Furthermore at the beginning of the algorithm $u_j^{(1)}$ is the maximum tight element of I_j. Next we show the running time of a basic step for the case of directed vertex connectivity augmentation.

We implement the above basic step of Procedure ALTERNATE-PUSHDOWN for graph augmentation problems, we use the reduction of vertex connectivity augmentation to poset covering as in Theorem 2.5. For each interval $I = [m_i, M_i] \in \mathcal{I}$ we augment the graph by an edge with tail in a vertex corresponding to m_i and head in a vertex

corresponding to M_i. If $\mathcal{J}$ covers all poset elements in $[m, M]$, then the minimum s–t cut in the augmented graph has value at least k.

As the initialization of the algorithm we compute $|\mathcal{I}|$ maximum flows, one corresponding to each interval in $\mathcal{I}$. For interval $[m_j, M_j]$ we compute a maximum s–t flow for s corresponding to m_j and t corresponding to M_j. Since $\mathcal{I}$ is a cover, the maximum flow value is at least k. If the s–t flow value is more than k, then $[m_j, M_j]$ contains no tight elements and we remove from the cover. Otherwise $u_j^{(1)}$ is the set pair corresponding to the maximal value k cut that can be obtained by a breadth-first search from t.

Lemma 4.3. *Consider the task of finding the maximum tight element of an interval $I_j = [m_j, M_j]$ for certain set of intervals $\mathcal{J}_{j,i}$ that cover I_j (the* basic step*). Using the maximum flow computed at the initialization, this step requires $O(1)$ breadth-first search (BFS) computations.*

Proof. We consider the maximum flow computed for $[m_j, M_j]$ at the initialization. We add the edge corresponding to $[m_i, M_j]$ to the graph, and remove the one corresponding to $[m_i, M_i]$. 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 the sink we either obtain the maximum tight element or deduce that there are no tight elements and Procedure Reduce can be called. □

The number of basic steps is polynomial in the number of initial intervals j and the length of a longest chain ℓ in the poset: we take $j\ell$ basic steps for one Reduce while the latter may happen $O(j)$ times. For vertex connectivity problems $\ell = O(n)$, giving $O(j \cdot n)$ basic steps for a single Reduce step.

The number of Reduce steps will be bounded using approximate augmentation results. When we increase connectivity by only one, the approximate augmentation result of Jordán [21] gives an initial solution containing at most $O(k)$ more intervals than the optimal solution where k is the connectivity of the input graph. This implies at most $O(k)$ Reduce calls. By Corollary 4.7 of [12], in this case there is an optimal solution of cardinality at most n. These two results imply that j can be bounded by $n + k = O(n)$. Thus we need $O(n^3)$ basic steps and $O(n)$ maxflow computations. A basic step consists of $O(1)$ BFS on a graph of $O(n + m + j) = O(m)$ edges, thus the running time for this case is $O(n^3 m)$.

For the general case the non-polynomial algorithm of [13] can easily be turned to a polynomial one that finds a solution with $O(k^4)$ more edges than the optimum. This gives a running time $O(k^4 \cdot n^3(m + j))$. A trivial bound on j is n^2 since by adding a complete graph the input graph becomes $(n-1)$-vertex-connected, giving a running time of $O(n^5 \cdot \min\{k^4, n^2\})$.

Conclusion

We have given a combinatorial algorithm for covering posets satisfying a special property by the minimal number of intervals of the poset. As noticed by Frank and Jordán

[12], the result can be applied for certain directed edge augmentation problems. The existence of a strongly polynomial combinatorial algorithm, however, remains open. Another major open problem regards the complexity of undirected augmentation; here only approximate algorithms are known [22].

One may wonder of how strong the generalizational power of the interval covering problem. 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 [23], 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 [1] 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 [20].

References

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network Flows*, Prentice Hall (1993).

[2] András A. Benczúr, Pushdown-Reduce: An algorithm for connectivity augmentation and poset covering problems. *Discr. Appl. Math*, pp 233-262 vol 129 (2003)

[3] András A. Benczúr, Parallel and fast sequential algorithms for undirected edge connectivity augmentation, *Math. Prog. B* **84**(3):595–640 (1999) and *Proc. 26th Annual ACM Symp. on Theory of Comp.*, pp. 658–667 (1994)

[4] András A. Benczúr and David R. Karger, Augmenting undirected edge-connectivity in $\tilde{O}(n^2)$ time, *J. Alg.* **37**(1), pp. 2–36 (2000) and *Proc. 9th ACM-SIAM Symp. on Discrete Algorithms*, pp. 500–509. (1998)

[5] Benczúr, A.A., J. Förster and Z. Király, Dilworth's Theorem and its application for path systems of a cycle—implementation and analysis. *Proc. European Symp. Alg., Springer Lecture Notes in Computer Science* 1643:598–509 (1999)

[6] Cai, G–P. and Y–G. Sun, The minimum augmentation of any graph to a k-edge-connected graph, *Networks* **19** (1989), pp. 151–172.

[7] Frank, A., *Combinatorial algorithms, algorithmic proofs.* Doctoral Thesis, Budapest, Eötvös University (1976), in Hungarian

[8] Frank, A., Augmenting graphs to meet edge connectivity requirements, *SIAM J. Discr. Math.* **5**(1), pp. 25–53 (1992), and *Proc. 31st Annual IEEE Symp. on Foundations of Comp. Sci.* (1990)

[9] Frank, A., Finding minimum generators of path systems, *JCT B* **75** (1999), pp. 237–244.

[10] Frank, A., Finding minimum weighted generators of a path system, Contemporary Trends in Discrete Mathemaics (eds.: R.L. Graham, J. Kratochvil, J. Nesetril, and F.S. Roberts), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Volume **49** (1999), pp. 129–138.

[11] Frank, A., Finding minimum edge-coverings of pairs of subsets, EGRES Technical Report Series (2001). Available at `http://www.cs.elte.hu/egres/`

[12] Frank, A. and T. Jordán, Minimal edge-coverings of pairs of sets, *JCT* B **65** (1995), pp. 73–110.

[13] Frank, A. and Jordán, T., Directed vertex-connectivity augmentation *Math. Prog. 84*, (1999), pp. 537-553

[14] Franzblau, D.S. and D.J. Kleitman, An algorithm for constructing polygons with rectangles, *Information and Control* **63** (1984), pp. 164–189.

[15] Ford, L.R. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press (1962).

[16] Gabow, H.N., Efficient Splitting Off Algorithms for Graphs, *Proc. 26th Annual ACM Symp. on Theory of Comp.* (1994), pp. 696–705.

[17] Goemans, M.X. and D.P. Williamson, The primal-dual method for approximation algorithms and its application to network design problems, *Approximation Algorithms for NP-hard Problems* (Ed. D.S Hochbaum), PWS Publishing Co., Boston MA (1997).

[18] Grötschel, M., L. Lovász and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1**:169–197 (1981)

[19] Győri, E., A min-max theorem on intervals, *JCT* B **37** (1984), pp. 1–9.

[20] Hopcroft, J.E. and R.M. Karp, An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, *SIAM J. Comp.* **2** (1973), pp. 225–231.

[21] Jordán, T., Increasing the vertex-connectivity in directed graphs, In: Proceedings of the First Annual European Symposium on Algorithms, Bad Honnef, 1993, Springer Lecture Notes In Computer Science, Vol 726, pp. 236-247, Springer-Verlag, New York/Berlin, 1993

[22] Jordán, T., On the optimal vertex connectivity augmentation, *J. Combin. Theory Ser. B* **63** (1995), pp. 8–20.

[23] Karger, D.R. and M.S. Levine, Finding Maximum Flows in Simple Undirected Graphs is Easier than Bipartite Matching, *30th ACM Symposium on Theory of Computing* (1998).

[24] Knuth, D.E., Irredundant intervals, *ACM Journal of Experimental Algorithmics* **1** (1996)