

# Capacity-Preserving Subgraphs of Directed Flow Networks<sup>\*</sup>

Markus Chimani<sup>[0000-0002-4681-5550]</sup> and Max Ilsen<sup>✉[0000-0002-4532-3829]</sup>

Theoretical Computer Science, Osnabrück University, Osnabrück, Germany  
{markus.chimani,max.ilsen}@uos.de

**Abstract.** We introduce and discuss the MINIMUM CAPACITY-PRESERVING SUBGRAPH (MCPS) problem: given a directed graph and a retention ratio  $\alpha \in (0, 1)$ , find the smallest subgraph that, for each pair of vertices  $(u, v)$ , preserves at least a fraction  $\alpha$  of a maximum  $u$ - $v$ -flow's value. This problem originates from the practical setting of reducing the power consumption in a computer network: it models turning off as many links as possible while retaining the ability to transmit at least  $\alpha$  times the traffic compared to the original network.

First we prove that MCPS is NP-hard already on directed acyclic graphs (DAGs). Our reduction also shows that a closely related problem (which only considers the arguably most complicated core of the problem in the objective function) is NP-hard to approximate within a sublogarithmic factor already on DAGs. In terms of positive results, we present a simple linear time algorithm that solves MCPS optimally on directed series-parallel graphs (DSPs). Further, we introduce the family of laminar series-parallel graphs (LSPs), a generalization of DSPs that also includes cyclic and very dense graphs. Not only are we able to solve MCPS on LSPs in quadratic time, but our approach also yields straightforward quadratic time algorithms for several related problems such as MINIMUM EQUIVALENT DIGRAPH and DIRECTED HAMILTONIAN CYCLE on LSPs.

**Keywords:** Maximum flow · Minimum equivalent digraph · Series-parallel graphs · Inapproximability

## 1 Introduction

We present the MINIMUM CAPACITY-PRESERVING SUBGRAPH (MCPS) problem. Interestingly, despite it being very natural, simple to formulate, and practically relevant, there seems to have been virtually no explicit research regarding it. We may motivate the problem by recent developments in Internet usage and routing research: Not only does Internet traffic grow rapidly [18,10,34], current Internet usage shows distinct traffic peaks in the evening (when people, e.g., are streaming videos) and lows at night and in the early morning [30,19]. This has sparked research into the reduction of power consumption in backbone Internet providers (Tier 1) by turning off unused resources [35,9]: One natural way is to

<sup>\*</sup> Supported by the German Research Foundation (DFG) grant CH 897/7-1.

turn off as many connections between servers as possible, while still retaining the ability to route the occurring traffic. Typically, one assumes that (a) the original routing network is suitably dimensioned and structured for the traffic demands at peak times, and (b) the traffic demands in the low times are mostly similar to the peak demands but “scaled down” by some ratio.

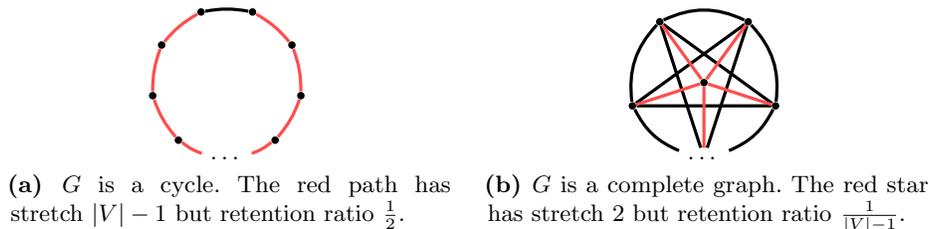
Graph-theoretically, we are given a directed graph  $G = (V, E)$ , a capacity function  $cap: E \rightarrow \mathbb{R}^+$  on its edges, and a retention ratio  $\alpha \in (0, 1)$ . All graphs are simple (i.e., contain no self-loops nor parallel edges). For every pair of vertices  $(s, t) \in V^2$ , let  $c_G(s, t)$  denote the value of a maximum flow (or equivalently, a minimum cut) from  $s$  to  $t$  in  $G$  according to the capacity function  $cap$ . Thus, in the following we unambiguously refer to  $c_G(s, t)$  as the *capacity* of the vertex pair  $(s, t)$  in  $G$ , which intuitively represents how much flow can be sent from  $s$  to  $t$  in  $G$ . The lowest capacity among all vertex pairs corresponds to the size  $\mathfrak{c}(G)$  of the global minimum cut. One may ask for an edge-wise minimum subgraph  $G' = (V, E')$ ,  $E' \subseteq E$ , such that  $\mathfrak{c}(G') \geq \alpha \cdot \mathfrak{c}(G)$ . We call this problem **MINIMUM GLOBAL CAPACITY-PRESERVING SUBGRAPH (MGCPS)**:

**Observation 1** *MGCPS is NP-hard, both on directed and undirected graphs, already with unit edge capacities.*

*Proof.* Identifying a Hamiltonian cycle in a directed strongly-connected (or undirected 2-edge-connected) graph  $G$  is NP-hard [22]. Consider an optimal MGCPS solution for  $G$  with unit edge capacities and  $\alpha = 1/\mathfrak{c}(G)$  ( $2/\mathfrak{c}(G)$ ): every vertex pair is precisely required to have a capacity of at least  $\lceil \alpha \cdot \mathfrak{c}(G) \rceil = 1$  ( $\lceil \alpha \cdot \mathfrak{c}(G) \rceil = 2$ ). Hence, an  $\alpha$ -MGCPS of  $G$  must also be strongly-connected (2-edge-connected, respectively) and is a Hamiltonian cycle if and only if one exists in  $G$ .  $\square$

However, in our practical scenario, MGCPS is not so interesting. Thus, we rather consider the problem where the capacities  $c_G(u, v)$  have to be retained for each vertex pair  $(u, v)$  individually: In the **MINIMUM CAPACITY-PRESERVING SUBGRAPH (MCPS)** problem, we are given a directed graph  $G = (V, E)$  including edge capacities  $cap$  and a retention ratio  $\alpha \in (0, 1)$ . We ask for a set of edges  $E' \subseteq E$  with minimum size  $|E'|$  yielding the subgraph  $G' = (V, E')$ , such that  $c_{G'}(s, t) \geq \alpha \cdot c_G(s, t)$  for all  $(s, t) \in V^2$ . For an MCPS instance  $(G, \alpha)$ , we will call a vertex pair  $(s, t)$  (or edge  $st$ ) *covered* by an edge set  $E'$  if the graph  $G' = (V, E')$  satisfies  $c_{G'}(s, t) \geq \alpha \cdot c_G(s, t)$ . In the following, we discuss the special setting where the capacity function  $cap$  assigns 1 to every edge—in this setting,  $c_G(s, t)$  equals the maximum number of edge-disjoint  $s$ - $t$ -paths in  $G$ .

**Related Work.** Capacity-preserving subgraphs are related to the research field of sparsification. There, given a graph  $G$ , one is typically interested in an upper bound on the size of a graph  $H$  that preserves some of  $G$ 's properties up to an error margin  $\varepsilon$ . Graph  $H$  may be a minor of  $G$ , a subgraph of  $G$ , or a completely new graph on a subset of  $G$ 's vertices (in which case it is called a vertex sparsifier [25]). Such research does not necessarily yield approximation algorithms w.r.t. minimum sparsifier size as the obtained upper bound may not



**Fig. 1.** Two examples of subgraphs (in red) whose stretch differs greatly from their retention ratio. All edge lengths and edge capacities are 1, making clear that using the reciprocals of the edge lengths as edge capacities does not lead to a direct relation between stretch and capacity either.

be easily correlated to the instance-specific smallest possible  $H$  (e.g., the general upper bound may be  $|E(H)| = \mathcal{O}(\frac{|V|\log|V|}{\epsilon^2})$  whereas there are instances where an optimal  $H$  is a path); however, sparsifiers often can be used as a black box in other approximation algorithms. A majority of cut/flow sparsification research only concerns undirected graphs: For example, Benczúr and Karger show how to create a subgraph that approximately preserves the value of every cut by sampling edges of the original graph [6,7]. Spielman and Teng present a more general result concerning spectral graph properties [31]. Furthermore, there exist techniques for finding vertex sparsifiers that preserve the congestion of any multi-commodity flow [5,17]. The main results for directed graphs concern sparsifiers that depend on cut balance [8], and spectral sparsifiers of strongly connected graphs [11,12] or general directed graphs [36] that, however, do not necessarily preserve cut values.

Closely related to sparsifiers are spanners (see, e.g., [1] for a survey on this rich field). These are subgraphs that preserve the length of a shortest path within a given ratio (stretch factor) between each pair of vertices. However, even the most basic results in this line of work cannot be applied to MCPS due to fundamental differences between shortest paths and minimum cuts (Figure 1 illustrates this point). Results by Räcke [27], later generalized in [4] and used for flow sparsification in [17], show (on undirected graphs) a direct correspondence between the existence of probabilistic mappings with stretch at most  $\varrho \geq 1$  and those with *congestion* at most  $\varrho$ . However, the notion of congestion differs greatly from capacity, where the former is defined as the maximum ratio between the flow routed over an edge in a multi-commodity flow setting and its capacity.

When the capacity is equal to the number of edge-disjoint paths (i.e. for unit edge capacities), MCPS is a special case of the DIRECTED SURVIVABLE NETWORK DESIGN (DSND) problem, where one asks for the smallest subgraph of a directed graph that satisfies given edge-connectivity requirements for each vertex pair. Dahl [13,14] studied DSND from a polyhedral point of view and presented an ILP approach that can easily be adapted to solve MCPS. But algorithmically, DSND has not received as much attention as its undirected counterpart [23] (for which a 2-approximation algorithm exists [21]).

Lastly, MCPS can be seen as a generalization of the well-established MINIMUM EQUIVALENT DIGRAPH (MED) problem [24,33,2]: Given a directed graph  $G = (V, E)$ , one asks for a cardinality-wise minimum edge set  $E' \subseteq E$  such that the subgraph  $G' = (V, E')$  preserves the reachability relation for every pair of vertices. We may think of the MED as a directed version of the MINIMUM SPANNING TREE (despite not being tree-like)—the latter contains an undirected path from each vertex to every other reachable vertex, the former contains a directed one. MED has been shown to be NP-hard via a reduction from DIRECTED HAMILTONIAN CYCLE [29,20,22]. The NP-hardness of MCPS follows from a simple observation:

**Observation 2** *MED is the special case of MCPS with  $\alpha = \min_{(s,t) \in V^2} 1/c_G(s,t)$ .*

There exist several polynomial approximation algorithms for MED, which are all based on the contraction of cycles [24,37]; the currently best ratio is 1.5 [33]. Moreover, MED can be solved optimally in linear time on graphs whose *shadow*—the underlying undirected graph obtained by ignoring edge orientations—is series-parallel [28], and in quadratic time on DAGs [2]. The latter algorithm simply deletes all those edges  $uv$  for which there exists another  $u$ - $v$ -path.

**Our Contribution.** In this paper, we introduce the natural problem MCPS, which we assume may be of wider interest to the algorithmic community.

Based on the fact that MED is simple on DAGs, one might expect to similarly find a polynomial algorithm for MCPS on DAGs as well. However, in Section 2 we show that the arguably most complex core of MCPS cannot even be approximated within a sublogarithmic factor, already on DAGs, unless  $P=NP$ .

In Section 3 we introduce the class of *laminar series-parallel graphs (LSPs)*—a generalization of directed series-parallel graphs (DSPs) that also allows, e.g., cycles and dense subgraphs. LSPs have the potential to allow simple algorithms and proofs for a wider array of problems, not just MCPS, due to their structural relation to DSPs. For example, the MCPS-analogue of a well-known spanner property [3] holds on LSPs but not on general graphs: if the retention constraint is satisfied for all edges, it is also satisfied for every vertex pair (see Theorem 11).

In Section 4, we complement the hardness result by a linear-time algorithm for MCPS on DSPs, and a quadratic one on LSPs. While the latter algorithm’s proof requires the one of the former, both algorithms themselves are independent and surprisingly simple. Lastly, we show a further use of LSPs: our algorithms can directly be applied to other related problems, and we prove that the algorithm for MED on DAGs described in [2] in fact also works on general LSPs (Section 4.3).

## 2 Inapproximability on DAGs

Since a capacity-preserving subgraph always contains an MED (which might be quite large), MCPS can be approximated on sparse graphs by simply returning an arbitrary feasible solution (i.e., a set of edges such that the corresponding subgraph satisfies the capacity requirement for every vertex pair).

**Observation 3** *Every feasible solution for a connected MCPS instance is an  $m/n-1$ -approximation.*

*Proof.* Every feasible MCPS solution must contain at least as many edges as an MED to ensure a capacity of 1 for all vertex pairs  $(u, v)$  where  $v$  is reachable from  $u$ . An MED of a connected graph is also connected. Thus, an optimal MCPS solution contains at least  $n - 1$  edges whereas a feasible one contains at most all  $m$  original edges.  $\square$

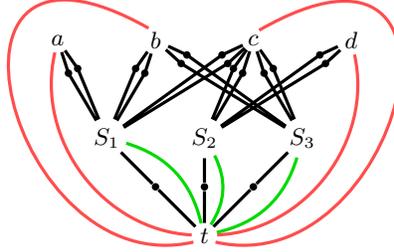
Hence, it seems sensible to consider a slightly altered version MCPS\* of the MCPS with a tweaked objective function  $|E'| - m_{\text{MED}}$ , which does not take into account the number of edges  $m_{\text{MED}}$  in an MED but aims at focusing on the problem's core complexity beyond the MED. We show that it is NP-hard to approximate MCPS\* on DAGs to within a sublogarithmic factor using a reduction from the decision variant of SET COVER (SC): given a universe  $U$  and a family of sets  $\mathcal{S} = \{S_i \subseteq U \mid i = 1, \dots, k\}$  with  $k \in \mathcal{O}(\text{poly}(|U|))$ , one asks for a subfamily  $\mathcal{C} \subseteq \mathcal{S}$  with minimum size  $|\mathcal{C}|$  such that  $\bigcup_{S \in \mathcal{C}} S = U$ . For an SC instance  $(U, \mathcal{S})$ , let  $f(u) := |\{S \in \mathcal{S} \mid S \ni u\}|$  denote  $u$ 's *frequency*, i.e., the number of sets that contain  $u$ , and  $f := \max_{u \in U} f(u)$  the maximum frequency.

**Theorem 4.** *Any polynomial algorithm can only guarantee an approximation ratio in  $\Omega(\log |E|)$  for MCPS\*, unless  $P=NP$ . This already holds on DAGs with maximum path length 4.*

*Proof.* We give a reduction from SC to MCPS\* on DAGs such that any feasible solution for the new MCPS\* instance can be transformed into a feasible solution for the original SC instance with an equal or lower objective function value in linear time. The size  $|E|$  of our MCPS\* instance is linear in the size  $N \in \mathcal{O}(|U| \cdot k) = \mathcal{O}(|U|^r)$  of the SC instance, i.e.,  $|E| = c \cdot |U|^r$  for some constants  $c, r$ : if it was possible to approximate MCPS\* on DAGs within a factor in  $o(\log |E|) = o(\log(c \cdot |U|^r)) = o(\log |U|)$ , one could also approximate SC within  $o(\log |U|)$ . However, it is NP-hard to approximate SC within a factor of  $\varepsilon \ln(|U|)$  for any positive  $\varepsilon < 1$  [15,26]. To create the MCPS\* instance  $(G, \alpha)$ , let  $\alpha := \frac{1}{2}$  and construct  $G$  as follows (see Figure 2 for a visualization of  $G$ ):

$$\begin{aligned} G &:= (V_U \cup V_S^U \cup V_S \cup V_t^S \cup \{t\}, E_U \cup E_S \cup E_G \cup E_{\mathcal{R}}) \\ V_U &:= \{v_u \mid \forall u \in U\} & V_S &:= \{v_S \mid \forall S \in \mathcal{S}\} \\ V_S^U &:= \{x_S^u, y_S^u \mid \forall S \in \mathcal{S}, u \in S\} & V_t^S &:= \{z_S \mid \forall S \in \mathcal{S}\} \\ E_U &:= \{v_u x_S^u, v_u y_S^u, x_S^u v_S, y_S^u v_S \mid \forall S \in \mathcal{S}, u \in S\} & E_S &:= \{v_S z_S, z_S t \mid s \in \mathcal{S}\} \\ E_G &:= V_S \times \{t\} & E_{\mathcal{R}} &:= V_U \times \{t\} \end{aligned}$$

As  $G$  is a DAG, its MED is unique [2]. This MED is formed by  $E_U \cup E_S$  and already covers all vertex pairs except  $V_U \times \{t\}$ . Let  $(v_u, t) \in V_U \times \{t\}$ : Its capacity in  $G$  is  $2f(u) + 1$  and this pair thus requires a capacity of  $\lceil \frac{1}{2} \cdot (2f(u) + 1) \rceil = f(u) + 1$  in the solution. Since the MED already has a  $v_u$ - $t$ -capacity of  $f(u)$ , only one additional edge is needed to satisfy the capacity requirement: either



**Fig. 2.** MCPS instance constructed from the SC instance  $(U, \mathcal{S})$  with  $U = \{a, b, c, d\}$ ,  $\mathcal{S} = \{\{a, b, c\}, \{c, d\}, \{b, c\}\}$ . An optimal solution contains the MED (drawn in black) as well as one corresponding red or green edge for each  $u \in U$ . Edges are directed from upper to lower vertices.

the *corresponding red edge*  $v_u t \in E_{\mathcal{R}}$ , or one of the *corresponding green edges*  $\{v_{st} \in E_{\mathcal{G}} \mid S \ni u\}$ . After choosing a corresponding red or green edge for each item  $u \in U$ , the number of these edges is the value of the resulting solution.

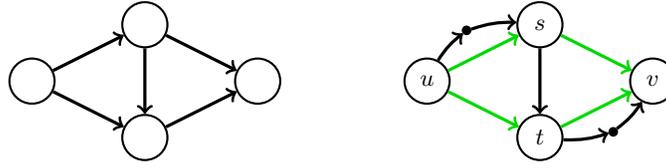
Given an SC solution  $\mathcal{C}$ , we can construct an MCPS\* solution  $E_U \cup E_{\mathcal{S}} \cup \{v_{st} \in E_{\mathcal{G}} \mid S \in \mathcal{C}\}$  with the same value. Since every item is covered by the sets in  $\mathcal{C}$ , the constructed MCPS\* solution includes at least one corresponding green edge for each item, ensuring its feasibility.

To turn a feasible solution  $E'$  for the MCPS\* instance into a feasible solution for the original SC instance with an equal or lower value, we remove all red edges  $v_u t \in E_{\mathcal{R}}$  from the MCPS\* solution and replace each of them—if necessary—by one green edge  $v_{st} \in E_{\mathcal{G}}$  with  $S \ni u$ . Since the MCPS\* solution has at least one corresponding green edge for each item  $u \in U$ , the resulting SC solution  $\{S \mid v_{st} \in E_{\mathcal{G}} \cap E'\}$  also contains at least one covering set for each item.  $\square$

The same reduction shows the NP-hardness of MCPS on DAGs: an optimal SC solution  $\{S \mid v_{st} \in E_{\mathcal{G}} \cap E'\}$  can be easily obtained from an optimal solution  $E'$  for the MCPS instance  $(G, \alpha)$ ,  $\alpha = \frac{1}{2}$ . Moreover, the largest capacity between any two vertices in  $G$  is  $2f + 1$ . Since SC is already NP-hard for  $f = 2$  (in the form of VERTEX COVER [20,22]), we arrive at the following corollary:

**Corollary 5.** MCPS is NP-hard already on DAGs  $G$  with maximum path length 4 and  $\max_{(u,v) \in V^2} c_G(u, v) = 5$ .

The above reduction for MCPS\* with  $\alpha = \frac{1}{2}$  can be generalized to MCPS\* for every  $\alpha = \frac{p}{p+1}$  with  $p \in \mathbb{N}_{>0}$ . This only requires a small change in the construction:  $E_U$  must contain  $p + 1$   $v_u$ - $v_S$ -paths of length 2 for all  $(v_u, v_S) \in V_U \times V_S$ , and  $E_{\mathcal{S}}$  must contain  $p$   $v_S$ - $t$ -paths of length 2 for all  $v_S \in V_S$ .



**Fig. 3.** (Left) The graph  $W$ , whose subdivisions cannot be contained in DSPs. (Right) Graph  $W$  with two added paths of length 2, see Observation 12. The  $u$ - $v$ -capacity is 3. For  $\alpha = \frac{1}{2}$ , all edges of the original graph are covered by the MED (black edges) but the vertex pair  $(u, v)$  is not. Observe that the graph is not a DSP but its shadow is  $s$ - $t$ -series-parallel.

### 3 Laminar Series-Parallel Graphs

In this section, we introduce laminar series-parallel graphs (LSPs)—a rich graph family that not only includes the directed series-parallel graphs (DSPs) but also cyclic graphs and graphs with multiple sources and sinks. A (directed) graph  $G$  is (*directed*)  $s$ - $t$ -series-parallel ( $s$ - $t$ -( $D$ )SP) if and only if it is a single edge  $st$  or there exist two (directed, resp.)  $s_i$ - $t_i$ -series-parallel graphs  $G_i$ ,  $i \in \{1, 2\}$ , such that  $G$  can be created from their disjoint union by one of the following operations [16]:

1. P-composition: Identify  $s_1$  with  $s_2$  and  $t_1$  with  $t_2$ . Then,  $s = s_1$  and  $t = t_1$ .
2. S-composition: Identify  $t_1$  with  $s_2$ . Then,  $s = s_1$  and  $t = t_2$ .

There also exists a widely known forbidden subgraph characterization of DSPs:

**Theorem 6** (see [32]). *A directed graph  $G = (V, E)$  is a DSP if and only if it is acyclic, has exactly one source, exactly one sink, and  $G$  does not contain a subgraph homeomorphic to  $W$  (displayed in Figure 3(left)), i.e., a subgraph that is a subdivision of  $W$ .*

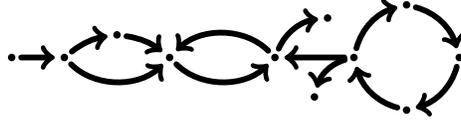
Given a directed graph  $G = (V, E)$ , for every vertex pair  $(u, v) \in V^2$ , let  $G(u, v)$  be the graph induced by the edges on  $u$ - $v$ -paths. Note that such a path-induced subgraph may contain cycles but a single path may not. If  $e = uv$  is an edge, we call  $G(u, v)$  an *edge-anchored subgraph (EAS)* and may use the shorthand notation  $G\langle e \rangle$ . Based on these notions, we can define LSPs:

**Definition 7 (Laminar Series-Parallel Graph).** *A directed graph  $G = (V, E)$  is a laminar series-parallel graph (LSP) if and only if it satisfies:*

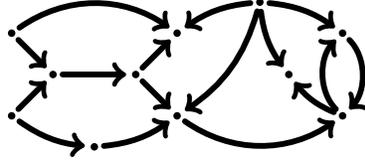
**P1** *For every  $(s, t) \in V^2$ ,  $G\langle s, t \rangle$  is either an  $s$ - $t$ -DSP or contains no edges; and*

**P2**  *$\{E(G\langle e \rangle)\}_{e \in E}$  form a laminar set family, i.e., for all edges  $e_1, e_2 \in E$  we have*

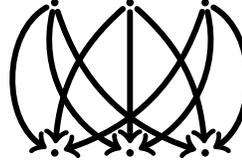
$$G\langle e_1 \rangle \subseteq G\langle e_2 \rangle \vee G\langle e_2 \rangle \subseteq G\langle e_1 \rangle \vee E(G\langle e_1 \rangle) \cap E(G\langle e_2 \rangle) = \emptyset.$$



(a) Graph whose biconnected components are all DSPs or cyclic DSPs



(b) A more complex biconnected LSP



(c) Complete bipartite graph with natural orientation

**Fig. 4.** Examples of LSPs. Every edge represents a DSP of arbitrary size.

Figure 4 shows some example LSPs. LSPs not only include the graphs whose biconnected components are all DSPs but also some cyclic graphs, e.g., *cyclic DSPs* constructed by identifying the source and sink of a DSP. Moreover, there exist very dense LSPs, e.g., the natural orientations of complete bipartite graphs. Below, we present some interesting properties of LSPs; for proofs see Appendix A.

**Theorem 8.** *A directed graph  $G = (V, E)$  satisfies P1 if and only if  $G$  does not contain a subgraph homeomorphic to  $W$  (displayed in Figure 3(left)).*

**Theorem 9.** *Every directed graph  $G$  has a subdivision  $\bar{G}$  that satisfies P2.*

**Theorem 10.** *Every DSP  $G$  is an LSP.*

## 4 Efficient Algorithms

We first present an algorithm that finds an optimal MCPS solution for DSPs in linear time (Section 4.1) and then a quadratic-time algorithm for LSPs (Section 4.2), which can also be applied to several related problems (Section 4.3).

### 4.1 MCPS on Directed Series-Parallel Graphs

Our linear-time algorithm will exploit a useful property of capacities in P1-graphs: if every edge is covered, then *all* vertex pairs are covered.

**Theorem 11.** *Given a retention ratio  $\alpha \in (0, 1)$ , let  $G = (V, E)$  be a P1-graph and  $G' = (V, E')$ ,  $E' \subseteq E$ , with  $c_{G'}(u, v) \geq \alpha \cdot c_G(u, v)$  for all edges  $uv \in E$ . Then,  $E'$  is a feasible  $\alpha$ -MCPS solution, i.e.,  $c_{G'}(u, v) \geq \alpha \cdot c_G(u, v)$  for all vertex pairs  $(u, v) \in V^2$ .*

---

**Algorithm 1** Compute an optimal MCPS in DSPs.

---

**Input:** DSP  $G = (V, E)$ , retention ratio  $\alpha \in (0, 1)$ .

- 1:  $E' \leftarrow E$
  - 2:  $T \leftarrow$  clean series-parallel decomposition tree for  $G$
  - 3: **for each** tree node  $\sigma$  in a bottom-up traversal of  $T$
  - 4:      $(H, (s, t)) \leftarrow$  graph and terminal pair corresponding to  $\sigma$
  - 5:     **if**  $\sigma$  is a P-composition **and**  $st \in E(H)$  **and**  $(s, t)$  is covered by  $(E' \cap E(H)) \setminus \{st\}$
  - 6:         remove  $st$  from  $E'$
  - 7: **return**  $E'$
- 

*Proof.* Consider any vertex pair  $(u, v) \in V^2$  with  $u \neq v$  (as vertex pairs with  $u = v$  are trivially covered). We can assume w.l.o.g. that a maximum flow from  $u$  to  $v$  passes only over edges of  $H := G\langle u, v \rangle$ : for any maximum flow from  $u$  to  $v$  that uses cycles outside of  $H$ , we can find one of the same value in  $H$  by simply removing these cycles. As  $G$  is a P1-graph,  $H$  is a  $u$ - $v$ -DSP. We use induction over the series-parallel composition of  $H$  to prove that  $(u, v)$  is covered. If  $uv \in E$ , the edge is already covered as asserted in the hypothesis of the theorem; this includes the base case of  $H$  containing a single edge.

Let  $H$  be created from the disjoint union of two smaller  $u_i$ - $v_i$ -DSPs  $H_i$ ,  $i \in \{1, 2\}$ , for which the theorem holds. Further, let  $X' := (V(X), E(X) \cap E(G'))$  for any subgraph  $X \in \{H, H_1, H_2\}$  of  $G$ . If  $H$  is constructed from an S-composition, i.e.  $u = u_1, v = v_2$ , and  $v_1 = u_2$ , each maximum  $u$ - $v$ -flow in  $H$  ( $H'$ ) passes through both  $H_1$  and  $H_2$  ( $H'_1$  and  $H'_2$ , resp.):  $c_{H'}(u, v) = \min\{c_{H'_1}(u, v_1), c_{H'_2}(v_1, v)\} \geq \min\{\alpha \cdot c_{H_1}(u, v_1), \alpha \cdot c_{H_2}(v_1, v)\} = \alpha \cdot c_H(u, v)$ . If  $H$  is constructed from a P-composition, i.e.  $u = u_1 = u_2$  and  $v = v_1 = v_2$ , its  $u$ - $v$ -capacity in  $H$  ( $H'$ ) is the sum of  $u$ - $v$ -capacities in  $H_1$  and  $H_2$  ( $H'_1$  and  $H'_2$ , resp.):  $c_{H'}(u, v) = c_{H'_1}(u, v) + c_{H'_2}(u, v) \geq \alpha \cdot c_{H_1}(u, v) + \alpha \cdot c_{H_2}(u, v) = \alpha \cdot c_H(u, v)$ .  $\square$

**Observation 12** *Theorem 11 does in general not apply to graphs for which only their shadow is series-parallel. In particular, it does not even hold for the graph  $W$ —the smallest graph without property P1—when two paths of length 2 are added to it, see Figure 3(right).*

We give a simple linear-time algorithm to solve MCPS in DSPs. The algorithm requires the series-parallel decomposition tree to be *clean*, i.e., if there are multiple P-compositions of several  $s$ - $t$ -DSPs  $H_0, \dots, H_k$  where  $E(H_0) = \{st\}$  is a single edge, we first compose  $H_1, \dots, H_k$  into a common  $s$ - $t$ -DSP  $H$  before composing  $H$  with  $H_0$ . Standard decomposition algorithms can easily achieve this property; the proof below also describes an independent linear-time method to transform a non-clean decomposition tree into a clean one.

**Theorem 13.** *Algorithm 1 is optimal for MCPS on DSPs, taking  $\mathcal{O}(|V|)$  time.*

*Proof.* We use induction over the clean series-parallel decomposition tree  $T$  of  $G$ , maintaining the following invariants: at the end of each for-loop iteration with  $(H, (s, t))$  as the graph and terminal pair for the respective tree node,  $E' \cap E(H)$  is an optimal solution for  $H$ , and all optimal solutions of  $H$  have equal  $s$ - $t$ -capacity.

Let  $\sigma$  be a leaf of  $T$ : A graph  $H$  with a single edge  $st$  only allows one feasible (and hence optimal) solution consisting of its only edge. The edge is added to  $E'$  during the algorithm's initialization and is not removed from it before  $\sigma$  has been processed. Now observe S-compositions and those P-compositions where no edge  $st$  exists in any of the components: They produce no additional paths between the endpoints of any edge (which are the only vertex pairs that have to be covered, see Theorem 11). Thus, the feasible (optimal) solutions of  $H$  are exactly those that can be created by taking the union of one feasible (optimal, respectively) solution for each respective component. The algorithm proceeds accordingly by keeping  $E'$  unchanged. Since the components' respective optimal solutions all have the same source-sink-capacity (per induction hypothesis), this also holds true for their unions, i.e., the optimal solutions of  $H$ .

Now consider a P-composition with  $st \in E(H)$ . As  $T$  is clean, there are two components  $H_1$  and  $H_2$  with  $E(H_2) = \{st\}$ , and  $G(s, t) = H$ . All edges  $e \in H_1$  are covered optimally by  $E' \cap E(H_1)$  both in  $H_1$  and in  $H$  since  $(s, t) \notin E(H \setminus e)$ .

**Case 1:** If one optimal solution for  $H_1$  already covers  $st$  in  $H$ , then all optimal solutions for  $H_1$  do so (as they all have the same  $s$ - $t$ -capacity per induction hypothesis). Then, the optimal solutions for  $H_1$  are exactly the optimal solutions for  $H$ , and the algorithm finds one of them by keeping its solution for  $H_1$  intact and removing  $st$  from  $E'$ . Note that this removal does not affect the feasibility of  $E'$  for subgraphs of  $G \setminus G(s, t)$  that have already been processed.

**Case 2:** If  $st$  is not yet covered by our optimal solution for  $H_1$ , it is not covered by any optimal solution for  $H_1$ . Our algorithm chooses the edge  $st$  by keeping its optimal solutions for both  $H_1$  and  $H_2$ . An optimal solution  $S$  for  $H$  must contain an optimal solution for  $H_1$ :  $S' := S \setminus \{st\}$  covers all edges of  $H_1$ . If  $S'$  were not optimal, there would exist another solution  $S''$  that covers all edges and thus vertex pairs of  $H_1$  with  $|S''| < |S'|$ . But  $S''' := S'' \cup \{st\}$  is feasible for  $H$  because the capacity requirements for vertex pairs in  $H$  and  $H_1$  only differ by at most one. We arrive at  $|S'''| = |S''| + 1 < |S'| + 1 \leq |S|$ , a contradiction. — In addition to an optimal solution for  $H_1$ , we need exactly one more edge to increase the  $s$ - $t$ -capacity and cover  $st$  in  $H$ : this additional edge is either  $st$  itself or another edge from  $H_1$ . Assume that adding an additional edge  $e_1 \in E(H_1)$  (instead of  $st$ ) increases the capacity for  $st$  or a later source-sink-pair by 1, then  $st$  by construction does so as well. Thus, adding  $st$  instead of  $e_1$  is never worse; furthermore, all optimal solutions for  $H$  have the same  $s$ - $t$ -capacity.

For the running time, note that a (clean) series-parallel decomposition tree  $T$  can be computed and traversed in linear time [32]. If  $T$  were not clean, it is trivial to establish this property in linear time: Traverse  $T$  bottom-up; whenever a leaf  $\lambda$  is the child of a P-node, ascend the tree as long as the parents are P-nodes. Let  $\varrho$  be the final such P-node, and  $\gamma$  the other child of  $\varrho$  that was not part of the ascent. Swap  $\lambda$  with  $\gamma$ . Observe that the ascents for different leaves are disjoint, and thus this operation requires overall only  $\mathcal{O}(|T|) = \mathcal{O}(|V|)$  time.

In each step during the traversal in line 3, we can compute the capacity of the current source and sink—for both the current solution and  $G$  overall—in constant time using the values computed in previous steps: a single edge is assigned a

---

**Algorithm 2** Compute an optimal MCPS in LSPs.

---

**Input:** LSP  $G = (V, E)$ , retention ratio  $\alpha \in (0, 1)$ .

- 1:  $E' \leftarrow \emptyset$
  - 2: **for each** edge  $e \in E$
  - 3:      $\mu_e \leftarrow$  number of edges in  $G\langle e \rangle$
  - 4: sort all edges  $e \in E$  by non-descending  $\mu_e$
  - 5: **for each** edge  $e = uv \in E$  in order
  - 6:     **if**  $uv$  is not covered by  $E'$
  - 7:         add  $e$  to  $E'$
  - 8: **return**  $E'$
- 

capacity of 1, and an S-composition (P-composition) is assigned the minimum (sum, respectively) of the capacities of its two components.  $\square$

## 4.2 MCPS on Laminar Series-Parallel Graphs

An intuitive approach to solve MCPS on an LSP  $G = (V, E)$  is based on the observation that every LSP can be partitioned into a set of edge-disjoint DSPs: Consider the *maximal edge-anchored subgraphs (MEASs)*, i.e., those  $G\langle e \rangle$  for  $e \in E$  such that there is no other edge  $e' \in E \setminus \{e\}$  with  $E(G\langle e \rangle) \subseteq E(G\langle e' \rangle)$ . Since LSPs are P1-graphs, each of these MEAS must be a DSP, and it suffices to cover its edges (see Theorem 11). Further, the EAS  $G\langle e'' \rangle$  for each edge  $e'' \in E$  is contained in a single MEAS (as LSPs are P2-graphs). Hence, one could identify the MEASs and run Algorithm 1 on each of them to obtain an optimal MCPS solution. We give a more straightforward but functionally equivalent Algorithm 2.

**Theorem 14.** *Algorithm 2 is optimal for MCPS on LSPs, taking  $\mathcal{O}(|E|^2)$  time.*

*Proof.* We prove by induction over  $\mu_e$  that for each DSP (and hence for each MEAS), Algorithm 2 returns the same result as Algorithm 1: Edges  $e$  with  $\mu_e = 1$  (i.e., MED-edges) are added to  $E'$  by Algorithm 2 in order to cover themselves. Similarly, Algorithm 1 will add such edges during its initialization and never remove them: edge  $e$  would only be removed if  $e$  connected the source and sink of a subgraph constructed with a P-composition, a contradiction to  $\mu_e = 1$ .

Now assume that the edges with a  $\mu$ -value smaller than  $i$  for some  $i > 1$  are already processed equivalently to Algorithm 1. Consider any edge  $e = uv$  with  $\mu_e = i$ . Since  $G$  is a P1-graph,  $H := G\langle e \rangle$  is a  $u$ - $v$ -DSP. As  $e \in E(H)$ ,  $H$  can be constructed with a P-composition from two graphs  $H_1$  and  $H_2$  where  $E(H_2) = \{e\}$ . All edges in  $H_1$  have already been processed (they have a  $\mu$ -value smaller than  $i$ ), and the solutions of Algorithm 2 and Algorithm 1 thus coincide on  $H_1$ . Hence, both algorithms produce the same solution for  $H$  as they both contain  $e$  if and only if  $e$  is not already covered by the current solution for  $H_1$ .

For each MEAS, Algorithm 1 and, as we have now shown, Algorithm 2 both find the smallest subgraph that covers all of its edges. As LSPs are P1-graphs, this suffices to guarantee an optimal MCPS solution for the MEAS by Theorem 11.

Further, we can consider the different MEASs in isolation since their solutions are independent of each other: LSPs are P2-graphs, and thus, for any edge  $uv \in E(G)$ , all  $u$ - $v$ -paths are completely contained in a single MEAS.

It remains to argue the running time. For each  $e = uv \in E$ , we compute  $\mu_e$  by starting a depth-first search at  $u$  and counting tree- and cross-edges when backtracking from  $v$ . Overall, this results in  $\mathcal{O}(|E|^2)$  time. The sorting of the edges can be done in  $\mathcal{O}(|E|)$  time as the domain are integer values between 1 and  $|E|$ . Lastly, to check whether an edge  $uv$  is covered, we precompute the  $s$ - $t$ -capacity for every edge  $st \in E$  on  $G\langle s, t \rangle$  and then, when needed, compute the  $u$ - $v$ -capacity on the graph  $G[E' \cap E(G\langle u, v \rangle)]$  for the current solution  $E'$ . Note that both of these subgraphs are DSPs as  $G$  is a P1-graph. This allows us to compute a series-parallel decomposition tree in  $\mathcal{O}(|E|)$  time and traverse it bottom-up to obtain the capacity (cf. the proof of Theorem 13). Doing so twice for every edge takes  $\mathcal{O}(|E|^2)$  time overall.  $\square$

### 4.3 Applications of Algorithm 2 to Other Problems

Consider the MINIMUM STRONGLY CONNECTED SUBGRAPH (MSCS) problem [24,33], the special case of MED on strongly connected graphs, i.e., graphs where every vertex is reachable from every other vertex. Since there are straightforward reductions from DIRECTED HAMILTONIAN CYCLE to MSCS to MED to MCPS that all use the original input graph of the instance, Algorithm 2 can be adapted to solve these problems as well: To solve MED, one simply has to set  $\alpha = \min_{(s,t) \in V^2} 1/c_G(s,t)$  and then run the algorithm on the input graph. However, with  $\alpha$  set this way, Algorithm 2 does precisely the same as the algorithm for finding the MED on DAGs [2]: it returns all those edges for which there is only one path between their endpoints (namely the edge itself). Hence, our new insight with regards to the MED is that the aforementioned approach does not only solve MED optimally on DAGs, but on arbitrary LSPs as well. Moreover, if the input graph is strongly connected (Hamiltonian), the returned MED forms the MSCS (directed Hamiltonian cycle, respectively).

**Corollary 15.** *There are quadratic time algorithms that return optimal solutions for DIRECTED HAMILTONIAN CYCLE, MINIMUM STRONGLY CONNECTED SUBGRAPH and MINIMUM EQUIVALENT DIGRAPH on any LSP.*

## 5 Conclusion and Open Questions

We have laid the groundwork for research into capacity-preserving subgraphs by not only showing the NP-hardness of MCPS on DAGs but also presenting a first inapproximability result as well as two algorithmically surprisingly simple algorithms for MCPS on DSPs and LSPs. Several questions remain, for example: Is MCPS on undirected graphs (which is a generalization of MINIMUM SPANNING TREE) NP-hard? Is it NP-hard to approximate MCPS within a sublogarithmic factor? Is there a linear-time algorithm for MCPS on LSPs? Moreover, one may investigate MCPS with non-unit edge capacities, or other problems on LSPs.

## References

1. Ahmed, A.R., Bodwin, G., Sahneh, F.D., Hamm, K., Jebelli, M.J.L., Kobourov, S.G., Spence, R.: Graph spanners: A tutorial review. *Comput. Sci. Rev.* **37**, 100253 (2020), <https://doi.org/10.1016/j.cosrev.2020.100253>
2. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. *SIAM J. Comput.* **1**(2), 131–137 (1972), <https://doi.org/10.1137/0201008>
3. Althöfer, I., Das, G., Dobkin, D.P., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. *Discret. Comput. Geom.* **9**, 81–100 (1993), <https://doi.org/10.1007/BF02189308>
4. Andersen, R., Feige, U.: Interchanging distance and capacity in probabilistic mappings. *CoRR* **abs/0907.3631** (2009), <https://arxiv.org/abs/0907.3631>
5. Andoni, A., Gupta, A., Krauthgamer, R.: Towards  $(1 + \epsilon)$ -approximate flow sparsifiers. In: *Proc. SODA 2014*. pp. 279–293. SIAM (2014), <https://doi.org/10.1137/1.9781611973402.20>
6. Benczúr, A.A., Karger, D.R.: Approximating  $s$ - $t$  minimum cuts in  $\tilde{O}(n^2)$  time. In: *Proc. STOC 1996*. pp. 47–55. ACM (1996), <https://doi.org/10.1145/237814.237827>
7. Benczúr, A.A., Karger, D.R.: Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.* **44**(2), 290–319 (2015), <https://doi.org/10.1137/070705970>
8. Cen, R., Cheng, Y., Panigrahi, D., Sun, K.: Sparsification of directed graphs via cut balance. In: *Proc. ICALP 2021. LIPIcs*, vol. 198, pp. 45:1–45:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021), <https://doi.org/10.4230/LIPIcs.ICALP.2021.45>
9. Chiaraviglio, L., Mellia, M., Neri, F.: Reducing power consumption in backbone networks. In: *Proc. ICC 2009*. pp. 1–6. IEEE (2009), <https://doi.org/10.1109/ICC.2009.5199404>
10. Cisco: Cisco annual internet report (2018–2023). <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf> (03 2020)
11. Cohen, M.B., Kelner, J.A., Kyng, R., Peebles, J., Peng, R., Rao, A.B., Sidford, A.: Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In: *Proc. FOCS 2018*. pp. 898–909. IEEE Computer Society (2018), <https://doi.org/10.1109/FOCS.2018.00089>
12. Cohen, M.B., Kelner, J.A., Peebles, J., Peng, R., Rao, A.B., Sidford, A., Vladu, A.: Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In: *Proc. STOC 2017*. pp. 410–419. ACM (2017), <https://doi.org/10.1145/3055399.3055463>
13. Dahl, G.: The design of survivable directed networks. *Telecommun. Syst.* **2**(1), 349–377 (1993), <https://doi.org/10.1007/BF02109865>
14. Dahl, G.: Directed steiner problems with connectivity constraints. *Discret. Appl. Math.* **47**(2), 109–128 (1993), [https://doi.org/10.1016/0166-218X\(93\)90086-4](https://doi.org/10.1016/0166-218X(93)90086-4)
15. Dinur, I., Steurer, D.: Analytical approach to parallel repetition. In: *Proc. STOC 2014*. pp. 624–633. ACM (2014), <https://doi.org/10.1145/2591796.2591884>
16. Duffin, R.J.: Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications* **10**(2), 303–318 (1965), [https://doi.org/10.1016/0022-247X\(65\)90125-3](https://doi.org/10.1016/0022-247X(65)90125-3)
17. Englert, M., Gupta, A., Krauthgamer, R., Räcke, H., Talgam-Cohen, I., Talwar, K.: Vertex sparsifiers: New results from old techniques. *SIAM J. Comput.* **43**(4), 1239–1262 (2014), <https://doi.org/10.1137/130908440>

18. Essiambre, R., Tkach, R.W.: Capacity trends and limits of optical communication networks. *Proc. IEEE* **100**(5), 1035–1055 (2012), <https://doi.org/10.1109/JPROC.2012.2182970>
19. Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O., Smaragdakis, G.: The lockdown effect: Implications of the COVID-19 pandemic on internet traffic. In: *IMC '20*. pp. 1–18. ACM (2020), <https://doi.org/10.1145/3419394.3423658>
20. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman (1979)
21. Jain, K.: A factor 2 approximation algorithm for the generalized steiner network problem. *Comb.* **21**(1), 39–60 (2001), <https://doi.org/10.1007/s004930170004>
22. Karp, R.M.: Reducibility among combinatorial problems. In: *Proc. COCO 1972*. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972), [https://doi.org/10.1007/978-1-4684-2001-2\\_9](https://doi.org/10.1007/978-1-4684-2001-2_9)
23. Kerivin, H., Mahjoub, A.R.: Design of survivable networks: A survey. *Networks* **46**(1), 1–21 (2005), <https://doi.org/10.1002/net.20072>
24. Khuller, S., Raghavachari, B., Young, N.E.: Approximating the minimum equivalent digraph. *SIAM J. Comput.* **24**(4), 859–872 (1995), <https://doi.org/10.1137/S0097539793256685>
25. Moitra, A.: Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In: *Proc. FOCS 2009*. pp. 3–12. IEEE Computer Society (2009), <https://doi.org/10.1109/FOCS.2009.28>
26. Moshkovitz, D.: The projection games conjecture and the np-hardness of  $\ln n$ -approximating set-cover. *Theory Comput.* **11**, 221–235 (2015), <https://doi.org/10.4086/toc.2015.v011a007>
27. Räcke, H.: Optimal hierarchical decompositions for congestion minimization in networks. In: *Proc. STOC 2008*. pp. 255–264. ACM (2008), <https://doi.org/10.1145/1374376.1374415>
28. Richey, M.B., Parker, R.G., Rardin, R.L.: An efficiently solvable case of the minimum weight equivalent subgraph problem. *Networks* **15**(2), 217–228 (1985), <https://doi.org/10.1002/net.3230150207>
29. Sahni, S.: Computationally related problems. *SIAM J. Comput.* **3**(4), 262–279 (1974), <https://doi.org/10.1137/0203021>
30. Schüller, T., Aschenbruck, N., Chimani, M., Horneffer, M., Schnitter, S.: Traffic engineering using segment routing and considering requirements of a carrier IP network. *IEEE/ACM Trans. Netw.* **26**(4), 1851–1864 (2018), <http://doi.ieeecomputersociety.org/10.1109/TNET.2018.2854610>
31. Spielman, D.A., Teng, S.: Spectral sparsification of graphs. *SIAM J. Comput.* **40**(4), 981–1025 (2011), <https://doi.org/10.1137/08074489X>
32. Valdes, J., Tarjan, R.E., Lawler, E.L.: The recognition of series parallel digraphs. *SIAM J. Comput.* **11**(2), 298–313 (1982), <https://doi.org/10.1137/0211023>
33. Vetta, A.: Approximating the minimum strongly connected subgraph via a matching lower bound. In: *Proc. SODA 2001*. pp. 417–426. ACM/SIAM (2001), <https://dl.acm.org/doi/10.5555/365411.365493>
34. Wong, R.T.: Telecommunications network design: Technology impacts and future directions. *Networks* **77**(2), 205–224 (2021), <https://doi.org/10.1002/net.21997>
35. Zhang, M., Yi, C., Liu, B., Zhang, B.: GreenTE: Power-aware traffic engineering. In: *Proc. ICNP 2010*. pp. 21–30. IEEE Computer Society (2010), <https://doi.org/10.1109/ICNP.2010.5762751>

36. Zhang, Y., Zhao, Z., Feng, Z.: Towards scalable spectral sparsification of directed graphs. In: Proc. ICSS 2019. pp. 1–2. IEEE (2019), <https://doi.org/10.1109/ICSS.2019.8782449>
37. Zhao, L., Nagamochi, H., Ibaraki, T.: A linear time  $5/3$ -approximation for the minimum strongly-connected spanning subgraph problem. Inf. Process. Lett. **86**(2), 63–70 (2003), [https://doi.org/10.1016/S0020-0190\(02\)00476-3](https://doi.org/10.1016/S0020-0190(02)00476-3)

## APPENDIX

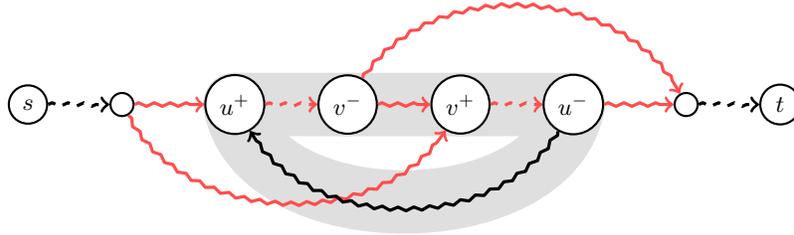
### A Proofs of LSP Properties

In the following, we give proofs for the theorems presented in Section 3 that concern properties of LSPs. The first one states that P1-graphs can be characterized as those graphs that do not contain a  $W$ -subdivision:

**Theorem 8.** *A directed graph  $G = (V, E)$  satisfies P1 if and only if  $G$  does not contain a subgraph homeomorphic to  $W$  (displayed in Figure 3(left)).*

*Proof.* Consider the smallest subgraph  $H$  of  $G$  that is a subdivision of  $W$ , if it exists. Let  $s$  and  $t$  denote the source and sink of  $H$  respectively, and note that  $H \subseteq G\langle s, t \rangle$ . Hence,  $G\langle s, t \rangle$  cannot be a DSP, and  $G$  is not a P1-graph.

If  $G$  is not a P1-graph, then it contains a vertex pair  $(s, t)$  such that  $H' := G\langle s, t \rangle$  is neither empty nor a DSP. Since  $H'$  contains exactly one source and one sink, Theorem 6 lets us conclude that  $H'$  (and thus  $G$ ) either contains a subgraph homeomorphic to  $W$  or a cycle  $C$ . In the latter case, consider a set of  $s$ - $t$ -paths  $\mathcal{P}$  that is inclusion-wise minimal with respect to spanning  $C$ . Let one such path  $P \in \mathcal{P}$  enter  $C$  for the first time at  $u^+$  and exit it for the last time at  $u^-$ . Since no single  $s$ - $t$ -path may contain  $C$  completely, there must be other  $s$ - $t$ -paths that contain the cycle's edge(s) starting at  $u^-$  and going to  $u^+$ . At the same time, these paths may not contain  $P \cap C$  completely as  $\mathcal{P}$  is inclusion-wise minimal. So consider the subpath  $S$  of  $C$  from  $u^+$  to  $u^-$ . There exists an  $s$ - $t$ -path which exits  $C$  at a vertex  $v^-$  and one entering  $C$  at the vertex  $v^+$  such that  $v^-$  comes after (or is identified with)  $u^+$  and  $v^+$  comes before (or is identified with)  $u^-$  along  $S$ . Moreover,  $v^-$  comes strictly before  $v^+$  along  $S$  as there exists at least one edge between them which is only contained in  $P$ , and not in any other path from  $\mathcal{P} \setminus \{P\}$  (since  $\mathcal{P}$  is inclusion-wise minimal). Thus, we can find a subdivision of  $W$  in  $H'$  (and consequently in  $G$ ) as shown in Figure 5.  $\square$



**Fig. 5.**  $G\langle s, t \rangle$  containing a cycle (shaded in gray). The wiggly lines denote paths. Dashed paths may have length 0, others must contain at least one edge. Red paths visualize the  $W$ -subdivision. Note that while  $u^+$  and  $u^-$  lie on the cycle, the subpath of  $P$  between them may leave and enter the cycle repeatedly.

Unlike P1-graphs, there can be no characterization of P2-graphs via forbidden subdivisions or (directed versions of) minors:

**Theorem 9.** *Every directed graph  $G$  has a subdivision  $\bar{G}$  that satisfies P2.*

*Proof.* Construct  $\bar{G}$  by subdividing every edge of  $G$ . Each edge  $e \in E(\bar{G})$  is either the only incoming or the only outgoing edge at some subdivision vertex. Thus,  $\bar{G}\langle e \rangle$  contains only edge  $e$ . As all  $\bar{G}\langle e \rangle$  are disjoint,  $\bar{G}$  satisfies P2.  $\square$

Nonetheless, it is easy to see:

**Theorem 10.** *Every DSP  $G$  is an LSP.*

*Proof.*  $G$  satisfies P1 by Theorem 8. For P2, we use induction over the directed series-parallel composition of  $G$ . A graph with a single edge  $e$  satisfies P2 since there is no other EAS that  $G\langle e \rangle$  could intersect with. Now consider the creation of a new  $s$ - $t$ -DSP  $G$  from two DSPs  $G_1$  and  $G_2$ , for which P2 holds. If  $st \notin E(G)$ , the composition does not create any new paths between the endpoints of any edge, so  $G\langle e \rangle = G_i\langle e \rangle$  for every  $e \in E(G_i)$ ,  $i \in \{1, 2\}$ ; we already know that  $\{E(G_i\langle e \rangle)\}_{e \in E(G_i)}$  for  $i \in \{1, 2\}$  form laminar set families, respectively, so  $\{E(G\langle e \rangle)\}_{e \in E(G_1) \cup E(G_2)}$  does so as well. Finally, assume  $st \in E(G)$ . We only have to show that the laminar set family  $\{E(G\langle e \rangle)\}_{e \in E(G) \setminus \{st\}}$  remains laminar when  $E(G\langle s, t \rangle)$  is added to it. This is the case since  $G\langle s, t \rangle$  fully contains  $G$ .  $\square$