# Visualizing, Finding and Packing Dijoins

F. B. Shepherd[†] and A. Vetta[§]

*March 16, 2003*

**Abstract.** We consider the problem of making a directed graph *strongly connected*. To achieve this, we are allowed for assorted costs to make some of the arcs bidirectional (or, equivalently, to add the reverse of the arc). A successful set of arcs, called a *dijoin*, must intersect every directed cut. Lucchesi and Younger gave a min-max theorem precisely for the problem of finding a dijoin of minimum cost. Another question of interest that we examine is the extent to which dijoins pack. One difficulty in handling dijoins is that they are not as easily visualized as other combinatorial objects such as matchings, trees or flows. We describe two results which act as visual certificates for dijoins. One of these, called a lobe decomposition, resembles Whitney's ear decomposition for 2-connected graphs. The lobe decomposition leads to a natural characterization of when a dijoin is optimal. This characterization, in turn, suggests a framework for generating minimum cost dijoins. We show how Frank's primal-dual combinatorial algorithm to find a minimum cost dijoin can be viewed in this framework. We also show how his algorithm can be simply described and implemented as a primal algorithm, using only greedy arborescence growing procedures. This runs in $O(n^2m)$ time, matching the best time due to Gabow. (The actual running time is $O(nm + nJ(n))$, where $J(n)$ denotes the time needed to update Frank's "jumping arcs" after performing a cycle augmentation.) On the question of packing, we look at the problem which is the "integer dual" to the Lucchesi-Younger Theorem. We consider the function $f(k)$ which is the maximum value such that every weighted directed graph whose minimum weight directed cut has weight at least $k$, admits a collection of $f(k)$ disjoint dijoins. We ask whether this function approaches infinity. It is not currently known whether $f(k_0) \geq 2$ for some $k_0$. For $k_0 = 2$, we give a simple condition on the arc weights that guarantees the existence of two disjoint dijoins. Moreover, for any $k$, we show the existence of a half-integral dijoin packing of size $\frac{k}{2}$. Towards the general goal, we introduce a concept of *weak submodular flow polyhedra* and give a conjecture concerning when their integer hulls are non-empty.

## 1. Introduction

We consider the basic problem of *strengthening* a network so that it becomes strongly connected. That is, we require that there be a directed path between any pair of nodes in both directions. To achieve this goal we are allowed to add to the graph (at varying costs) the complements of some of the network arcs. Equivalently, we are searching for a collection of arcs that induce a strongly connected graph when they are either contracted or made bi-directional. It is easy to see that our problem is that of finding a *dijoin*, a set of arcs that intersects every directed cut.

Despite its fundamental nature, the *minimum cost dijoin problem* is not a standard modelling tool for the combinatorial optimizer in the same way that shortest paths, matchings and network flows are. We believe that widespread adoption of these other problems stems from the fact that they can be tackled using standard concepts such as dynamic programming, greedy algorithms, shrinking, and tree growing. Consequently, simple and efficient algorithms can be implemented and also, importantly, taught. One objective of this paper, therefore, is to examine dijoins using classical combinatorial optimization techniques such as decomposition, arborescence growing and negative cycle detection.

[†]Math Sciences, Bell Laboratories, Murray Hill. Email: bshep@research.bell-labs.com

[§]School of Computer Science, McGill University, Montreal. Email: vetta@cs.mcgill.ca

Under this framework, we present a primal version of Frank's seminal primal-dual algorithm for finding a minimum cost dijoin. The running time of our implementation is $O(n^2m)$, which matches the fastest known running time (due to Gabow). We also consider the question of packing dijoins and along the way we present open problems which hopefully serve to show that the theory of dijoins is still a rich and evolving topic. We begin, though, with some dijoin history.

### 1.1. Background.

We start by discussing the origins of the first polytime algorithm (based on the Ellipsoid Method) for this problem: the Lucchesi-Younger Theorem. A natural lower bound on the number of arcs in a dijoin is the size of any collection of disjoint directed cuts. The essence of Lucchesi-Younger is to show that such lower bounds are strong enough to certify optimality. A natural generalization also holds when an integer cost vector $c$ is given on the arcs. A collection $\mathcal{C}$ of directed cuts is a *c-packing* if for any arc $a$, at most $c_a$ of the cuts in $\mathcal{C}$ contain $a$.

**Theorem 1.1** (Lucchesi-Younger Theorem [25]). *Let $D$ be a digraph with a cost $c_a$ on each arc. Then*

$$\min\{\sum_{a\in T} c_a : \ T \textit{ is a dijoin}\} = \max\{|\mathcal{C}| : \mathcal{C} \textit{ is a c-packing}\}.$$

This was first conjectured for planar graphs in the thesis of Younger [35]. It was conjectured for general graphs in 1969 [36] and independently by Robertson (cf. [25]). (A preliminary result for bipartite graphs appeared in [26].) This theorem generated great interest after it was announced at the 1974 International Congress of Mathematicians in Vancouver. It proved also to be a genesis of sorts; we describe now some of the historical developments and remaining questions in combinatorial optimization which grew from it.

Consider the $0,1$ matrix $C$ whose rows correspond to the incidence vectors of directed cuts in $D$. Theorem 1.1 implies that the dual of the linear program $\min\{cx : Cx \geq 1, x \geq 0\}$ always has an integral optimum for each integral vector $c$. Obviously, the primal linear program also always has integral optimal solutions since any $0,1$ solution identifies a dijoin. Matrices with this primal integrality property are called *ideal*. As discussed shortly, this new class of ideal matrices did not behave as well as predecessors, such as matrices arising from bipartite matchings and network flows, each of which consisted of totally unimodular matrices.

Consider next an *integer dual* of the minimum dijoin problem where we reverse the roles of what is being minimized with what is being packed. Formally, for a $0,1$ matrix $C$, its *blocking matrix* is the matrix whose rows are the minimal $0,1$ solutions $x$ to $Cx \geq 1$. A result of Lehman [22] states that a matrix $C$ is ideal if and only if its blocking matrix is ideal. Note that the blocking matrix of our directed cut incidence matrix $C$, is just the dijoin incidence matrix, which we denote by $M$. It follows that a minimum cost directed cut in a graph is obtained by solving the linear program $\min\{cx : Mx \geq 1, x \geq 0\}$.

Unlike the directed cut matrix, however, Schrijver [29], showed that the dual of this linear program does not always possess an integral optimum. In particular, this implies that $M$ is not totally unimodular. His example is a weighted digraph whose minimum weight directed cut is 2, but for which there does not exist a pair of disjoint dijoins amongst the arcs of positive weight. The example depends critically on the use of zero-weight arcs. Indeed, the following long-standing unweighted conjecture is still unresolved.

**Woodall's Conjecture.** *The minimum cardinality of a directed cut equals the maximum cardinality of a collection of disjoint dijoins.*

Schrijver [30] and Feofiloff and Younger [10] verified the conjecture (even in the weighted case) if the digraph is *source-sink connected*, that is, there is a dipath from each source to each sink. Note that restricting the conjecture to planar graphs, and then translating to the dual map, one has the following question which is also open. In a planar digraph with no directed cycle of length less than $k$, there is a partition of its arcs into $k$ disjoint *feedback arc sets* (collections of arcs whose deletion destroys all directed cycles)[1]. Observe that two disjoint feedback arc sets can be trivially found in any graph. If we take an ordering of the nodes $v_1, v_2, \ldots, v_n$ then $A' = \{(v_i, v_j) \in A : i < j\}$ and $A - A'$ are feedback arc sets. Nothing is evidently known for $k > 2$, except in the case of series-parallel digraphs for which the problem was recently settled [21].

We define a function $f(k)$ which is the maximum value such that every weighted digraph, whose minimum weight directed cut is at least $k$, contains $f(k)$ disjoint dijoins. We ask whether $f(k)$ goes to infinity as $k$ increases. We do not even currently know if $f(k_0) \geq 2$ for some $k_0$. As suggested by Pulleyblank [28], one tricky aspect in verifying Woodall's Conjecture is that dijoins are not as easily visualized as directed cuts themselves or other combinatorial objects such as trees, matchings, flows etc. For a given subset $T$ of arcs, one must resort to checking whether each directed cut does indeed include an element of $T$. We thus, in Section 2, devise two "visual" certificates for dijoins. One of these, called a *lobe decomposition*, resembles Whitney's well-known ear decompositions for 2-connected and strongly connected graphs. This decomposition is used later to define augmenting structures for dijoins, and it also immediately implies the following.

**Theorem 2.7** *Let $D$ be a digraph with arc weights $w$ whose minimum directed cut is of weight at least 2. If each component of the graph induced by the support of $w$ induces a 2-edge-connected undirected graph, then $D$ contains two disjoint dijoins.*

In Section 4 we discuss an approach to determining whether there exists a constant $k_0$ such that $f(k_0) \geq 2$. We actually conjecture the following.

**Conjecture 1.2.** *Let $D$ be a weighted digraph whose minimum weight directed cut is of size $k$. Then there is an $O(k)$ weighted packing of dijoins in $D$.*

We prove a weaker result: there always exists such a "large" half-integral packing.

**Theorem 1.3.** *Let $D$ be a weighted digraph whose minimum weight dicut is of size $k$. Then there is a half-integral packing of dijoins of size $\frac{k}{2}$.*

[1] It was actually the feedback arc problem which originally motivated Younger in [35].

We also discuss a possible approach for finding integral packings based on a notion of weak supermodularity. This is related to a number of other recent results on generalized Steiner problems in undirected graphs and on network design problems with orientation constraints. Weak submodular flows may themselves be an interesting direction for future research.

1.2. Algorithms.

Frank's [8] original $O(n^3m)$ combinatorial algorithm for the minimum dijoin problem[2] is a primal-dual algorithm which essentially looks for an augmenting cycle of negative cost much like Klein's cycle cancelling algorithm for minimum cost network flows. It differs in two main respects. First, in some iterations, no progress is made in the primal, but rather some of the dual variables are altered. Second, the negative cycles are not computed in the residual digraph associated with the current dijoin. Rather, such cycles are computed in an extended graph which contains new arcs called *jumping arcs*; these may not correspond to any arc in the original digraph. Avoiding jumping arcs altogether is a difficult task, but there are two reasons to attempt this. The first is that it is more natural to work in a residual digraph associated with the original digraph; for example, this is what is done for minimum cost network flows. The second is that computation of these arcs has proved to be the bottleneck operation in terms of running time. Frank's original algorithm computed these arcs in time $O(n^2m)$ per iteration. Gabow developed a sophisticated theory of centroids and used it to compute these arcs in $O(nm)$ time. In Section 3.2 we discuss a simple primal $O(n^2m)$ implementation of Frank's algorithm. There we give a simple $O(nm)$ algorithm for computing the jumping arcs. Putting this together, we exhibit an $O(n^2m)$ algorithm which is based only on negative cycle detection and arborescence growing routines.

The Lucchesi-Younger Theorem led Edmonds and Giles to develop *submodular flows*, a common generalization of network flows and dijoins. Frank's algorithm for dijoins proved to be the prototype for the original "combinatorial" algorithms for submodular flows, see for example [5]. In particular, the notion of jumping arc carried over to that of an *exchange capacity* which is at the heart of every algorithm for submodular flows. Computation of exchange capacities corresponds to the problem of minimizing a submodular function (a problem for which combinatorial algorithms have only recently been devised [15, 31]).

Recently, it was shown [11] how to compute minimum cost submodular flows without an explicit call to an submodular flow minimization routine; instead they capitalize on a structural result of [31] on submodular flow extreme points. In a similar vein, we seek optimality conditions in terms of the original topology given by $D$. In this direction, we describe a cycle *flushing* operation (similar to augmentations), inspired by the above-mentioned lobe decomposition, which allows one to work in a modified auxiliary graph whose arcs are parallel to those of $D$ (and so no expensive computation is required to build it). We show that any pair of minimal dijoins can be obtained from one another by a sequence of cycle flushings. Unlike network flows, however, we may not always be guaranteed to be able to improve the cost on each flushing operation. That is, we may not restrict to negative cost cycle augmentations. We show instead that a dijoin is optimal if and only if there is no negative cost strongly

[2] More complex algorithms for the problem were found also by Lucchesi [24] and Karzanov [18].

connected subgraph structure in the modified auxiliary graph. Specifically, for a dijoin $T$ we define a residual graph $\mathcal{D}(T)$ which includes the arcs $T \cup (\overline{A-T})$ each with cost zero, the arcs $A - T$ each with their original cost, and the arcs $\bar{T}$ each with the negative of their original cost (for a set $X$ of arcs, $\bar{X}$ denotes a set of new arcs associated with the reverse arcs for those in $X$).

**Theorem 3.3** *A dijoin $T$ is optimal if and only if $\mathcal{D}(T)$ contains no negative cost strongly connected subgraph, without any negative cycle of length two.*

Detecting such negative cost subgraphs is NP-hard in general, as shown in Section 3.1.2, although in this setting there is evidently a polytime algorithm to find such subgraphs. This result effectively determines a "test set" for local search. In other words, call two dijoins *neighbourly* whenever one can be obtained from the other by flushing along the cycles in such a strong strongly connected subgraph. Then the result implies that the set of neighbourly pairs, includes all adjacent dijoins on the dijoin polyhedron. We have not characterized the adjacent pairs however (cf. [2] where adjacent sets on the stable set polytope are characterized).

## 2. Visualizing and Certifying Dijoins

In this section we describe two "visual certificates" concerning whether a set of arcs forms a dijoin. They have a similar flavour but are of independent use depending on the setting. First we introduce the necessary notation and definitions. We consider a digraph $D = (V, A)$. For a nonempty, proper subset $S \subseteq V$, we denote by $\delta^+(S)$ the set of arcs with with tail in $S$ and head in $V - S$. We let $\delta^-(S) = \delta^+(V - S)$. We also set $\delta(S) = \delta^+(S) \cup \delta^-(S)$ and call $\delta(S)$ the cut *induced* by $S$. A cut is *directed* if $\delta^+(S)$, or $\delta^-(S)$, is empty. We then refer to the set of edges as a *directed cut* and call $S$ its *shore*. The shore of a directed cut induces an *in-cut* if $\delta^+(S) = \emptyset$, and an *out-cut* otherwise. Clearly if $S$ induces an out-cut, then $V - S$ induces an in-cut. We may abuse terminology and refer to the cut $\delta(S)$ by its *shore* $S$.

A *dijoin* $T$ is a collection of arcs that intersects every directed cut, i.e., at least one arc from each cut is present in the dijoin. An arc, $a \in T$, is said to be *critical* if, for some directed cut $\delta^+(S)$, it is the only arc in $T$ belonging to the cut. In this case we say that $\delta^+(S)$ is a *justifying cut* for $a$. A dijoin is *minimal* if all its arcs are critical.

Observe that if $a \in A$ induces a cut edge in the *underlying undirected graph* then a set of arcs $A'$ is a dijoin if and only if $a \in A'$ and $A' - \{a\}$ is a dijoin of $D - a$. Thus, we make the assumption throughout that the underlying undirected graph is 2-edge connected. Notice, also, that we may assume that the graph is acyclic since we may contract the nodes in a directed cycle without affecting the family of directed cuts.

A (simple) *path* $P$ in an undirected graph is defined as an alternating sequence $v_0, e_0, v_1, e_1, \ldots, e_{l-1}, v_l$ of nodes and edges such that each $e_i$ has endpoints $v_i$ and $v_{i+1}$. We also require that none of the nodes are repeated, except possibly $v_0 = v_l$ in which case the path is called a *cycle*. Note that the path $v_l, e_{l-1}, v_{l-1}, \ldots, e_0, v_0$ is distinct from $P$, and is called the *reverse* of $P$, denoted by $P^-$. A *path* in a digraph $D$ is defined similarly as a sequence $P = v_0, a_0, v_1, a_1, \ldots, a_{l-1}, v_l$ of nodes and arcs where for each $i$, the head and tail of $a_i$ are $\{v_i, v_{i+1}\}$. If the head of $a_i$ is $v_{i+1}$, then it is called a *forward* arc of $P$. Otherwise, it is called *backward* arc. Such a path is called a cycle if $v_0 = v_l$. The path (cycle) is *directed* if every arc is forward. Finally, for an arc $a$ with tail $s$ and head $t$, we denote by $\bar{a}$ a new arc associated to $a$ with tail $t$ and head $s$. For a subset $F \subseteq A$, we let $\bar{F} = \{\bar{a} : a \in F\}$. We also let $\bar{D}$ denote the digraph $(V, \bar{A})$.

2.1. A Decomposition via Cycles.

A cycle (or path), $C$, is *complete* with respect to a dijoin, $T$, if all of its forward arcs lie in $T$. In [26] the notion of a complete path (called *minus paths*) is already used. A complete cycle (or path) is *flush* if, in addition, none of its backward arcs is in the dijoin.

**Lemma 2.1.** *Let $T$ be a dijoin of $D$ and $C$ be a complete cycle in $D$. Then either $C$ is flush or $T$ is not minimal.*

**Proof.** Suppose that $a \in T$ is a reverse arc of $C$ and $\delta^+(S)$ is a justifying cut for $a$. Since $\delta^+(S)$ is a directed cut, $C$ intersects it in an equal number of forward and reverse arcs. In particular, $\delta^+(S) \cap T$ contains some forward arc of $C$. It follows that $T - \{a\}$ is a dijoin. □

**Theorem 2.2.** *Every dijoin contains a complete cycle.*

**Proof.** We actually show that every non-dijoin arc is contained in a complete cycle. Since the underlying graph is 2-edge connected there is a non-dijoin arc (otherwise any cycle in $D$ is complete). Let $a = (s, t)$ be such an arc. We grow a tree $\mathcal{T}$ rooted at $s$. Initially, let $V(\mathcal{T}) = s$ and $A(\mathcal{T}) = \emptyset$. At each step we consider a node $v \in \mathcal{T}$ which we have yet to examine and consider the arcs in $\delta(v) \cap \delta(\mathcal{T})$. We add such an arc $a'$ to $\mathcal{T}$ if $a' \in \delta^-(\mathcal{T})$ or if $a' \in \delta^+(\mathcal{T}) \cap T$. It follows that this process terminates either when $\mathcal{T} = V$ or when $\delta(\mathcal{T})$ consists only of out-going, non-dijoin arcs. This later case can not occur, otherwise $\delta(\mathcal{T})$ is a directed cut that does not intersect the dijoin $T$. So we have $t \in T$. In addition $a = (s, t)$ is not in $\mathcal{T}$ as $a$ was an out-going, non-dijoin arc when it was examined in the initial stage. Now take the path $P \in \mathcal{T}$ connecting $s$ and $t$. All of its forward arcs are dijoin arcs and by adding $a$ we obtain a cycle, $C$, in which $a$ is a backward arc. Hence all the forward arcs in $C$ are dijoin arcs and so $C$ is a complete cycle. □

The preceding tree algorithm also gives the following useful check of whether or not a dijoin arc is critical.

**Lemma 2.3.** *A dijoin arc $a = (s, t)$ is critical if and only if there is no flush path from $s$ to $t$ path in $D - \{a\}$. Moreover, any such arc lies on a flush cycle.*

**Proof.** This is equivalent to the following. A dijoin arc $a = (s, t)$ is critical if and only if the tree algorithm fails to find an $s - t$ path when applied to $D - \{a\}$. To prove this, remove $a$ from $D$ and grow $\mathcal{T}$ from $s$ as before. If $t \in \mathcal{T}$ upon termination of the algorithm then let $P$ be the path from $s$ to $t$ in $\mathcal{T}$. All the forward arcs in $P$ are in the dijoin. So $C = P \cup \{a\}$ is a complete cycle. However, $a$ is a backward arc with respect to $C$ and it too is in the dijoin. Thus, by Lemma 2.1, $a$ is not critical.

Suppose then, on termination, $t \notin \mathcal{T}$ (notice that since we have removed a dijoin arc it is possible that the algorithm terminates with $\mathcal{T} \neq V$). Clearly $\delta(\mathcal{T})$ is a justifying cut for $a$, and so $a$ is critical. $\square$

**Corollary 2.4.** *A dijoin $T$ is minimal if and only if every complete cycle in $T$ is flush.* $\square$

Consider a complete cycle $C$ with respect to $T$. We denote by $D \star C$ (there is an implied dependency on $T$) the digraph obtained by contracting $C$ to a single node, and then contracting the strong component containing that node to make a new node $v_C$. The subgraph of $D$ corresponding to $v_C$ is denoted by $H(C)$. Hence, $D \star C = D/H(C)$, where / denotes the contraction operation. Observe that any cut of $D$ that splits the nodes of $H(C)$ either contains a dijoin arc $a \in T \cap C$ or is not a directed cut. This observation lies behind our consideration of the digraph $D \star C$.

**Lemma 2.5.** *Let $D$ be an acyclic digraph and $T$ a dijoin. If $C$ is a complete cycle, then $T - H(C)$ is a dijoin in $D \star C$. If $T$ is minimal in $D$ then $T - A(C)$ is also minimal in $D \star C$.*

**Proof.** First, note that any directed cut in $D \star C$ corresponds to a directed cut, $\delta^+(S)$, in $D$ (with $V(H(C)) \subseteq S$ or $V(H(C)) \subseteq V - S$). It follows that $T - V(C)$ is a dijoin in $D \star C$. Now suppose that $T$ is minimal. We first show that every arc $a \in H(C) - A(C)$ is contained in a complete cycle $C_a$ whose forward arcs are a subset of $C$'s forward arcs. For if $a$ is such an arc, then since $H(C)/V(C)$ is strongly connected, there is a directed path $P$ which contains $a$, is internally disjoint from $V(C)$, and whose endpoints lie in $V(C)$; the endpoints, say $x$ and $y$, are distinct as $P$ is itself not a directed cycle in $D$. We may then identify a complete cycle by traversing a subpath of $C$ from $x$ to $y$, and then traversing $P$ in the reverse direction. Lemma 2.1 now implies that no arc of $P$ lies in $T$. In particular, this shows that $T \cap H(C) - A(C) = \emptyset$.

Now consider $a \in T - A(C)$ and let $S_a$ be a justifying cut in $D$ for $a$. If $V(H(C)) \subseteq S_a$ or $V(H(C)) \cap S_a = \emptyset$, then it is a justifying cut for $a$ (with respect to $T - A(C)$) in $D \star C$. Otherwise, $\delta^+(S_a)$ must contain an arc $a' \in H(C) - A(C)$. But then $C_{a'}$ must intersect $\delta^+(S_a)$ in some arc of $T \cap C$, contradicting the fact that $\delta^+(S_a) \cap T = \{a\}$. $\square$

Set $D_0 = D$ and let $C_0$ be a complete cycle in $D_0$. A *lobe decomposition* (which supports $T$) is a sequence $\mathcal{S} = \{C_0, \mathcal{P}^0, C_1, \mathcal{P}^1, \ldots, C_k, \mathcal{P}^k\}$ such that

- For each $i > 0$, $C_i$ is a complete cycle for $T$ in $D_i = D_{i-1} \star C_{i-1}$ containing the special node $v_{C_{i-1}}$.
- For each $i \geq 0$, $\mathcal{P}^i = \{P_0^i, P_1^i, \ldots, P_{n_i}^i\}$ is a directed ear decomposition of $H_i/(H_{i-1} \cup C_i)$. Here $H_i = H(C_i)$, that is, $V(H_i) = \bigcup_{j=0}^{i} (\cup_t P_t^j \cup C_j)$.
- $D_{k+1}$ consists of a single node ($H_{k+1} = V$).

Alternatively, we could replace the first condition, by one which does not look for a complete cycle in $D_{i-1} \star C_{i-1}$ but rather for a complete path with distinct endpoints $x, y \in H_{i-1}$ and whose internal nodes lies in $V - H_{i-1}$. An example of a lobe decomposition is shown in Figure 1. We refer to the $C_i$'s as the *lobes* of the decomposition. The *ears* for some particular lobe are the directed paths $P_j^i$ in any directed ear decomposition for the subgraph $H(C_i)$. Each $P_i$ or $C_j$ is called a *segment* of the decomposition. We say that a lobe decomposition $\mathcal{S}$ is *flush* if $C_i$ is a flush for each $i$. We remark that,

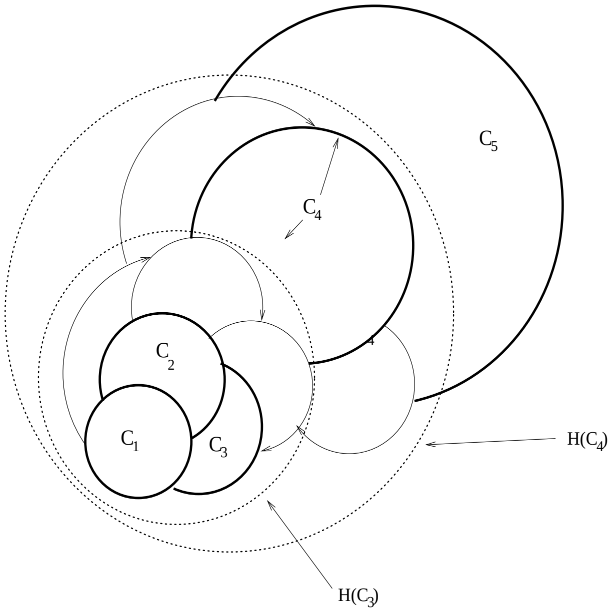


FIGURE 1. A Lobe Decomposition.

whilst each $C_i$ is only a complete cycle in $D_i$ and not necessarily $D$, it can easily be extended to give a complete cycle $C'_i$ in $H_i \subseteq D$ using arcs of $H_{i-1}$ (possibly using arcs in some $C_j, j < i$). Thus, we may also think of $T$ as being generated by a sequence of $C'_0, C'_1, \ldots, C'_k$ of flush cycles in $D$.

This leads to the following decomposition theorem.

**Theorem 2.6** (Lobe Decomposition)**.** *A set of arcs $T$ is a dijoin if and only if it has a lobe decomposition. A dijoin $T$ is minimal if and only if every such decomposition is flush with respect to $T$.*

**Proof.** Suppose that $T = T_0$ is a dijoin in $D = D_0$. Then we may find a complete cycle $C_0$ in $D_0$ by Lemma 2.2. Now set $D_1 = D_0 \star C_0$. By Lemma 2.5, $T_1 = T - A(C_0)$ is again a dijoin in $D_1$. Thus we may find a complete cycle $C_1$ in $D_1$. We may repeat this process on $D_{i+1} = D_i \star C_i$ until some $D_k$ consists of a single node. Hence the $C_i$ give rise to a lobe decomposition.

Conversely, first suppose that $T$ has a lobe decomposition $\mathcal{S} = \{C_0, \mathcal{P}^0, C_1, \mathcal{P}^1, \ldots, C_k, \mathcal{P}^k\}$ and let $\delta(S)$ be a directed cut. Since $D_k$ consists of a single node, the nodes in $S$ and $V - S$ must have been contracted together at some point. Suppose this occurs for the first time in stage $j$. Thus there is some node $x \in S$ and $y \in V - S$ which are 'merged' at this point by contracting the arcs within $C_j$ or one of its ears. Since $\delta(S)$ is a directed cut and since $D_j$ either lies entirely in $S$ or entirely in $V - S$, none of $C_j$'s ears may intersect the cut $\delta^+(S)$. Hence $x$ and $y$ were merged via the complete cycle $C_j$. Hence $T$ intersects $\delta(S)$ and is, therefore, a dijoin.

Now suppose that $T$ is minimal but there is a lobe decomposition for which some $C_i$ is not flush. Let $i$ be the smallest index of such a cycle. Lemma 2.5 implies that $T_i$ is minimal in $D_i$. However, $C_i$ is complete but not flush, contradicting Corollary 2.4 applied to $T_i$. Conversely, suppose that every lobe decomposition is flush. If $T$ is not minimal, then we may find a non-flush complete cycle $C$ by

Corollary 2.4. But then we may start with this cycle $C_0 = C$, and proceed to obtain a non-flush lobe decomposition, a contradiction. $\square$

This theorem immediately implies:

**Theorem 2.7.** *Let $D$ be a digraph with arc weights $w$ whose minimum directed cut is of weight at least 2. If each nontrivial component of the graph induced by the support of $w$ induces a 2-edge-connected undirected graph, then $D$ contains two disjoint dijoins made up of positive weight arcs.*

**Proof.** Let $H_1, H_2, H_3, \ldots, H_t$ be the nontrivial connected components of the subgraph induced by positive weight arcs. For each component, create a lobe decomposition and let $F_i$ be those arcs that are traversed in the forward direction of this decomposition. Now set $F := \cup_i F_i$, and $F' := \cup_i (A(H_i) - F_i)$. We claim that $F, F'$ are the desired dijoins. This is because if $\delta^+(S)$ is a directed cut, then since it has positive $w$-weight, for some $i$, $S \cap V(H_i)$ is a proper, nonempty subset of $V(H_i)$. One easily sees that $F_i$, and $A(H_i) - F_i$ intersect this cut, hence the result. $\square$

This clearly extends to the cardinality case, for which two disjoint dijoins was first observed by Frank (cf. [32]).

**Corollary 2.8.** *If $D$ is a connected digraph with no directed cut of size 1, then it contains two disjoint dijoins.* $\square$

2.2. A Decomposition via Trees.

We now discuss a decomposition based on building a connected subgraph on the underlying undirected graph. We begin with several more definitions. Given our acyclic digraph $D = (V, A)$, we denote by $V^+$ and $V^-$ the set of sources and sinks, respectively. An ordered pair of nodes $(u, v)$ is *legal* if $u$ is a source, $v$ is a sink, and there is a directed path from $u$ to $v$ in $D$. A (not necessarily directed) path is called a *source-sink* path if its start node is a source node $u$ of $D$ and its terminating node $v$ is a sink of $D$. A source-sink path is *legal* if the pair $(u, v)$ is legal.

A *cedar* is a connected (not necessarily spanning) subgraph $\mathcal{K}$ that contains every source and sink, and can be written in the form

$$A(\mathcal{K}) = \bigcup_{P \in \mathcal{P}} A(P)$$

where $\mathcal{P}$ is a collection of legal source-sink paths. We call $\mathcal{P}$ the *decomposition* of the cedar. Given a cedar $\mathcal{K}$, we denote by $F(\mathcal{K})$ the set arcs that are oriented in a forward direction along some path in the source-sink decomposition.

We start with several lemmas.

**Lemma 2.9.** *If $\mathcal{K}$ is a cedar, then $F(\mathcal{K})$ is a dijoin.*

**Proof.** Suppose that $\delta^+(S)$ is a directed cut. Note that since $D$ is acyclic, there must be some source in $S$ and some sink in $V - S$. Since $\mathcal{K}$ is a cedar, it is connected and hence there is a path $P \in \mathcal{P}$ joining a node of $S$ to some node of $V - S$. Let $u$ and $v$ be the source and sink of this path. Since $P$ is legal, it can not be the case that $u \in V - S$ and $v \in S$. It is easy to see that, in each of the remaining three cases, $P$ must have traversed an arc of $\delta^+(S)$ in the forward direction. Thus $F(\mathcal{K})$ does indeed cover the cut. $\square$

Thus, the complete paths induced by a cedar form a dijoin. Now, for each source $v$, we denote by $T_v$ the maximal out-arborescence rooted at $v$ in $D$. Let $\mathcal{R}$ be the subgraph obtained as the union of the $T_v$'s. The following lemma follows trivially from the acyclicity of $D$.

**Lemma 2.10.** *There is a directed path from any node to some sink in $D$. There is also a dipath to any node from some source. In particular, each node lies in some $T_v$.*

**Lemma 2.11.** *The digraph $\mathcal{R}$ is connected.*

**Proof.** Suppose that $S \neq V$ is a connected component of $\mathcal{R}$. By the connectivity of $D$, we may assume that there is an arc $(x, y)$ such that $x \in S$ and $y \notin S$. By Lemma 2.10, there exists a source $v \in \mathcal{R}$ such that $x \in T_v$. The existence of the arc $(x, y)$ then contradicts the maximality of $T_v$. $\square$

Associated with a digraph $D$ is a *derived digraph*, denoted by $D'$. The node set of $D'$ is $V^+ \cup V^-$ and there is an arc $(u, v)$ for each legal pair $(u, v)$ such that $u \in V^+$ and $v \in V^-$.

**Lemma 2.12.** *The derived graph $D'$ is connected.*

**Proof.** Let $S$ be a nonempty proper subset of $V^+ \cup V^-$. It is enough to show that $\delta_{D'}(S)$ is nonempty. By Lemma 2.10, we may assume that both $S$ and $V - S$ contain a source nodes. Let $v_1, \ldots, v_p$ be those sources in $S$, and let $w_1, \ldots, w_q$ be those sources in $V - S$. Evidently, no $T_{v_i}$ contains a sink in $V - S$ and no $T_{w_j}$ contains a sink in $S$. On the other hand, by Lemma 2.11, there exists some $i$ and $j$ for which $T_{v_i}$ and $T_{w_j}$ share a common node, $x$ say. Hence, by Lemma 2.10, there is a dipath from $x$ to some sink node $y$. Therefore, there is a dipath to $y$ from both $v_i$ and $w_j$. It follows that $\delta_{D'}(S) \neq \emptyset$ as required. $\square$

**Lemma 2.13.** *If $T$ is a minimal dijoin, then there exists a cedar $\mathcal{K}$ such that $T = F(\mathcal{K})$.*

**Proof.** Given a minimal dijoin $T$, we construct the desired cedar $\mathcal{K}$. For any legal pair $(u, v)$ in distinct components of $\mathcal{K}$, there is a complete path, $P_{uv}$, from $u$ to $v$. Suppose that such a complete path does not exist. Then there is a subset $S$ such that $u \in S$ and $v \notin S$ with $\delta^-(S) = \emptyset$ and that $\delta^+(S) \cap T = \emptyset$. That is, $S$ defines a directed cut which is not covered by $T$, a contradiction. Now we add the arcs of $P_{uv}$ to the cedar. The desired cedar is then obtained from the final subgraph $\mathcal{K}$ by throwing out any singleton nodes. The resulting digraph is necessarily connected by Lemma 2.12 and, hence, is a cedar. Moreover, by Lemma 2.9, $F(\mathcal{K})$ is a dijoin. Since $F(\mathcal{K}) \subseteq T$, we have that $F(\mathcal{K}) = T$, by the minimality of $T$, $\square$

One simple consequence is the following.

**Corollary 2.14** (Cedar Decomposition)**.** *A set of arcs $T$ is a dijoin if and only if there is a cedar $\mathcal{K}$ with $T \subseteq F(\mathcal{K})$. Moreover, $T$ is minimal if and only if $T = F(\mathcal{K})$ for every cedar $\mathcal{K}$ with $T \subseteq F(\mathcal{K})$.* $\square$

## 3. Finding Minimum Cost Dijoins

In this section, we consider the problem of finding minimum cost dijoins. We begin by presenting a "flushing" operation that can be used to transform one dijoin into another. We then use this operation to characterize when a dijoin is optimal. Finally, we give a simple efficient primal implementation of Frank's algorithm.

### 3.1. Augmentation and Optimality in the Original Topology.

Our approach, in searching for a minimum cost dijoin, is to transform a non-optimal dijoin into an instance of lower cost by augmenting along certain cycle structures. The augmentation operation is motivated by the lobe decomposition in Section 2.1. In due course, we will develop a primal algorithm for the dijoin problem along the lines of that given by Klein [20] for minimum cost flows.

Given a dijoin $T$ and a cycle $C$, let $T' = T \otimes C$ be the resultant graph where $C$ is made into a flush cycle. We call this operation *flushing* the cycle $C$. We may make the resultant flush cycle have either clockwise or anti-clockwise orientation by adjusting whether we flush on $C$ or $\bar{C}$. These two possibilities are shown in Figure 2. Similar operations have been applied in various ways previously to paths instead of cycles (see, for example, [8, 12, 25, 26, 37]). One key difference is that we introduce the reverse of arcs from outside the current dijoin.

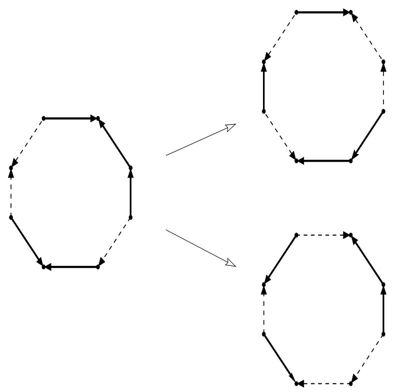

Figure 2. Flushing a Cycle.

We now formalize this operation and introduce a cost structure. Given $T$, we construct an auxiliary digraph, $\mathcal{D}(T)$, as follows. Add each arc $a \in T$ to $\mathcal{D}(T)$ and give it cost 0; also add a reverse copy $\bar{a}$ with cost $-c_a$ (corresponding to the action of removing $a$ from $T$). For each $a \notin T$, add arc $a$ to $\mathcal{D}(T)$ with cost $c_a$ (corresponding to the action of adding $a$ to $T$), and include a reverse copy with cost 0. We call the arcs $T \cup \overline{(A - T)}$ the *zero arcs* (or benign arcs) of the auxiliary digraph. Now for a directed cycle $C$ in $\mathcal{D}(T)$, we define $T \otimes C$ as $T \cup \{a \in C : a \notin T\} - \{a \in T : \bar{a} \in C\}$); we also define $T \otimes A'$ in the obvious fashion for an arbitrary set of arcs $A'$. The value of this operation is illustrated by the following simple lemma.

**Lemma 3.1.** *Let $T$ be a dijoin in $D$, and let $C$ be a cycle of length at least 3 in $\mathcal{D}(T)$. Then $T \otimes C$ is also a dijoin.*

**Proof.** If $S$ induces an out-cut, then if $C$ crosses this cut, then it crosses it at least once in the forward direction, and hence an arc of $T \otimes C$ intersects this cut. If $C$ did not cross this cut, then $T \cap \delta^+(S) = (T \otimes C) \cap \delta^+(S)$. Thus $T \otimes C$ is indeed a dijoin. □

We now see that one may move from one dijoin to another by repeatedly flushing along cycles. To this end, consider a directed graph $\mathcal{G}$ whose nodes correspond to the dijoins of $D$ and for which there is an arc $(T, T')$ whenever there is a cycle $C$, of length at least 3 in $\mathcal{D}(T)$, such that $T' = T \otimes C$. We have the following result.

**Theorem 3.2.** *Given a dijoin $T$ and a minimal dijoin $T^*$ there is a directed path in $\mathcal{G}$ from $T$ to $T^*$. In particular, there is a directed path in $\mathcal{G}$ from a minimal dijoin to any other minimal dijoin.*

**Proof.** Let $T^*$ have a flush lobe decomposition that is generated by the flush cycles $C_0, C_1, \ldots, C_k$. Take $T_0 = T$ and let $T_{i+1} = T_i \otimes C_i$. Note that $T^* \subseteq T_{k+1}$. If $T^* = T_{k+1}$ then we are done. Otherwise take an arc $a \in T_{k+1} - T^*$. Since $T^*$ is a dijoin, the arc $a$ is not critical. So there is a complete cycle $C_a$ for which $a$ is a backward arc. Applying $T_{k+2} = T_{k+1} \otimes C_a$ removes the arc $a$. We may repeat this process for each arc in $T_{k+1} - T^*$. The result follows. □

Observe that there is a cost associated with flushing along the cycle $C$. This cost is precisely $\sum_{a \in C: a \notin T} c_a - \sum_{a \in T: \bar{a} \in C} c_a$. We call a directed cycle $C$, of length at least 3 in $\mathcal{D}(T)$, *augmenting* (with respect to $c$) if it has negative cost. (We note that the general problem of detecting such a negative cycle is NP-hard as is shown in Section 3.1.2.) Clearly, if $C$ is augmenting, then $c(T \otimes C) < c(T)$. Hence if $\mathcal{D}(T)$ contains an augmenting cycle, then $T$ can not be optimal.

### 3.1.1. Auxiliary Networks and Optimality Certificates.

Ideally one could follow the same lines as Klein's negative cycle cancelling algorithm [20] for minimum cost flows. He uses the well-known residual (auxiliary) digraph where for each arc $a$, we include a reverse arc if it has positive flow, and a forward arc if its capacity is not yet saturated. A current network flow is then optimal if and only if there is no negative cost directed cycle in this digraph. This auxiliary digraph is not well enough endowed, however, to provide such optimality certificates for the dijoin problem. Instead Frank introduces his notion of jumping arcs which are added to the auxiliary digraph. In this augmented digraph the absence of negative cost directed cycles does indeed characterize optimality of a dijoin.

We attempt now to characterize optimality of a dijoin working only with the auxiliary digraph $\mathcal{D}(T)$ since it does not contain any jumping arcs. We describe an optimality certificate in this auxiliary digraph. Conceptually this avoids having to compute or visualize jumping arcs; note that $\mathcal{D}(T)$ is trivial to compute since all its arcs are parallel to those of $D$. This comes at a cost, however, in that the new certificate gives rise to a computational task which seems not as simple detecting a negative cycle (at least not without adding the jumping arcs!). This is forewarned by several complexities possessed by the dijoin problem which are absent for network flows. For instance, if a network flow $f$ is obtained from a flow $f'$ by augmenting on some cycle $C$, then we may obtain $f'$ back again, by augmenting the (topologically) same cycle. The same is not true for dijoins. Figure 3 shows gives two examples in which two dijoins can each be obtained from the other in a single cycle flushing. Any such cycles, however, are necessarily topologically distinct.

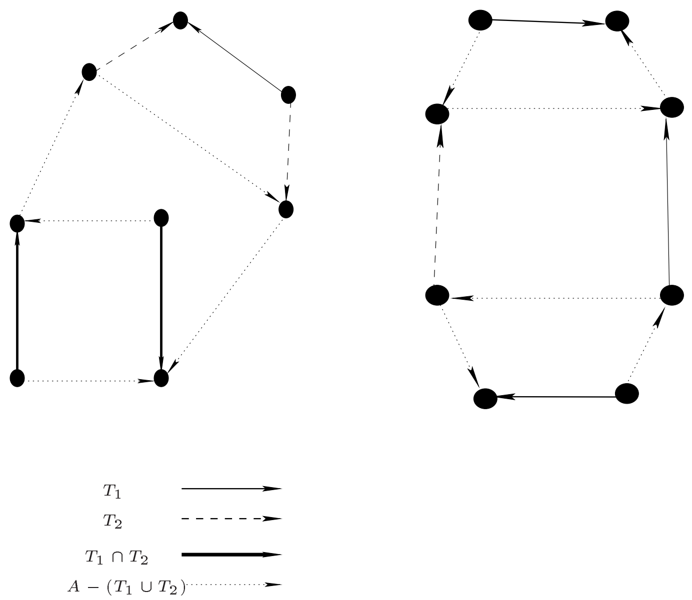


FIGURE 3. The Asymmetry of Flushing.

One might hope that any non-optimal dijoin $T$, could be witnessed by the existence in $\mathcal{D}(T)$ of an augmenting cycle. Unfortunately, Figure 4 shows that this is not the case. The dijoin $T_1$ is non-optimal; there is, however, no augmenting cycle in $\mathcal{D}(T_1)$. We remark that we do not know whether the non-existence of an augmenting cycle guarantees any approximation from an optimal dijoin. (Neither do we know, whether this approach leads to a reasonable heuristic algorithm in practice.)

An alternative optimality condition is suggested by the fact that, whereas any network flow can be decomposed into directed flow cycles, dijoins admit a lobe decomposition. Thus we focus instead on strongly connected subgraphs of the auxiliary graph. We say that a subgraph is *clean* if it contains no digons (directed cycles of length two). Now take a dijoin $T$ and consider a clean strongly connected subgraph $H$ of $\mathcal{D}(T)$. Since $H$ can be written as the union of directed cycles, it follows by Lemma 3.1 that the flushing operation produces produces another dijoin $T' = T \otimes H$. The resulting dijoin has cost $c(T') = c(T) + \sum_{a \in H} c_a$. Consequently, if $H$ has negative cost then we obtain a better dijoin. The absence of a negative cost clean strongly connected subgraph will, in fact, certify optimality. In order to prove this, we need one more definition. Given a clean subgraph $H$ and a directed cycle $C$, we define $H \diamond C$ as follows: firstly, take $H \cup C$ and remove any multiple copies of any arc; secondly, if $a$ and $\bar{a}$ are in $H \cup C$ then keep only the arc from $C$. Our certificate of optimality now follows.

**Theorem 3.3.** *A dijoin $T$ is of minimum cost if and only if $\mathcal{D}(T)$ induces no negative cost clean strongly connected subgraph.*

**Proof.** As we have seen, if $T$ is of minimum cost, then clearly $\mathcal{D}(T)$ contains no negative cost clean strongly connected subgraph. So suppose that $T$ is not a minimum cost dijoin and let $T^*$ be an optimal

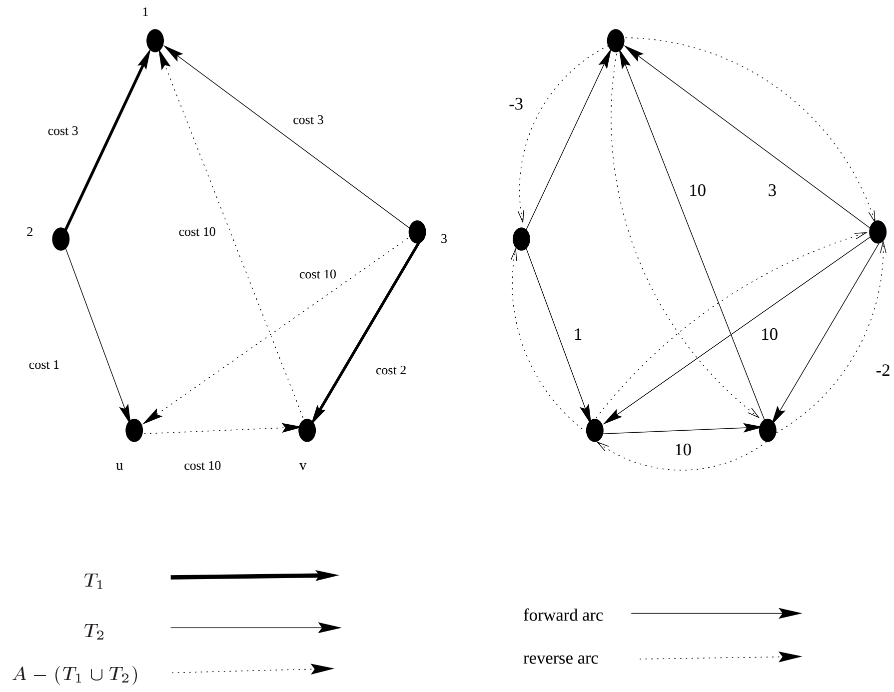


Figure 4. A non-optimal dijoin with no augmenting cycle.

dijoin. Then, by Theorem 3.2, there are cycles $C_1, \ldots, C_t$ such that $T^* = T \otimes C_1 \otimes C_2 \cdots \otimes C_t$. Let $H^1 = C_1$ and $H^{r+1} = H^r \diamond C_{r+1}$. We now show that $H^t$ is a negative cost, strongly connected, clean subgraph. The theorem will then follow as $T^* = T \otimes H^t$.
i) By the use of the operation $\diamond$, no digons may be present in $H^i$ and, hence, $H^i$ is clean.
ii) Since $T^* = T \otimes H^t$, we have that $c(H^t) = c(T^*) - c(T) < 0$. Therefore the cost of $H^t$ is negative.
iii) We prove that $H^i$ is a strongly connected subgraph by induction. $H^1 = C_1$ is strongly connected since $C_1$ is a directed cycle. Assume that $H^{i-1}$ is strongly connected. Then, clearly, $H^{i-1} \cup C_i$ is also strongly connected. Now $H^i$ is just $H^{i-1} \cup C_i$ minus, possibly, the complements of some of the arcs in $C_i$. Suppose, $a = (u, v) \in H^{i-1}$ is the complement of an arc in in $C_i$. Now $a \notin H^i$ but since all the arcs in $C^i$ are in $H^i$ there is still a directed path from $u$ to $v$ in $H^i$. Thus, $H^i$ is a strongly connected subgraph. $\square$

**Corollary 3.4.** *A dijoin $T$ is of minimum cost if and only if $\mathcal{D}(T)$ induces no negative cost strongly connected subgraph that contains no negative cost digon.*

**Proof.** If $T$ is not optimal then by Theorem 3.3 there is a negative cost clean strongly connected subgraph $H$ in $\mathcal{D}(T)$. Clearly, $H$ is a negative cost strongly connected subgraph containing no negative cost digon. Conversely, take a negative cost strongly connected subgraph $H$ that contains no negative cost digon. Suppose that $H$ contains a digon $C = (a, \bar{a})$. This digon has non-negative cost and, therefore $a$ is a non-dijoin arc. If we then flush along $H$ (insisting here that for a digon, the non-digon arc $a$ is flushed after $\bar{a}$) we obtain a dijoin of lower cost. $\square$

These results can be modified slightly so as to insist upon spanning subgraphs. For example we obtain:

**Theorem 3.5.** *A dijoin $T$ is of minimum cost if and only if $\mathcal{D}(T)$ contains a spanning negative cost clean strongly connected subgraph.*

Theorem 3.5 follows directly from Theorem 3.3 and the following lemma.

**Lemma 3.6.** *If $T$ is a minimal dijoin, then the zero arcs induce a strongly connected subgraph.*

**Proof.** Recall that the zero arcs are those arcs in $T \cup \overline{(A-T)}$. Now, consider any pair of nodes $u$ and $v$. If there is no dipath consisting only of zero arcs, then there is a subset $S \subseteq V$ containing $u$ but not $v$ such that $\delta^+(S) \subseteq A - T$ and $\delta^-(S) \subseteq T$. Moreover, $\delta^-(S) \neq \emptyset$ and so contains some arc $a$. One now sees that $a$ cannot be contained on a flush cycle, contradicting Lemma 2.3. □

3.1.2. A Hardness Result.

We have now developed our optimality conditions. In this section, however, we present some bad news. The general problem of finding negative cost clean subgraphs is hard. We consider a digraph $D$ and cost function $c : A \to \mathbb{Q}$. Consider the question of determining whether $D$ contains a negative cost directed cycle whose length is at least three. The corresponding dual problem is to find shortest length clean paths. As we now show, this problem is difficult.

We note that the question of whether there exists any cycle of length greater than 2 is answered in polytime as follows. A *bi-tree* is a directed graph obtained from a tree by replacing each edge by a directed digon. A *long acyclic order* for digraph $D$ is an ordered partition $V_1, V_2, \ldots, V_q$ such that each $V_i$ induces a bi-tree and any arc $(u, v)$ with $u \in V_i$ and $v \in V_j$ with $i \neq j$, satisfies $i < j$. One may prove the following fact

**Proposition 3.7.** *A digraph $D$ has no directed cycle of length at least 3 if and only if it has a long acyclic order.*

We now return to examine the complexity of finding a negative cost such circuit.

**Theorem 3.8.** *Given an digraph $D$ with a cost function $c$. The task of finding shortest path distances is NP-hard, even in the absence of negative cost clean cycles.*

**Proof.** So our directed graph $D$ may contain negative cost digons but no negative cost clean cycles. By a reduction from 3-Satisfiability we show that the problem of finding shortest path distances is NP-hard. Given an instance, $C_1 \wedge C_2 \wedge \cdots \wedge C_n$, of 3-SAT we construct a directed graph $D$ as follows. Associated with each clause $C_j$ is a *clause gadget.* Suppose $C_j = (x \vee y \vee z)$ then the clause gadget consists of 8 disjoint directed paths from a node $s_j$ to a node $t_j$. This is shown in Figure 5.

Each of the paths contains 5 arcs. For each path, the three interior path arcs represent one of the 8 possible true-false assignments for the variables $x, y$ and $z$. The two end arcs associated with each path have a cost 0, except for the path corresponding to the variable assignment $(\bar{x}, \bar{y}, \bar{z})$. Here the final arc has a cost 1. Assume for a moment that the three interior path arcs also have cost 0. Then the 7

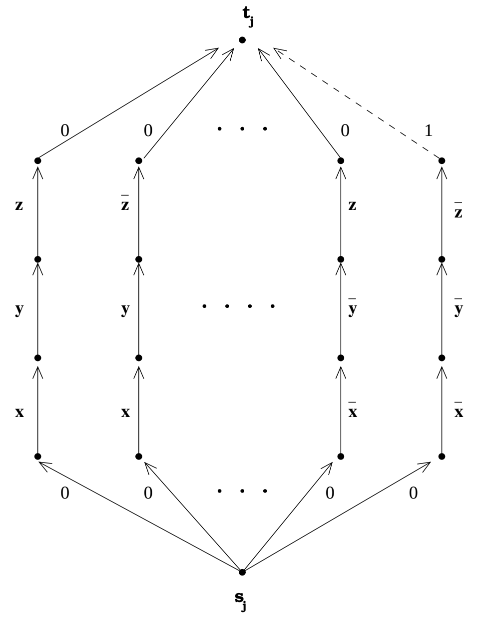


FIGURE 5. The Clause gadget for a $C_j = (x \vee y \vee z)$.

variable assignments that satisfy the clause $C_j$ correspond to paths of cost 0, whilst the non-satisfying assignment corresponds to a path of cost 1.

Now, for $1 \leq j \leq n-1$, we identify the node $t_j$ of clause $C_j$ with the node $s_{j+1}$ of clause $C_{j+1}$. Our goal then is to find a shortest path from $s = s_1$ to $t = t_n$. Such a path will correspond to an assignment of the variables that satisfies the maximum possible number of clauses. We do, though, need to ensure that the variables are assigned consistently throughout the path. This we achieve as follows. Each arc representing a variable will in fact be a structure, called a *variable gadget*.

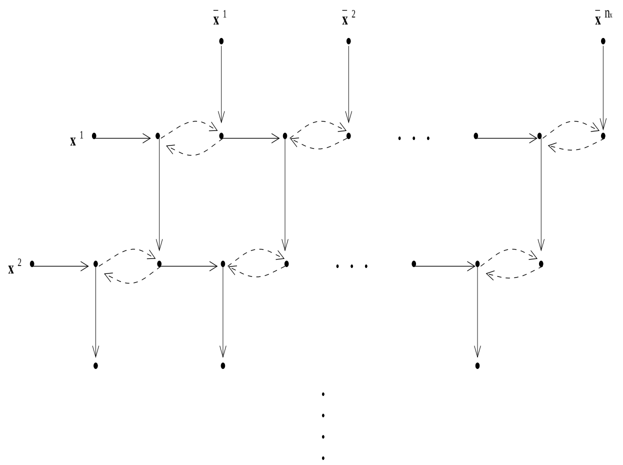


FIGURE 6. Interleaving Variable gadgets.

Note that a variable appears an equal number of times in its unnegated form $x$ and its negated form $\bar{x}$. This is due to the fact that we have four occurrences of $x$ and four of $\bar{x}$ for each clause containing $x$

or $\bar{x}$. Thus if $x$ and $\bar{x}$ appear in a total of $n_x$ clauses we have $4n_x$ occurrences of $x$ and of $\bar{x}$. The variable gadget representing $x$ or $\bar{x}$ will be a directed path consisting of $8n_x$ arcs, with the arcs alternating in cost between $L$ and $-L$. We link together the $4n_x$ directed paths corresponding to the $x$ assignments with the $4n_x$ directed paths corresponding to the $\bar{x}$ assignments as follows. Each of the $4n_x$ negative cost arcs in an $x$ structure forms a digon with a negative cost arc from one of the $\bar{x}$ structures, and vice versa. This is shown in Fig. 6. Here the positive cost arcs are solid whilst the negative cost arcs are dashed. In addition, the $4n_x$ gadgets corresponding to $x$ are labelled $x^1, x^2, \ldots, x^{n_x}$ etc. Hence each pair of variable gadgets $x^i$ and $\bar{x}^j$ meet at a unique negative cost digon.

This completes the description of the directed graph corresponding to our 3-SAT instance. Note that any consistent assignment of the variables corresponds to a unique clean $s-t$ path. This path has the property that it either traverses every arc in a variable gadget or none of the arcs. Note that if a gadget corresponding to $x$ is completely traversed, then none of the gadgets corresponding to $\bar{x}$ may be completely traversed, since our shortest paths must be clean.

We say that a variable gadget is *semi-traversed* if some but not all of the arcs in the gadget are traversed. We do not require any gadget to explicitly enforce consistency within the variables. Instead we show that in an optimal path, none of the variable gadgets may be semi-traversed. By our previous observation, this implies that the path corresponds to a consistent variable assignment. To see that none of the variable gadgets are semi-traversed note that, in any path, the arcs immediately before and after an arc of cost $-L$ must have cost $L$. This follows as we may not use both arcs in a negative cost digon. However, if we have a semi-traversed gadget, then locally, with respect to that gadget, the path must contain two consecutive arcs of cost $L$. As a result the overall cost of the path is at least $L$. For $L > n$, though, clearly there are paths corresponding to consistent variable assignments with smaller cost. The result follows. $\square$

Fortunately, as we will now see, the residual graph (associated with a dijoin) that we consider has enough structure to allow us to find negative cost clean strongly connected subgraphs efficiently.

### 3.2. A Primal Algorithm.

We now present a simple primal implementation of Frank's primal-dual algorithm for the minimum cost dijoin problem. We also show how this algorithm fits into the framework we have just formulated. As discussed in the introduction, Frank's algorithm looks for augmentations of a dijoin in a residual digraph that contains a collection of *jumping arcs* that are not necessarily parallel to any arc of $D$. We begin by defining this residual digraph. To do this we require the concept of a *strict set*. For a dijoin $T$, a directed cut $\delta(S)$ is strict if the number of dijoin arcs in $\delta(S)$ equals the number of weakly connected components in $D-S$. The *minimal strict set* containing a node $v$ is defined as

$$P(v) = \{x : \text{there is no directed cut } \delta^+(S) \text{ such that } x \in S, v \notin S \text{ and } |\delta^+(S) \cap T| = 1\}.$$

¿From these minimal strict sets, we obtain a collection of jumping arcs $\mathcal{J} = \{(u,v) : u \in P(v)\}$. The residual digraph is then $\mathcal{F}(T) = (V, A^F)$, where $A^F = A_r \cup A_f \cup \mathcal{J}$ and $A_r = \{\bar{a} : a \in T\}$ and $A_f = \{a : a \notin T\}$. Frank actually generates dual variables and uses them to select a subgraph of

$(V, A^F)$ to work on[3]. We add the following cost structure to $(V, A^F)$. Each arc $a$ of $A_r$ receives a cost of $-c_{\bar{a}}$, each arc $a \in A_f$ receives a cost $c_a$, and all jumping arcs have cost 0. Given these costs, the following result is then implied by the analysis of Frank's algorithm[4].

**Theorem 3.9** (Optimality Theorem, Frank). *A dijoin $T$ is optimal if and only if $\mathcal{F}(T)$ has no negative cycle.*

Frank does not actually look for negative cycles but rather finds node potentials in the aforementioned subgraph of $\mathcal{F}(T)$. He then either improves the primal dijoin, or updates his dual variables. He shows that this need be done at most $O(n)$ times. The running time of each iteration is dominated by the time to build the jumping arcs, which he does in $O(n^2m)$ time. Gabow showed how to compute these arcs in $O(nm)$ time and, thus, improved the total running time of Frank's algorithm from $O(n^3m)$ to $O(n^2m)$. Gabow's approach is based on so-called centroids of a poset and is applicable to a broader class of problems. His method is very general and also rather complex to describe and implement. In Section 3.2.1 we give an elementary $O(nm)$ algorithm for building the jumping arcs for a given dijoin.

We now describe a simple primal implementation of Frank's Algorithm that builds directly on the Optimality Theorem. We call this the PENDING-ARC algorithm. The algorithm assumes the availability of a subroutine (such as is given in the next section) for computing the jumping arcs of a dijoin.

PENDING-ARC ALGORITHM

In each iteration $i$, we have a current dijoin $T_i$ and subgraph $G_i$ of $\mathcal{F}(T_i)$ that contains no negative cost cycles. The (negative cost) arcs in $\mathcal{F}(T_i) - G_i$ are called the *pending arcs* at time $i$; we denote by $P_i$ this set of arcs. Initially we find any (minimal) dijoin $T_0$, and let $G_0$ consist of the jumping and forward arcs of $\mathcal{F}(T_0)$.

Iteration $i+1$ consists of adding some arc $a$ of $P_i$ to $G_i$. We then look for a negative cycle containing $a$. If none exists, then $G_{i+1} = G_i + a$ has no negative cost cycles and $P_{i+1} = P_i - a$ is a smaller set of pending arcs. Otherwise, we find a negative cost cycle $C$ containing $a$ and we augment on it (dijoin and non-dijoin arcs have their status reversed, jumping arcs are ignored) to obtain $T_{i+1}$. Provided that this cycle is chosen (as detailed below), to be a minimal length cycle of minimum cost, one can show that $G_{i+1} = \mathcal{F}(T_{i+1}) - (P_i - a)$ also has no negative cost cycle. Since $|P_0| = |T_0| \leq n$ and the number of pending arcs decreases by one in each iteration, we have that $P_n = \emptyset$ and so $G_n = \mathcal{F}(T_n)$. Since $G_n$ has no negative cycles, the Optimality Theorem implies that $T_n$ is a minimum cost dijoin.

Note that, in each iteration, we may clearly determine whether $a$ lies in a negative cost cycle in time $O(nm)$ (an $O(m)$ implementation is given below). If one is found, then we must recompute the jumping arcs for the new dijoin $T_{i+1}$. This can also be done in $O(nm)$ time, and hence the total running time of the algorithm is $O(n^2m)$.

---

[3]In fact, Frank works in the digraph where all arcs are reversed.

[4]We remark that phrasing optimality in terms of negative cycles was first done by Fujishige [12] for independent-flows and Zimmerman [37] for submodular flows.

Since one of our aims is to motivate thought on solving dijoin optimization in the original topology, without explicit reference to jumping arcs, we now show how the augmenting structure from Section 3.1.1 is obtained directly from a negative cycle $C$ in $\mathcal{D}(T)$.

**Lemma 3.10.** *The cycle $C$ gives rise to a negative cost clean strongly connected subgraph in $\mathcal{D}(T_i)$.*

**Proof.** The arcs in $C - \mathcal{J}$ are present in $\mathcal{D}(T_i)$ and have a negative total cost. In addition, it follows from Frank that augmenting on $C$ produces a dijoin $T_{i+1}$ of lower cost than $T_i$. Hence, repeating the arguments used in the proof of Theorem 3.3, we see that $T_{i+1} = T \otimes H$, where $H$ is a negative cost clean strongly connected subgraph. Moreover, $H$ is the union of $C - \mathcal{J}$ and a collection of zero arcs, and is therefore easily obtained. □

We now describe a faster implementation of (and fill in some technical details concerning) the PENDING-ARC algorithm. In particular, we show how to compute a suitable cycle $C$ in $O(m)$ time. Therefore, letting $J(n)$ denote the worst case time needed to update the set of jumping arcs in an $n$-node graph after a cycle augmentation is performed, we obtain the following result.

**Theorem 3.11.** *The algorithm* PENDING-ARC *solves the minimum dijoin problem in $O(nm + nJ(n))$ time.*

Theorem 3.11 suggests there may be hope of finding even more efficient methods for making a digraph strongly connected. It clearly levels the blame for inefficiency squarely on the shoulders of computation of the jumping arcs. We discuss the implications of this in the next section. Here, however, we prove Theorem 3.11, that is, correctness of the PENDING-ARC algorithm. This amounts to showing two things. (1) After each iteration, $T_{i+1}$ is again a dijoin and (2) $G_{i+1}$ has no negative cycle. Clearly if we do not augment on a negative cycle, then both these conditions hold. So we assume that we do apply an augmentation. Frank's work indeed shows (see also Lemma 3.13) that if $C$ is chosen as a minimal length minimum cost cycle, then $T_{i+1}$ is again a dijoin. In order to prove (2) we need to more completely describe how to find the cycle $C$.

The simplest way to establish the result, is to maintain shortest path distances (from an arbitrary fixed node $r$) in each $G_i$ as we go along. Let $d_i(x)$, or simply $d(x)$, denote this distance for each node $x$. Let $a = (u, v)$ be the new pending arc to be included. We are looking for a shortest path from $v$ to $u$ in the digraph $G_i$. This can be achieved in $O(m)$ time by allowing Dijkstra's algorithm to revisit some nodes which were already placed in in the shortest path tree (this is described in [1] in a special setting arising from minimum cost flows, but his technique works for any digraph without negative cycles). A more traditional approach, however, is as follows. For each arc $b = (x, y) \in G_i$ let $\hat{c}_b$ denote its cost in this auxiliary digraph, and set $c'_b = d(x) + \hat{c}_b - d(y)$. Thus, this approach does require the use of the distance labels from the previous iteration. Consequently, each $c'_b$ is non-negative since the shortest path values satisfy Bellman's inequalities: $d(y) \leq d(x) + c_{(x,y)}$ for each arc $(x, y)$. In the following, we refer to an arc as *tight*, relative to the $d$-values, if $c'_b = 0$.

Grow a shortest path tree $F$ from $v$ using the costs $c'$. If the shortest path to $u$ is at least the auxiliary cost $|\hat{c}_a|$ of $a$ then there is no negative cost cycle in $G_i \cup \{a\}$. In this case, there may be shorter path from $r$ to $v$ using the arc $(u, v)$. We can then easily update the rest of the distance labels simply by growing an arborescence from $v$. So assume there is a negative cost cycle in $G_i \cup \{a\}$. We obtain such a cycle $C$ in $\mathcal{F}(T_i)$ via the union of $a$ with a $v-u$ path in $F$. Note that since $c' \geq 0$, we may assume without loss of generality that $C$ does not contain any shortcuts via a jumping arc (these have auxiliary cost 0). We now perform an augmentation on $T_i$ along $C$, and let $T_{i+1}$ be the resulting set of arcs. We also set $P_{i+1} = P_i - \{a\}$ and $G_{i+1} = \mathcal{F}(T_{i+1}) - P_{i+1}$.

It remains to show that $G_{i+1}$ has no negative cost cycle. In order to do this we show how to update the shortest path distances $d(x)$. Note that it is sufficient to find new values that (i) satisfy the Bellman inequalities and (ii) for each node $x \neq r$, there is a tight arc entering $x$. In order to establish these facts we need one technical lemma, whose proof is deferred to the end of the section.

**Lemma 3.12.** *If $(x, y)$ is a jumping arc on $C$, then after augmenting $T_i$ along $C$, there is a jumping arc $(y, x)$.*

Let $\delta$ be the length of the path in $F$ from $v$ to $u$. For each integer $i \leq \delta \leq |\hat{c}_a|$, and each node $x \in F$ at distance $i$ from $v$, we set $d(x) = d(x) - (\delta - i)$. In particular, we reduce $d(v)$ by $\delta$. One easily checks that after performing this change, every arc of $C - a$ becomes tight under the new $d(x)$ values. Thus, using Lemma 3.12, the reverse of all of these arcs occurs in $G_{i+1}$ and each of these arcs is tight. One also checks that the Bellman inequalities still hold for all arcs of $G_i - C = G_{i+1} - \bar{C}$. We have thus established (i). Moreover, one checks that the above facts imply that (ii) holds for every node except possibly $u$. In addition, it fails at $u$ only if there was only a single tight arc into $u$ previously and this arc was on $C$.

To correct this, we grow an arborescence $R$ from $u$ as follows. First increase $d(u)$ until $\bar{a}$ becomes tight; that is, $d(u) = d(v) + |\hat{c}_a|$. Now $u$ has a tight arc entering it, but this may have destroyed condition (ii) for some other node if its only tight arc came from $u$ (and, in fact, was the reverse of an arc on $C$!). We scan $u$'s out-neighbours to check for this. If we find such a neighbour $x$, we add it and the tight arc to $R$. We then update $x$'s label; this involves searching $x$'s in-neighbours and, possibly, discovering a new tight arc. We then repeat this process until no further growing is possible. The process terminates in $O(m)$ time provided we do not revisit a node. Such a revisitation would correspond to a directed cycle of tight arcs. This, however, could not be the case, since each of these tight arcs was the unique such arc entering its head and every node originally had a directed path of tight arcs from the source node $r$. Thus after completing this process, we have amended the labels $d(x)$ so that every node satisfies (ii), and every arc satisfies (i). Thus we have new shortest path labels.

We now prove Lemma 3.12 and, thus, Theorem 3.11. To do so, we invoke the following result from the work of Frank.

**Lemma 3.13** (Frank)**.** *Let $S$ induce a directed in-cut in $D$ and let $j$ be the number of jumping arcs in $C \cap \delta^-(S)$. Then $|\delta^-(S) \cap T_i| \geq 1 + j$.* $\square$

One easily checks that this lemma implies that $T_{i+1}$ is again a dijoin. From this we also obtain the following structure on directed cuts that become "strict" after augmentation.

**Lemma 3.14.** *Suppose that $S$ induces a directed in-cut with $|\delta^-(S) \cap T_{i+1}| = 1$. Then either $C$ did not cross the cut induced by $S$, or*

(1) *Every arc of $C \cap \delta^+_{G_i}(S)$ is the reverse of an arc of $T_i$.*
(2) *Every arc of $C \cap \delta^-_{G_i}(S)$ is a jumping arc except possibly one which is an arc of $A - T_i$.*

**Proof.** Since $S$ induces an in-cut, each arc of $C$ in $\delta^+(S)$ is either a jumping arc or the reverse of an arc of $T_i$. Let $j_1$ be the number of jumping arcs, and $t$ be the number of negative cost arcs. Similarly, each arc of $C$ in $\delta^-(S)$ must be either a jumping arc or an arc of $A - T_i$. Let $j$ be the number of jumping arcs and $k$ be the number of arcs of $A - T_i$. Note that $|\delta^-(S) \cap T_{i+1}| \geq k$ and hence $k \in \{0, 1\}$. Note next that $j_1 + t = j + k$. Moreover, by the previous lemma we have that $|\delta^-(S) \cap T_i| \geq 1 + j$. Thus $1 = |\delta^-(S) \cap T_{i+1}| \geq j + 1 - t + k$, and so $j + k - t \leq 0$ which implies $j_1 = 0$. This completes the proof. □

Using this, our desired result follows.

**Proof of Lemma 3.12.** Suppose that $(y, x)$ is not a jumping arc after $T_i$ is augmented along $C$. Then there is a directed cut induced by a set $S$ such that $x \in S$, $y \notin S$ and $|\delta^-(S) \cap T_{i+1}| = 1$. But then by Lemma 3.14 every arc of $\delta^+(S) \cap C$ must be the reverse of a dijoin arc, contradicting the fact that $(x, y)$ lies in this set. □

3.2.1. Computing the Jumping Arcs.

The bottleneck operation in obtaining a minimum cost dijoin is obtaining the set $\mathcal{J}$ of jumping arcs. In order to find the jumping arcs we have to find the minimal strict set, denoted $P(v)$, with respect to each node $v$. Frank [8] showed how to construct all the minimal strict sets in $O(n^3m)$ time, and also proved that this need be done at most $n$ times, giving a total running time of $O(n^2m)$. He then asserted that "it would be useful to have a procedure with running time $O(n^3)$ (or perhaps $O(n^2)$)".

Gabow [14] described such a procedure that runs in $O(nm)$, improving the running time of Frank's algorithm to $O(n^2m)$. Underlying Gabow's result is the following simple observation. Consider the digraph obtained from $D$ by replacing each arc in $D$ by two parallel arcs and adding the complement of every dijoin arc. Then a node $u$ is in $P(v)$ if and only if there are two arc-disjoint paths from $v$ to $u$ in this digraph. Gabow developed a general method which builds efficient representations of arbitrary posets of strict sets. This is achieved by repeatedly finding a centroid (like a separator) of the poset. He then uses the final representation to answer 2-arc connection queries and, thus, find the jumping arcs.

Here we describe an elementary $O(nm)$ algorithm, based upon arborescence growing, for computing the minimal strict sets. We also discuss the hardness of improving upon this. We use the terminology that, for a dijoin arc $a$, an out-cut $\delta^+(S)$ is *$a$-separating (for $v$)* if $a$ is the unique arc of $T$ in $\delta^+(S)$ and $v \notin S$. We refer to the set $S$ as being *$a$-separating (for $v$)* as well. Observe that any node that is not in $P(v)$ is contained in (the shore of) a separating cut for $v$. If no such cut exists then the node is in $P(v)$. This motivates the following question: given a dijoin arc $a$ and a node $v$, find the $a$-separating cut for $v$ (if it exists) which is induced by a maximal set $S_a(v)$. We use the notation $S_a$ instead of $S_a(v)$ if the context is clear. We call such a question a *red query*, and using such queries we can compute each $P(v)$

in $O(m)$ time. To do this we grow an arborescence, called the *jumping tree*, rooted at $v$ in $D \cup \bar{D}$. As we proceed, the nodes are coloured blue and red depending upon the outcome of the query. At the end of the algorithm, the minimal strict cut $P(v)$ consists exactly of those nodes coloured blue.

Growth of the jumping tree $\mathcal{T}$ alternates between blue phases and red phases. A *blue phase* consists of repeatedly finding a blue node $u$ in the tree and a node $w$ not yet in the tree, for which there is an arc $(u, w) \in D$. We then add the arc to the tree and label $v$ as blue. We repeat this until no such arcs exist. At this point the nodes of $\mathcal{T}$ induces an in-cut. We then choose a dijoin arc $(u, w)$ in this in-cut, and attempt to determine the colour of $u$. More precisely, we make a *red query* with respect to $a$ and $u$, the result of which is an arborescence $S_a$ rooted at $u$. If $S_a$ is empty, in which case there are no $a$-separating cuts for $v$, we colour $u$ blue. Otherwise, $S_a$ identifies a maximal $a$-separating cut, and all nodes in this set are coloured be red. The algorithm is formally described below. Here $\Gamma^+(S)$ denotes the set of out-neighbours of $S$, that is, those nodes $y \notin S$ such that there is an arc $(x, y)$.

JUMPING-TREE

```
Colour v blue; set T = ({v}, ∅)
While V(T) ≠ V
    If there is an arc (u,w) ∈ A(D) such that u ∈ T, w ∉ T
        Colour w blue; add w and (u,w) to T
    Otherwise there exists a = (u,w) ∈ T such that u ∉ T, w ∈ T
        Add u, ā to T and let S_a =REDQUERY(a)
        If V(S_a) = ∅ then colour u blue
        Else colour all nodes of S_a red and add S_a to T
            Colour all nodes of Γ+(S_a) − V(T) blue
EndWhile
```

**Theorem 3.15.** *The algorithm Jumping-tree correctly calculates $P(v)$.*

**Proof.** Notice that a node $u$ can only be coloured red if there is a dijoin arc $a$ for which there is a justifying cut $\delta^+(S)$ with $u \in S$ and $v \in V - S$. Since this is a separating cut, such a node is not contained in $P(v)$ and is, therefore, correctly labelled. Next we show that any blue nodes is in $P(v)$. This we achieve using induction beginning with the observation that the blue node $v$ is correctly labelled. Now a node $w$ may be coloured blue for one of two reasons. First, there is a blue node $u$ and an arc $(u, w)$. Now, if $u$ is correctly labelled then $w$ must be as well. Suppose the converse, and let $\delta^+(S)$ be a separating cut with respect to $v$ that contains $w$ in its shore. Clearly $u$ must then also be in this shore, a contradiction. The node $u$ can also be coloured blue if there is a dijoin arc $a$ inducing a maximal separating cut $\delta^+(S)$ with respect to $v$ contains a non-dijoin arc $(y, u)$. Suppose that there is a dijoin arc $a'$ inducing a maximal separating cut $\delta^+(S')$ with respect to $v$ whose shore contains $u$. Since there is an arc $(y, u)$, we have $y \in S'$ and therefore $S$ and $S'$ intersect. It follows that $S \subseteq S'$, otherwise $\delta^+(S')$ is not a maximal $a'$-separating cut for $v$. This implies that the head of arc $a$, say $z$ is in $S'$. We obtain a contradiction since $z$ is coloured blue. $\square$

**Lemma 3.16.** *The algorithm* JUMPING-TREE *can be used to find the jumping arcs in $O(nm)$ time.*

**Proof.** The algorithm JUMPING-TREE evidently runs in $O(m)$ time if the red queries can be answered in say $O(|S_a|)$-time. To achieve this, we implement an $O(nm)$ time preprocessing phase. In particular, for each dijoin arc $a = (u, w) \in T$, we spend $O(m)$ time to build a structure that allows us later to find an arbitrary set $S_a$ in time $O(|S_a|)$.

Our method is as follows. Let $D_a = D \cup (\bar{T} - \bar{a})$. Initially, we call node $u$ *alive*. We then repeatedly grow, in $D_a$, maximal in-arborescences $A_x$ from any existing alive node $x$. In building such arborescences we may spawn new alive nodes. As we proceed, we let $X$ be the set of *visited* nodes. We add a new node to some $A_x$ only if it is not already in $X$. If it is added, then it is marked as visited and put in $X$. In addition, if this node had been labelled alive, then this label is now removed. Each node $z$ in $A_x$ is also marked by an $(x)$ so we know that it was node $x$'s tree that first visited $z$.

After $A_x$ is completed, observe that $X$ induces an strict cut containing $a$ as its only dijoin arc. All out-neighbours of $X$, except $w$, are then marked as alive, i.e., put on a stack of candidates for growing future trees. Upon completion, we have generated a poset structure for the $a$-strict sets. We use this structure to create an acyclic graph $H$ which has a node $y$ for each arborescence $A_y$ which was grown. There is an arc $(x, y) \in H$ if there is an arc with tail in $A_x$ and head in $A_y$. In other words, $H$ is obtained by contracting each $A_x$ to a single node. This graph is easily built as the algorithm runs.

Given this preprocessing, we now show how to find the maximal $a$-separating sets for $v$. Consider a partial arborescence $\mathcal{T}$ created during the the course of the jumping tree algorithm for $v$. Observe that if some node $y \in H$ lies in $S_a$, then $A_y \subseteq V(S_a)$ by construction of $A_y$. Thus, our task amounts to determining which nodes of $H$ lie in $S_a$. In addition, observe that if some node $y \in H$ lies in $V - \mathcal{T}$, then either (i) $v \in A_y$ or (ii) $A_y \subseteq V - V(\mathcal{T})$. To see this, suppose that $v \notin A_y$ and that $A_y \cap \mathcal{T} \neq \emptyset$. Then $\mathcal{T}$ must have grown along some arc into $A_y \cap \mathcal{T}$. However, no such arc exists in $D \cup \overline{T - a}$. In particular, this also shows that $S_a \subseteq V - \mathcal{T}$. It then follows that a node $y \in H$ lies in $S_a$ if and only if $y \notin \mathcal{T}$.

Consequently, we may build $S_a$ by starting with the node of $H$ which contains the tail of $a$ and growing a maximal arborescence in $H - \mathcal{T}$. Then $S_a$ is the union of those $A_y$'s which were visited in this process. Given $H$ and the $A_y$'s, this can be done in $O(|S_a|)$ time, as required. □

We remark that Gabow also showed how fast matrix multiplication can be used to find all the strict minimal cuts in $O(n^{MM(n)})$ time (here, $MM(n)$ denotes the time to do matrix multiplication). Our algorithm may also be implemented in this time. We end this section by commenting on the hardness of finding a more efficient algorithm for calculating the $P(v)$. In particular, we show that finding all the minimal strict sets is as hard as boolean matrix multiplication. To achieve this we show that *transitive closure* in acyclic graphs is a special case of the minimal strict set problem.

**Lemma 3.17.** *The minimal strict set problem is as hard as transitive closure.*

**Proof.** Given an acyclic graph $D$ we form a new graph $D'$ as follows. Add two new nodes $s$ and $t$ with an arc $(s, t)$. In addition, add an arc from $s$ to every source in $D$ and add an arc to $t$ from every sink in $D$. Clearly, the arc $(s, t)$ is itself a dijoin $T$ in $D'$. Now observe that, for each node $v$ in $D$, there is a correspondence between the reachability sets for $v$ in $D$ and the minimal strict set, with respect to $T$, containing $v$ in $D'$. The result follows. $\square$

To see that transitive closure is as hard as boolean matrix multiplication. Take two matrices $A$ and $B$ and consider the acyclic graph defined by the adjacency matrix

$$M = \begin{pmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}$$

Now the transitive closure of $M$ is then:

$$Cl(M) = \begin{pmatrix} I & A & AB \\ 0 & I & B \\ 0 & 0 & I \end{pmatrix}$$

Therefore, trying to speed up the minimum cost dijoin algorithm by finding more efficient methods to calculate the minimal strict cuts may be difficult. One possible way around this would be to avoid calculating the minimal strict cuts from scratch at each iteration. Instead, it may well be possible to more efficiently update the minimal strict cuts after each augmentation.

## 4. Packing Dijoins

As mentioned in the introduction, an important question regarding dijoins is the extent to which they pack. Here, we discuss this topic further. We consider a digraph $D$ with a non-negative integer vector $u$ of arc weights, and denote by $\omega_D(u)$ the minimum weight of a directed cut in $D$. An initial question of interest is: determine the existence of a constant $k_0$ such that every weighted digraph $D$, with $\omega_D(u) \geq k_0$, admits a pair of disjoint dijoins contained in the support of $u$. (The unweighted case was discussed in Section 2.) We approach this by considering submodular network design problems with associated orientation constraints (as were recently studied in [9] and [19]). First, however, we look at the problem of fractionally packing dijoins.

### 4.1. Half-Integral Packing of Dijoins.

By blocking theory and the Lucchesi-Younger Theorem (see Section 1.1), any digraph $D$ with arc weighting vector $u$ has a fractional $u$-packing of dijoins (a packing such that each arc $a$ is in at most $u_a$ of the dijoins) of size $\omega_D(u)$. We show now that there is always a large $\frac{1}{2}$-integral packing of dijoins.

**Theorem 4.1.** *For any digraph $D$ and non-negative arc vector $u$, there is a half-integral $u$-packing of dijoins of size $\frac{\omega_D(u)}{2}$.*

**Proof.** Let $k = \omega_D(u) \geq 2$. We now fix a node $v$ and consider a partition of the set of directed cuts $\delta^+(S)$ into two sets. Following a well-known trick, let $\mathcal{O} = \{S : \delta^-(S) = \emptyset, v \in S\}$ and $\mathcal{I} = \{S : \delta^+(S) = \emptyset, v \in S\}$. Next note that an arc vector $x$ identifies a (fractional) dijoin if and only if $x(\delta^+(S)) \geq 1$ for each $S \in \mathcal{O}$ and $x(\delta^-(S)) \geq 1$ for each $S \in \mathcal{I}$.

Consider now the digraph $H$ obtained from $D$ by deleting any zero-weight arcs and adding infinitely many copies of each reverse arc. It follows that for each proper $S \subseteq V$ containing $v$ we have $|\delta^+_H(S)| \geq k$.

Edmonds Branching Theorem [6] then implies that $H$ contains $k$ disjoint spanning out-arborescences rooted at $v$. Call these arborescences $O_1, O_2, \ldots, O_k$, and for each $i$, let $x^i$ be the $0,1$ incidence vector in $\mathbb{R}^A$ of the set of forward arcs $O_i \cap A$. Obviously for each arc $a$, we have $\sum_i x^i_a \leq 1$. Similarly, there are $k$ disjoint spanning in-arborescences rooted at $v$ and an associated sequence $y^i$ of arc vectors. We thus have, for each $i$ and $j$, that $x^i(\delta^+(S)) \geq 1$ for each $S \in \mathcal{O}$ and $y^j(\delta^+(S)) \geq 1$ for each $S \in \mathcal{I}$. Hence, the support of the integral vector $x^i + y^j$ identifies a dijoin $T_{i,j}$ for each $i, j$ pair. Since any arc is in at most two of the dijoins $T_{1,1}, T_{2,2}, \ldots, T_{k,k}$, a $\frac{1}{2}$-integral dijoin-packing of size $\frac{k}{2}$ is obtained by giving each such dijoin weight one half. □

4.2. WEAK SUPERMODULARITY AND PACKING DIJOINS.

We begin this section by recalling some definitions. Two sets $A$ and $B$ are *intersecting* if each of $A - B, B - A, A \cap B$ are non-empty; the sets are *crossing* if, in addition, $V - (A \cup B)$ is non-empty. A family of sets $\mathcal{F}$ is a *crossing family* (respectively, *intersecting family*) if for any pair of crossing (respectively, intersecting) sets $A, B \in \mathcal{F}$ we have $A \cap B, A \cup B \in \mathcal{F}$. The submodular flow polyhedra of Edmonds and Giles are given in terms of set functions defined on such crossing families of sets. We consider a larger family of set functions based on a notion of weak submodularity, a concept introduced for undirected graphs in [34]. A set family $\mathcal{F}$ is *weakly crossing* if for each intersecting pair $A$ and $B$ either $A \cap B, A \cup B \in \mathcal{F}$ or $A - B, B - A \in \mathcal{F}$. A real-valued function $f$ defined on $\mathcal{F}$ is *weakly supermodular* if for each intersecting pair $A, B \in \mathcal{F}$ one of the following holds:

(i) $A \cap B$, $A \cup B \in \mathcal{F}$ and $f(A) + f(B) \leq f(A \cap B) + f(A \cup B)$.

(ii) $A - B$, $B - A \in \mathcal{F}$ and $f(A) + f(B) \leq f(A - B) + f(B - A)$.

We claim that for an arbitrary digraph $D$, the family $\mathcal{F}^{\pm}(D) = \{S : \delta^+(S) = \emptyset \text{ or } \delta^-(S) = \emptyset\}$ is weakly crossing. Indeed, consider intersecting members $A$ and $B$ of $\mathcal{F}^{\pm}$. Suppose first that both $A, B$ induce out-cuts (or in-cuts). If $A$ and $B$ are crossing then $A \cap B, A \cup B$ are also out-cuts (respectively in-cuts). If $A$ and $B$ are not crossing then both $A - B$ and $B - A$ induce in-cuts (respectively out-cuts). Finally, suppose that $A$ induces an out-cut and $B$ induces an in-cut. Then $A - B$ is an out-cut and $B - A$ is an in-cut. Thus the claim is verified.

We are interested in *weakly supermodular network design problems* with *orientation constraints*. That is, we have a digraph $D = (V, A)$ and a weakly supermodular function $f$ defined on a weakly crossing family $\mathcal{F}$ of subsets of $V$. We are also given a partition of arcs into pairs $a$ and $\bar{a}$, and for each such pair there is a capacity $u_a$. We then define $\mathcal{P}_D(f, u)$ to be the polyhedron of all non-negative vectors $x \in \mathbb{Q}^A$ such that $x(\delta^+(S)) \geq f(S)$ for each $S \in \mathcal{F}$ and $x_a + x_{\bar{a}} \leq u_a$ for each arc pair $a$ and $\bar{a}$. In general, the region $\mathcal{P}_D(f, u)$ need not be integral and we are interested in finding minimum cost vectors in its integer hull. There are a number of related results in this direction. Notably, Melkonian and Tardos [27] show that if $f$ is crossing supermodular and if the orientation constraints are dropped, then each extreme point of this polyhedron contains a component of value at least $\frac{1}{4}$. Khanna, Naor and Shepherd [19] describe a 4-approximation algorithm for the case with orientation constraints provided and $f(S) = 1$ for every proper subset $S$. They also show that the polyhedron is integral in the case that $f$ is intersecting supermodular (see [9] for generalizations of this latter result).

In terms of packing dijoins, we are interested in the polyhedron $\mathcal{P}_H(f,u)$ where $H = D \cup \bar{D}$ and $f(S) = 1$ for each $S \in \mathcal{F}^{\pm}$. Observe that, for any digraph $D$ and weighting $u$ with $\omega_D(u) \geq 2$, we have $\mathcal{P}_H(f,u)$ is non-empty since we may assign $\frac{1}{2}$ to each arc variable. For now, we may as well assume $u$ is a $0,1$ vector. Suppose that $\mathcal{P}_H(f,u)$ has an integral solution $x$. Let $F$ be those arcs $a \in D$ such that $x_a = 1$, and let $K = A - F$. We claim that the arc sets of $D$ associated with both $F$ and $K$ are dijoins. For suppose that $S$ induces an out-cut. Then the constraint $x(\delta^+(S)) \geq 1$ implies that $F$ contains an arc from this cut, whereas the constraint $x(\delta^+(V-S)) \geq 1$ implies that $K$ intersects this cut[5]. The example of Schrijver [29] implies that there is a weighted digraph with $\omega_D(u) = 2$ for which $\mathcal{P}_H(f,u)$ has no integral point (since $D$ does not have a pair of disjoint dijoins amongst the support of $u$). We ask:

**Conjecture 4.2.** *Is there a constant $k_0$ such that, for every weighted digraph $D$, if $\omega_D(u) \geq k_0$ then $\mathcal{P}_H(f,u)$ contains an integral point? Is this true for $k_0 = 4$?*

We note that, if $\omega_D(u) \geq 4k$ were to imply that $\mathcal{P}_H(kf,u)$ contains an integral point, then the methods used at the beginning of this section can be made to yield an $O(k)$ packing of dijoins.

## References


[1] R. Bhandari, "Optimal diverse routing in telecommunication fiber networks", *Proceedings of IEEE Infocom*, June 1994, 1498–1508.

[2] V. Chvátal, On certain polytopes associated with graphs. *J. Combin. Theory Ser. B* **18** (1975) 138-154.

[3] W. Cook, W. Cunningham, W. Pulleyblank and A. Schrijver, *Combinatorial Optimization*, Wiley-Interscience, New York, 1998.

[4] W. Cui and S. Fujishige, "A primal algorithm for the submodular flow problem with minimum-mean cycle selection", *J. Operation Research Society of Japan*, **31**, pp431-440, 1998.

[5] W. Cunningham and A. Frank, "A primal-dual algorithm for submodular flows", *Mathematics of Operations Research*, **10(2)**, pp251-261, 1985.

[6] J. Edmonds, "Edge-disjoint branchings", in *Combinatorial Algorithms*, (R. Rustin, ed.) Alg. Press, New York, pp91-86, 1973.

[7] J. Edmonds and R. Giles, "A min-max relation for submodular functions on graphs", *Annals of Discrete Math.*, **1**, pp185-204, 1977.

[8] A. Frank, "How to make a Digraph Strongly Connected", *Combinatorica*, **1**, p145-153, 1981.

[9] A. Frank, T. Király and Z. Király, "On the orientation of graphs and hypergraphs", *Technical Report TR-2001-06 of the Egerváry Research Group*, Budapest, Hungary, 2001.

[10] P. Feofiloff and D. Younger, "Directed cut transversal packing for source-sink connected graphs", *Combinatorica*, **7**, pp255-263, 1987.

[11] L. Fleischer and S. Iwata, "Improved algorithms for submodular function minimization and submodular flow", *STOC 2000*, pp107-116, 2000.

[12] S. Fujishige, "Algorithms for solving the independent flow problem", *J. Operation Research Society of Japan*, **21(2)**, 1978.

[13] S. Fujishige, "Submodular functions and optimization", *Annals of Discrete Math.*, **47**, Monograph, North Holland Press, Amsterdam, 1991.


[5] We remark, that we could have also cast this as a "weak" submodular flow model as well; this could be achieved by defining $f(S) = |\delta(S)| - 1$ and then requiring $x(\delta^+(S)) - x(\delta^-(S)) \leq f(S)$ for each $S \in \mathcal{F}$.

[14] H. Gabow, "Centroids, representations, and submodular flows", *J. Algorithms*, **18**, pp586-628, 1995.

[15] S. Iwata, L. Fleischer and S. Fujishige, "A combinatorial strongly polynomial-time algorithm for minimizing submodular functions", to appear in *J. ACM.*

[16] S. Iwata, S. McCormick and M. Shigeno, "Fast cycle canceling algorithms for minimum cost submodular flow", August 1999.

[17] K. Jain, "A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem". *Combinatorica*, **21(1)**, pp39-60, 2001.

[18] A. V. Karzanov, "On the minimal number of arcs of a digraph meeting all its directed cutsets", Abstract in *Graph Theory Newsletter*, **8**, 1979.

[19] S. Khanna, S. Naor and F. B. Shepherd, "Directed network design problems with orientation constraints", *SODA '99*, San Francisco, January 1999.

[20] M. Klein, "A primal method for minimal cost flows", *Management Sci.*, **14**, pp205-220, 1967.

[21] O. Lee and Y. Wakabayashi, "A note on a min-max conjecture of Woodall", to appear in *Journal of Graph Theory.*

[22] A. Lehman, Width-length inequality and degenerate projective planes, *Polyhedral Combinatorics, Proceedings of the DIMACS Workshop, Morristown, New Jersey, June 1989*, Eds. P.D. Seymour, W. Cook, American Mathematical Society, (1990), 101-106.

[23] L. Lovász, "On two minmax theorems in graphs", *J. Combinatorial Theory (B)*, **21**, pp96-103, 1976.

[24] C. Lucchesi, "A minimax equality for directed graphs", Ph.D. thesis, *University of Waterloo*, Waterloo, Ontario, 1976.

[25] C. Lucchesi and D. Younger, "A minimax theorem for directed graphs", *J. London Math. Society (2)*, **17**, pp369-374, 1978.

[26] I. McWhirter and D. Younger, "Strong covering of a bipartite graph", *J. London Math. Society (2)*, **3**, pp86-90, 1971.

[27] V. Melkonian and É. Tardos, "Approximation algorithms for a directed network design problem", *IPCO '99.*

[28] W. R. Pulleyblank, Personal communication, 1994.

[29] A. Schrijver, "A counterexample to a conjecture of Edmonds and Giles", *Discrete Math.* **32**, pp213-214, 1980.

[30] A. Schrijver, "Min-max relations for directed graphs", *Annals of Discrete Math.*, **16**, 261-280, 1982.

[31] A. Schrijver, "A combinatorial algorithm minimizing submodular functions in strongly polynomial time", *J. Combinatorial Theory (B)*, **80**, pp346-355, 2000.

[32] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, *Springer Verlag*, 2002.

[33] C. Wallacher and U. Zimmermann, "A polynomial cycle canceling algorithm for submodular flows", *Math. Programming (Series A)*, **86(1)**, pp1-15, 1999.

[34] D. Williamson, M. Goemans, M. Mihail and V. Vazirani, "A primal-dual approximation algorithm for generalized steiner network problems", *Combinatorica*, **15**, pp435-454, 1995.

[35] D. Younger, "Feedback in a directed graph", Ph.D. thesis, *Columbia University*, New York, 1963.

[36] D. Younger, "Maximum families of disjoint directed cuts", *Recent Progress in Combinatorics*, pp329-333, Academic Press, New York, 1969.

[37] U. Zimmermann, "Minimization on submodular flows", *Discrete Appl. Math.*, **4**, pp303-323, 1982.