Listing minimal edge-covers of intersecting families with applications to connectivity problems

Zeev Nutov

The Open University of Israel

nutov@openu.ac.il

Abstract

Let $G = (V, E)$ be a directed/undirected graph, let $s, t \in V$, and let $\mathcal{F}$ be an intersecting family on V (that is, $X \cap Y, X \cup Y \in \mathcal{F}$ for any intersecting $X, Y \in \mathcal{F}$) so that $s \in X$ and $t \notin X$ for every $X \in \mathcal{F}$. An edge set $I \subseteq E$ is an edge-cover of $\mathcal{F}$ if for every $X \in \mathcal{F}$ there is an edge in I from X to $V - X$. We show that minimal edge-covers of $\mathcal{F}$ can be listed with polynomial delay, provided that for any $I \subseteq E$ the minimal member of the residual family $\mathcal{F}_I$ of the sets in $\mathcal{F}$ not covered by I can be computed in polynomial time. As an application, we show that minimal undirected Steiner networks, and minimal k-connected and k-outconnected spanning subgraphs of a given directed/undirected graph, can be listed in incremental polynomial time.

1 Introduction

1.1 The problem, motivation, and previous work

We consider listing problems of the following type: given a graph G and a prescribed graph property Π, list the family $\Pi(G)$ of all subgraphs H of G that satisfy Π. For example, the property Π can be "H is a spanning tree", "H is an st-path", "H is a cycle", and so on, c.f., [10]. In listing problems, the output may be exponential in the input size, and thus the time complexity is usually expressed in both the input and the output sizes (see e.g., [12, 9, 6]). A listing algorithm runs in *incremental polynomial time* if it lists any $N \leq |\Pi(G)|$ members of $\Pi(G)$ in time polynomial in N and the input size; if every additional member of $\Pi(G)$ is listed in time polynomial in the size of G, then the algorithm has *polynomial delay*.

Graph connectivity is a fundamental concept in network reliability theory. While in the simplest case only the connectedness is required, in many applications higher levels of connectivity are desirable. Some methods computing network reliability depend on the efficient listing of all minimal sub-networks with the required connectivity [12, 4]. In a very general setting, the type of problems we consider can be defined as follows. The *S-connectivity* $\lambda_G^S(u,v)$ of (u,v) in G is the maximum number of uv-paths such that no two of them have an edge or a node in $S-\{u,v\}$ in common.

Steiner Networks Listing (SNL)
Input: A (possibly directed) graph $G=(V,E)$, $S\subseteq V$, and requirements $r(u,v)$ on $V\times V$.
Output: A list of minimal spanning subgraphs H of G that satisfy

$$\lambda_H^S(u,v)\geq r(u,v)\quad \forall (u,v)\in V\times V\ . \tag{1}$$

Common particular choices of S are: $S=\emptyset$ (edge-connectivity), $S=V$ (node-connectivity), and any S so that $r(u,v)=0$ whenever $u\in S$ or $v\in S$ (so called element-connectivity). In the undirected setting, the requirements are symmetric, namely, $r(u,v)=r(v,u)$ for all $u,v\in V$. For brevity, let $\lambda_H(u,v)=\lambda_H^{\emptyset}(u,v)$ and $\kappa_H(u,v)=\lambda_H^V(u,v)$.

In [3] it was shown that for $\{0,1\}$-requirements undirected SNL can be solved in incremental polynomial time, while for directed graphs no such algorithm exists, unless P=NP. However, for listing minimal directed Steiner trees, [3] gave an algorithm with polynomial delay. We consider the undirected edge-connectivity SNL, and directed/undirected edge/node-connectivity variants of the following two problems. A graph is *k-outconnected from s* if $\kappa(s,v)\geq k$ for all $v\in V-s$, namely, if it contains k internally-disjoint sv-paths for every $v\in V$. A graph is *k-connected* if it is k-outconnected from every $s\in V$.

k-Outconnected Subgraphs Listing (k-OSL)
Input: A (possibly directed) graph $G=(V,E)$, $s\in V$, and an integer k.
Output: A list of minimal k-outconnected from s spanning subgraphs of G.

k-Connected Subgraphs Listing (k-CSL)
Input: A (possibly directed) graph $G=(V,E)$ and an integer k.
Output: A list of minimal k-connected spanning subgraphs of G.

We also consider edge-connectivity listing problems. These include as a special case the problems of listing: directed/undirected spanning trees [10, 11, 5], minimal strongly connected subgraphs [2], simple st-paths [10], and others. In [1] is given an incremental polynomial time algorithm for undirected k-Edge-Connected Subgraphs Listing, and for k-CSL for any *fixed* k. In [1] was posed an open problem if k-CSL admits an incremental polynomial time algorithm when k is not fixed, but is a part of the input.

Throughout the paper, let $G = (V, E)$ denote the input graph. Let $n = |V|$ and $m = |E|$. Given G and graph property Π, our goal is to list the family $\Pi_{\min}$ of minimal members of the family $\Pi(G)$ of all subgraphs of G that satisfy Π. We assume that $G \in \Pi(G)$ and that $\Pi(G) \neq \emptyset$; otherwise our algorithms can be easily modified to return an error message.

1.2 Results in this paper

In this paper we characterize SNL instances that can be solved in incremental polynomial time; recall that in [3] it was shown that directed SNL with $\{0, 1\}$-requirements (in this case all the choices of S are equivalent) does not admit such algorithm, unless P=NP. We prove:

Theorem 1.1 *Undirected edge-connectivity* SNL*, directed/undirected edge/node-connectivity* k-CSL*, and directed edge/node-connectivity* k-OSL*, admit an incremental polynomial time algorithm.*

Theorem 1.1 is just a summary of (some) applications of an algorithm that lists minimal edge-covers of an intersecting st-family. A family $\mathcal{F}$ of subsets of V is an *intersecting family* if $X \cap Y, X \cup Y \in \mathcal{F}$ for any intersecting $X, Y \in \mathcal{F}$. $\mathcal{F}$ is an *st-family* if $s \in X$ and $t \in V - X$ for every $X \in \mathcal{F}$. An *edge set I covers $\mathcal{F}$* if for every $X \in \mathcal{F}$ there is $uv \in I$ with $u \in X$ and $v \in V - X$. We give an algorithm for listing minimal (possibly directed) edge-covers of an intersecting st-family $\mathcal{F}$, but its efficient implementation, in case $\mathcal{F}$ is not given explicitly, requires that certain queries related to $\mathcal{F}$ can be answered in polynomial time. Given an edge set I on V, the *residual family $\mathcal{F}_I$ of $\mathcal{F}$* (w.r.t. I) consists of all members of $\mathcal{F}$ that are uncovered by edges of I. It is well known that if $\mathcal{F}$ is intersecting, or if $\mathcal{F}$ is an st-family, so is $\mathcal{F}_I$, for any I. An inclusion minimal member of $\mathcal{F}$ is called an *$\mathcal{F}$-core*. Clearly, the $\mathcal{F}$-cores of an intersecting family $\mathcal{F}$ are pairwise disjoint, and an intersecting st-family has a unique core. For any edge set I on V, make the following assumption:

The Core Assumption:
Computing the $\mathcal{F}_I$-core C of an intersecting st-family $\mathcal{F}$, or determining that I is an $\mathcal{F}$-cover, can be implemented in $Q(m, n)$ time, where $Q(m, n)$ is polynomial in m and n.

Theorem 1.2 *Directed/undirected minimal edge-covers of an intersecting st-family can be listed with delay $O(n(Q(m, n) + m))$, under the Core Assumption.*

Let $G = (V, E)$ be a graph. For disjoint $X, Y \subseteq V$ let $\delta_G(X, Y) = \delta_E(X, Y)$ be the set of edges from X to Y in E; let $\delta_E(X) = \delta_E(X, V - X)$ and let $d_E(X) = |\delta_E(X)|$ be the *degree* of X in G.

Example: A natural example of an intersecting st-family is the family $\mathcal{F} = \{X \subseteq V - t : s \in X, d_H(X) = \lambda_H(s,t)\}$ of minimum st-cuts in a (possibly directed) graph $H = (V, E)$. It is well known that this $\mathcal{F}$ is intersecting. The $\mathcal{F}$-core C can be found using one max-flow computation; after a maximum flow is computed, C is the set of nodes reachable from s in the corresponding "residual" network.

We consider the following generalizations of Theorem 1.2. Let $p : 2^V \to Z_+$ be a *set-function* on V. An edge set J on V is a *p-cover*, if $d_J(X) \geq p(X)$ for every $X \subseteq V$. Connectivity listing problems can be formulated as a **Set-Function Covers Listing** (SFCL) problem: given a directed/undirected graph $G = (V, E)$ and a set-function p on V, list all minimal p-covers contained in E. In the undirected setting, we assume that p is *symmetric*, namely, that $p(X) = p(V - X)$ for all $X \subseteq V$. The following two types of set-functions often arise in various connectivity problems (c.f., [8]).

Definition 1.1 *A set function p on V is* skew-supermodular *if for every $X, Y \subseteq V$ at least one of the following holds:*

$$p(X) + p(Y) \leq p(X \cap Y) + p(X \cup Y) \tag{2}$$

$$p(X) + p(Y) \leq p(X - Y) + p(Y - X) \tag{3}$$

If (2) always holds whenever $X \cap Y \neq \emptyset$ and $X \cup Y \neq V$ then p is crossing-supermodular.

Definition 1.2 *Let p be a set function on V and let J be an edge set on V. The* residual function p_J of p *(w.r.t. J) is*

$$p_J(X) = \max\{p(X) - d_J(X), 0\} \quad \forall X \subseteq V \ .$$

Let $\mathcal{F}(p) = \{X \in \mathcal{F} : p(X) > 0\}$ denote the support family of p. *For $s \in V$ let $\mathcal{F}^s(p) = \{X \in \mathcal{F}(p) : s \in X\}$.*

Note that for any p-cover J and $e = st \in J$, p_{J-e} is a $0,1$ function and $\mathcal{F}^s(p_{J-e})$ is an st-family.

Proposition 1.3 *If p is symmetric skew-supermodular, or if p is crossing-supermodular, then $\mathcal{F} = \mathcal{F}^s(p_{J-e})$ is an intersecting st-family for any $e = st \in J \in \Pi_{\min}$.*

Proof: If p is symmetric skew-supermodular, or if p is crossing-supermodular, so is p_J for any edge set J, c.f., [8]. Thus if p is skew supermodular then $X \cup Y, X \cap Y \in \mathcal{F}$ or $X - Y, Y - X \in \mathcal{F}$ for any $X, Y \in \mathcal{F}$; if p is crossing-supermodular then $X \cap Y, X \cup Y \in \mathcal{F}$ for any $X, Y \in \mathcal{F}$. In both cases, since $\mathcal{F}$ is an st-family, it is an intersecting st-family. □

We say that the *Core Assumption* holds for p if $\mathcal{F} = \mathcal{F}^s(p_{J-e})$ satisfies the Core Assumption for any p-cover J and $e = st \in J$. Using Theorem 1.2 we prove:

Theorem 1.4 SFCL *admits an incremental polynomial time algorithm for: undirected G and symmetric skew-supermodular p, and for directed G and crossing-supermodular p, under the Core Assumption.*

Theorems 1.2, 1.4, and 1.1 are proved in Sections 2, 3, and 4, respectively.

2 Proof of Theorem 1.2

Let $G = (V, E)$ be a graph and let $\mathcal{F}$ be a non-empty intersecting st-family on V. Clearly, $\mathcal{F}$ has a unique core. A naive approach to generate an $\mathcal{F}$-cover I is as follows: start with $I = \emptyset$, and while I is not an $\mathcal{F}$-cover, repeatedly add an edge $e \in \delta(C_I)$ to I, where C_I is the minimal $\mathcal{F}_I$-core. Note that then, at every step, the edges in I have both endpoints in C_I. This indeed generates an $\mathcal{F}$-cover, but it might not be minimal, e.g., the last edge added might cover the whole $\mathcal{F}$, and thus the edges added at previous steps are redundant. We may fix this by removing redundant edges from the obtained cover, but this makes difficult to force the algorithm to generate every cover exactly once. To ensure minimality, we maintain the following property of a partial cover I: for every $e \in I$ there exists $C \in \mathcal{F}$ so that $\delta_I(C) = \{e\}$. To achieve that, when an edge $e \in \delta(C_I)$ is added to I, we delete all the other edges in $\delta(C_I)$, after verifying that $E - \delta_E(C_I)$ covers $\mathcal{F}_{I+e}$. This guarantees that the $\mathcal{F}$-cover produced is minimal, and we use recursion to produce each $\mathcal{F}$-cover exactly once. Formally, the algorithm is:

Initialization: $E' \leftarrow E$, $I \leftarrow \emptyset$.
LIST(E', I)
Let C be the $\mathcal{F}_I$-core.
For every $e \in \delta_{E'}(C)$ do:
  If e is an $\mathcal{F}_I$-cover then return $I + e$;
  Else If $E' - \delta_{E'}(C)$ is an $\mathcal{F}_{I+e}$-cover then do:
    LIST$(E' - \delta_{E'}(C), I + e)$.
  EndIf
  EndIf
EndFor

The following (known) statement is used to prove that the algorithm lists every minimal $\mathcal{F}$-cover exactly once.

Lemma 2.1 *Let I be a minimal (directed or undirected) cover of an intersecting st-family $\mathcal{F}$ and let C be the $\mathcal{F}$-core. Then $d_I(C) = 1$.*

Proof: Suppose to the contrary that there are $e \neq f \in \delta_I(C)$. By the minimality of I, there exists $W_e, W_f \in \mathcal{F}$ such that $\delta_I(W_e) = \{e\}$ and $\delta_I(W_f) = \{f\}$. Note that $C \subseteq W_e \cap W_f$ and thus each of e, f covers $W_e \cap W_f$. There is an edge in I that covers $W_e \cup W_f$, since $W_e, W_f \in \mathcal{F}$ implies $W_e \cup W_f \in \mathcal{F}$. This edge must be one of e, f, since if for arbitrary intersecting sets X, Y an edge covers $X \cup Y$ then it also covers one of X, Y. Consequently, one of e, f covers both $W_e \cap W_f$ and $W_e \cup W_f$, and thus covers each of W_e, W_f (if for arbitrary intersecting sets X, Y an edge covers both $X \cap Y, X \cup Y$ then it also covers each of X, Y). This contradicts that $\delta_I(W_e) = \{e\}$ and $\delta_I(W_f) = \{f\}$. □

Corollary 2.2 *Every minimal $\mathcal{F}$-cover of an intersecting st-family $\mathcal{F}$ is listed by* LIST *exactly once.*

Proof: By induction on $|\mathcal{F}|$. For $|\mathcal{F}| = 1$ the statement is obvious. Assuming that the statement is true for any family $\mathcal{F}'$ with $1 \leq |\mathcal{F}'| < |\mathcal{F}|$, we will prove it for $\mathcal{F}$. Let C be the minimal $\mathcal{F}$-core. By Lemma 2.1, any minimal $\mathcal{F}$-cover I contains a unique edge in $\delta_E(C)$. We claim that when an edge $e \in \delta_{E'}(C)$ is considered in the main loop, the algorithm lists exactly once every minimal $\mathcal{F}$-cover in E containing e. As we examine every edge in $\delta_E(C)$ exactly once, the statement follows. Indeed, if e covers $\mathcal{F}$, then $\{e\}$ is the unique minimal $\mathcal{F}$-cover containing e. If $E - \delta_E(C) + e$ is not an $\mathcal{F}$-cover, then no $\mathcal{F}$-cover containing e exists. Else, by the induction hypothesis, the algorithm lists exactly once every minimal cover of $\mathcal{F}' = \mathcal{F}_e$, and adds e to each $\mathcal{F}_e$-cover listed. □

Lemma 2.3 LIST *can be implemented to run with delay $O(n(Q(m,n)+m))$.*

Proof: Let I be a partial $\mathcal{F}$-cover generated in the run of the algorithm, and let C be an $\mathcal{F}_I$-core. Recall that C can be found in $Q(m,n)$ time. Using fundamental data structures, we can find an edge e (in fact, all the edges) in $\delta_{E'}(C)$ in $O(m)$ time. Checking if e is an $\mathcal{F}_I$-cover, or if $E' - \delta_{E'}(C)$ is an $\mathcal{F}_{I+e}$-cover can be done in $Q(m,n)$ time. Hence the time invested in the pair C, e is $O(Q(m,n)+m)$. After e is chosen (added to the partial cover), at most $n - |C| = O(n)$ core examinations occur until a new minimal $\mathcal{F}$-cover is discovered. Hence the delay is $O(n(Q(m,n)+m))$. □

The proof of Theorem 1.2 is complete.

3 Proof of Theorem 1.4

To prove Theorem 1.4, we use a reduction from [3, 7] implied by the backtracking method for enumeration [10]. Given a monotone family Π (namely $I \in \Pi$ and $I \subseteq I'$ implies $I' \in \Pi$)

of subsets of E, let $\Pi_{\min}$ be the family of inclusion minimal members of Π. Recall that our goal is to list the family $\Pi_{\min}$.

Proposition 3.1 ([3, 7]) *Let Π be a monotone family of subsets of E so that that the membership in Π can be tested in polynomial time. If for any pair e, J with $e \in J \in \Pi_{\min}$ we can list the minimal members of $\Pi(J,e) = \{Y \subseteq E - J : J - e + Y \in \Pi\}$ in incremental polynomial time, then $\Pi_{\min}$ can also be listed in incremental polynomial time.*

Let $G = (V, E)$ be a directed/undirected graph, let p be a (symmetric, if G is undirected) set function on V, and let Π be the family of p-covers in E. Clearly, the family of p-covers is monotone, and under the Core Assumption, checking whether J is a (minimal) p-cover can be done in polynomial time. Thus Proposition 3.1 together with Theorem 1.2 determines the following reduction.

Corollary 3.2 *If $\mathcal{F}^s(p_{J-e})$ is an intersecting st-family for any $e = st, J$ with $e \in J \in \Pi_{\min}$, then $\Pi_{\min}$ can be listed in incremental polynomial time, under the Core Assumption.*

Proof: Since J is an $\mathcal{F}$-cover, p_{J-e} is a $0,1$ valued function. It is easy to see that in the directed setting, $\mathcal{F}(p_{J-e}) = \mathcal{F}^s(p_{J-e})$, namely, that $p_{J-e}(X) = 1$ implies $s \in X$ and $t \in V - X$. Thus in the directed setting, an edge set is a p_{J-e}-cover if, and only if, it is an $\mathcal{F}^s(p_{J-e})$-cover. This is also so in the undirected setting, since p, and thus also p_{J-e}, is symmetric. The statement now easily follows from Proposition 3.1 and Theorem 1.2. □

Theorem 1.4 now follows from Proposition 1.3.

4 Proof of Theorem 1.1

Edge-connectivity SNL can be formulated as SFCL as follows (c.f., [8]). By Menger's Theorem, $H = (V, J)$ satisfies (1) (with $S = \emptyset$) if, and only if,

$$d_J(X) \geq p(X) \equiv \max\{r(u,v) : u \in X, v \in V - X\} \quad \forall\, \emptyset \neq X \subset V\ . \tag{4}$$

We set $p(\emptyset) = p(V) = 0$. (*Remark:* For general SNL instances a more general model is required, where p is defined on pairs of subsets of V, see [8].) The set-function p defined in (4) is skew-supermodular, c.f., [8]. Furthermore, p is symmetric, if r is symmetric and J is undirected. Thus undirected edge-connectivity SNL is equivalent to the problem of listing minimal edge covers of the skew-supermodular symmetric set-function p defined by (4).

Lemma 4.1 *Let p defined by (4). Then $\mathcal{F} = \mathcal{F}^s(p_{J-e})$ is an intersecting st-family for any (undirected) p-cover J and $e = st \in J$; furthermore, $\mathcal{F}$ satisfies the Core Assumption.*

Proof: $\mathcal{F}$ is an intersecting st-family by Proposition 1.3. We show that $\mathcal{F}$ satisfies the Core Assumption. Given an edge set I, the minimal $\mathcal{F}_I$-core can be computed in polynomial time as follows. In the graph $H = (V, J - e + I)$, for every $\{u, v\} \subseteq V$, compute a maximum uv-flow. If its value is $r(u, v)-1$, find the minimal set C_{uv} so that $s \in C_{uv}$, $|C_{uv} \cap \{u, v\}| = 1$, and $d_H(C_{uv}) = r(u, v) - 1$. If no set C_{uv} exists, then I is an $\mathcal{F}$-cover. Otherwise, the minimum size set among the sets C_{uv} computed is the $\mathcal{F}$-core. □

The SNL part of Theorem 1.1 now follows from Theorem 1.4.

It remains to prove the k-OSL/k-CSL part. Undirected edge-connectivity k-CSL is just a particular case of SNL. For edge-connectivity, directed k-OSL (in fact, the reverse problem of k-OSL, when we require k disjoint paths from every $v \in V$ to s) and k-CSL are particular cases of directed SFCL with crossing supermodular p. However, we do not see such an immediate reduction for directed node-connectivity k-OSL/k-CSL, nor for the undirected node-connectivity k-CSL. We thus will present a different formal proof, which we will also be able to extend to all the variants of k-OSL/k-CSL considered in Theorem 1.1.

We show that Proposition 3.1 reduces both directed/undirected k-CSL and the directed k-OSL (but *not* the undirected k-OSL) to the directed/undirected variants of following problem which we think is of independent interest:

Minimal k-Paths Augmentations Listing (k-PAL):
Input: An integer k, a graph $H = (V, J)$ with $(k-1)$ edge/internally-disjoint st-paths where $s, t \in V$, and an edge set E on V disjoint to J.
Output: A list of minimal augmenting edge-sets $I \subseteq E$ so that $H + I$ has k edge/internally-disjoint st-paths.

Lemma 4.2 *If directed/undirected edge/node-connectivity k-PAL admits an incremental polynomial time algorithm, then so are directed/undirected edge/node-connectivity k-CSL and directed edge/node-connectivity k-OSL.*

Proof: This follows from Proposition 3.1 and the following two known facts. Let $H = (V, J)$ be a graph, let $e \in J$, and let I be an edge set on V.

Fact 1: If H is directed/undirected k-edge/node-connected and $e = st \in J$, then $H - e + I$ is k-connected if, and only if, $H - e + I$ contains k edge/internally-disjoint st-paths.

Fact 2: If H directed and k-outconnected from s, and $e = ut \in J$, then $H - e + I$ is k-outconnected from s if, and only if, $H - e + I$ contains k edge/internally-disjoint st-paths.

□

Remark: Fact 2 in the proof of Lemma 4.2 does not extend to *undirected* node-connectivity k-OSL. E.g., let $V = \{s, x, y, u, t\}$, $J = \{sx, sy, su, st, xy, ut\}$, $e = ut$, and $I = xt$. The reason is that some of the "deficient sets" created by the deletion of ut contain u, while the others contain t (so we should require both k internally disjoint st- and su-paths). We believe that undirected k-OSL also admits an incremental polynomial time algorithm, but resolving this question is beyond the scope of this paper.

To complete the proof of Theorem 1.1, in the rest of this section we prove:

Theorem 4.3 *Directed/undirected edge/node-connectivity k-PAL admits an algorithm with polynomial delay.*

The proof of Theorem 4.3 folows. The directed/undirected version of k-PAL when the paths are required to be edge-disjoint is easily reduced to the the problem of listing minimal directed/undirected covers of an intersecting st-family. Specifically, it is well known that the set family $\mathcal{F} = \{X \subset V - t : s \in X, d_H(X) = k - 1\}$ is intersecting if H has $k - 1$ edge-disjoint st-paths, and that (by Menger's Theorem) $H + I$ has k edge-disjoint st-paths if, and only if, I is an $\mathcal{F}$-cover. The other details are also straightforward.

To handle *directed* k-PAL with internally-disjoint paths, we apply a standard reduction to the edge-disjoint variant of k-PAL. Specifically, one can view the graph $H = (V, J + E)$ as a network with source s and sink t where the nodes in $V - s$ and all edges have unit capacity. Apply a standard conversion of node capacities to edge capacities: replace every node $v \in V - s$ by the two nodes v^+, v^- connected by the edge v^+v^- having the same capacity as v, and redirect the heads of the edges entering v to v^+ and the tails of the edges leaving v to v^-.

A natural approach to solve the *undirected* (edge/node-connectivity) k-PAL is to reduce it to the directed k-PAL with H and E replaced by their bidirections $D(H)$ and $D(E)$, that are obtained from H and E, respectively, by replacing every undirected edge by two opposite directed edges. Then we list all directed augmenting edge sets for the obtained directed k-PAL instance, and output their underlying undirected edge sets. It is not hard to see that this will produce all augmenting edge sets for the original undirected k-PAL instance. However, since in general, an undirected edge set has many orientations, it is not obvious that such an algorithm will list every undirected augmenting edge set exactly once. Hence, to prove that this approach works, we need the following statement:

Lemma 4.4 *Let $H = (V, J)$ be an undirected graph containing $k - 1$ internally disjoint st-paths, let I be a minimal edge set on V so that $H + I$ contains k internally disjoint st-paths, and let D be the bidirection of H. Then there exists a unique orientation I_D of I so that*

$D + I_D$ contains k internally disjoint st-paths.

The proof of Lemma 4.4 follows. We need several definitions and preliminary statements.

Definition 4.1 *An ordered pair (S, T) of disjoint subsets of V is called a* setpair*; (S, T) is an st-*setpair *if $s \in S$ and $t \in T$. An edge set I covers a setpair family $\mathcal{F}$ if $d_I(S, T) \geq 1$ for all $(S, T) \in \mathcal{F}$. A family $\mathcal{F}$ of setpairs is* intersecting *if $(S' \cap S'', T' \cup T''), (S' \cup S'', T' \cap T'') \in \mathcal{F}$ for any $(S', T'), (S'', T'') \in \mathcal{F}$ with $S' \cap S'', T' \cap T'' \neq \emptyset$; $(S, T) \in \mathcal{F}$ is an $\mathcal{F}$-*core *if $S \subseteq S'$ and $T \supseteq T'$ for any $(S', T') \in \mathcal{F}$.*

Other definitions, e.g., the residual family $\mathcal{F}_I$ of a setpair family $\mathcal{F}$, are also natural analogues of the ones used for set families. The proof of the following "setpair analogue" of Lemma 2.1 is identical to that of Lemma 2.1, and thus is omitted.

Lemma 4.5 *Let I be a minimal (directed or undirected) cover of an intersecting setpair st-family $\mathcal{F}$ and let (S, T) be the $\mathcal{F}$-core. Then $d_I(S, T) = 1$.*

Corollary 4.6 *Let I be a minimal directed/undirected cover of an intersecting setpair st-family $\mathcal{F}$. Then there exist a unique ordering $e_1, \ldots, e_q$ of I, and a family $(S_1, T_1), \ldots (S_q, T_q)$ of setpairs in $\mathcal{F}$, so that:*
(i) *$S_1 \subseteq S_2 \cdots \subseteq S_q$ and $S_1 \supseteq S_2 \cdots \supseteq S_q$;*
(ii) *(S_j, T_j) is the $\mathcal{F}_{I_{j-1}}$-core where $I_0 = \emptyset$ and $I_{j-1} = \{e_1, \ldots, e_{j-1}\}$ for $j = 2, \ldots, q$;*
(iii) *$\delta_I(S_j, T_j) = \{e_j\}$, $j = 1, \ldots, q$.*
Thus if I is undirected, then I has a unique orientation that covers $\mathcal{F}$, namely, every e_j is oriented from S_j to T_j.

Proof: The required orderings are uniquely determined as follows. (S_1, T_1) is the $\mathcal{F}$-core. By Lemma 4.5, there is a unique edge in $\delta_I(S_1, T_1)$, say e_1. (S_2, T_2) is the $\mathcal{F}_{e_1}$-core and $\delta_I(S_2, T_2) = \{e_2\}$. And so on, namely, (S_j, T_j) is the $\mathcal{F}_{I_{j-1}}$-core and $\delta_I(S_j, T_j) = \{e_j\}$. □

Let us now get back to the proof of Lemma 4.4.

Proof of Lemma 4.4 We may assume that $st \notin J$; otherwise, the same proof applies on $H - st$ with k replaced by $k - 1$. Let us say that an st-setpair is *tight* in H (in D), if $|S \cup T| = k - 1$ and $d_H(S, T) = 0$. Since D is a bidirection of H, (S, T) is tight in D if, and only if, it is tight in H. Let $\mathcal{F}$ be the family of tight setpairs in H (in D). By Menger's Theorem, $H + I$ (or $D + I$) contains k internally disjoint st-paths if, and only if, I covers $\mathcal{F}$. It is also known that $\mathcal{F}$ is an intersecting setpair family. Thus, by Corollary 4.6, I has a unique orientation I_D that covers $\mathcal{F}$. The statement follows. □

The proof of Theorem 4.3, and thus also of Theorem 1.1 is complete.

Remark: Our algorithm for listing minimal directed/undirected edge-covers of an intersecting st-family easily extends from set-families to setpair-families, by replacing the set-core C by the setpair-core (S, T). This results in the following statement, that also provides an alternative proof of Theorem 4.3:
Directed/undirected minimal edge-covers of an intersecting st-family can be listed with delay $O(n(Q(m,n)+m))$, under the Core Assumption.
The proof of this statement is identical to that of Theorem 1.2, except that we use Lemma 4.5 instead of Lemma 2.1.

5 Conclusions and open problems

In this paper we characterized several minimal connectivity structures that can be listed in incremental polynomial time. In particular, we gave incremental polynomial time listing algorithms for: undirected edge-connectivity SNL, directed/undirected k-CSL, and directed k-OSL. We note that the undirected element-connectivity SNL admits a similar result, by a similar proof.

One open problem is which among the problems we considered admits a listing algorithm with polynomial delay. Another question is whether there are additional interesting SNL/SFCL instances, especially those that correspond to directed graphs, that admit an efficient listing algorithm; e.g., the result of [3] that directed Steiner trees can be listed with polynomial delay, *cannot* be deduced from any statement in this paper. Finally, a natural question is whether the results in this paper can be extended to *setpair* families, that correspond to node-connectivity requirements. In particular, is the *undirected* k-OSL admits an efficient listing algorithm?

Acknowledgment: I thank an anonymous referee for many useful comments.

References

[1] E. Boros, K. Borys, K. Elbassioni, V. Gurvich, K. Makino, and G. Rudolf. Generating k-vertex connected spanning subgraphs and k-edge connected spanning subgraphs. Manuscript, 2007.

[2] E. Boros, K. Elbassioni, V. Gurvich, and L. Khachiyan. Enumerating minimal dicuts and strongly connected subgraphs and related geometric problems. In *Integer Program-*

ming and Combinatorial Optimization (IPCO), volume LNCS 3153, pages 152–162, 2004.

[3] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. Generating paths and cuts in multi-pole(di)graphs. In *MFCS*, volume 3153, pages 298–309, 2004.

[4] C. J. Coulbourn. *The Combinatorics of Network Reliability.* Oxford University Press, 1987.

[5] H. N. Gabow and E. W. Mayers. Finding all spanning trees of directed and undirected graphs. *SIAM Journal on Computing*, 7(3):280–287, 1978.

[6] D. S. Johnson and H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, 1988.

[7] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, V. Gurvich, and K. Makino. Generating cut conjunctions and bridge avoiding extensions in graphs. In *Algorithms and Computation: 16th International Symposium, ISAAC 2005*, pages 156–165, 2005.

[8] G. Kortsarz and Z. Nutov. Approximating minimum cost connectivity problems, in *Approximation Algorithms and Metaheuristics,* T. F. Gonzalez ed.,. CRC, 2005.

[9] E. Lawler, J. K. Lenstra, and A. H. G. R. Kan. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9:558–565, 1980.

[10] R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. *Networks*, 5:237–252, 1975.

[11] A. Shioura, A. Tamura, and T. Uno. An optimal algorithm for scanning all spanning trees of undirected graphs. *SIAM Journal on Computing*, 26(3):678–692, 1997.

[12] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.