MATHEMATICAL ENGINEERING TECHNICAL REPORTS

An Algorithm for Minimum Cost Arc-Connectivity Orientations

Satoru IWATA and Yusuke KOBAYASHI

METR 2005–16 July 2005

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

The METR technical reports are published as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

An Algorithm for Minimum Cost Arc-Connectivity Orientations

Satoru IWATA and Yusuke KOBAYASHI

Department of Mathematical Informatics
Graduate School of Information Science and Technology
University of Tokyo
{iwata, Yusuke_Kobayashi}@mist.i.u-tokyo.ac.jp

July, 2005

Abstract

Given a $2k$-edge-connected undirected graph, we consider to find a minimum cost orientation that yields a k-arc-connected directed graph. This minimum cost k-arc-connected orientation problem is a special case of the submodular flow problem. Frank (1982) devised a combinatorial algorithm that solves the problem in $\mathrm{O}(k^2n^3m)$ time, where n and m are the numbers of vertices and edges, respectively. Gabow (1995) improved Frank's algorithm to run in $\mathrm{O}(kn^2m)$ time by introducing a new sophisticated data structure. We describe an algorithm that runs in $\mathrm{O}(k^3n^3 + kn^2m)$ time without using sophisticated data structures. In addition, we present an application of the algorithm to find a shortest dijoin in $\mathrm{O}(n^2m)$ time, which matches the current best bound.

Key Words. Arc-connectivity, Graph orientation, Submodular flow problem, Crossing family, Dijoin.

1 Introduction

Given an undirected graph $\overline{G} = (V, E)$, we consider to orient each edge in either direction to obtain a k-arc-connected directed graph $G = (V, A)$. We say a directed graph $G = (V, A)$ is k-arc-connected if there are at least k arcs from X to $V \setminus X$ for any proper nonempty vertex subset X. We also say an undirected graph $\overline{G} = (V, E)$ is k-edge-connected if there are at least k edges connecting X and $V \setminus X$ for any proper nonempty vertex subset X. The following theorem clarifies when we are able to obtain a k-arc-connected directed graph.

Theorem 1 (Nash-Williams [12]). *An undirected graph $\overline{G} = (V, E)$ has a k-arc-connected orientation if and only if it is $2k$-edge-connected.*

This theorem naturally gives rise to the following optimization problem.

Problem. Given a $2k$-edge-connected undirected graph $\overline{G} = (V, E)$ and costs of orientations in both directions for each edge, find a minimum cost orientation that results in a k-arc-connected directed graph.

This problem is referred to as the *minimum cost k-arc-connected orientation problem.* In particular, if $k = 1$, it is called the *minimum cost strongly connected orientation problem.* The minimum cost k-arc-connected orientation problem is a special case of the submodular flow problem, for which strongly polynomial algorithms are known. Exploiting characteristic properties of the specific problems, however, we should be able to construct a more efficient algorithm.

We will consider graphs with n vertices and m edges. Frank [5] reduced the minimum cost k-arc-connected orientation problem to the submodular flow problem and constructed an algorithm that runs in $\mathrm{O}(k^2n^3m)$ time. Gabow [9] improved Frank's algorithm to run in $\mathrm{O}(kn^2m)$ time by introducing a new data structure called a *centroid tree.* We present an algorithm that runs in $\mathrm{O}(k^3n^3 + kn^2m)$ time without using such a sophisticated data structure. This simpler algorithm runs as fast as Gabow's when $k^2n = \mathrm{O}(m)$. These algorithms are shown in Table 1, where $M(n)$ is the time to multiply two $n \times n$ matrices. The current best known bound is $M(n) = \mathrm{O}(n^{2.38})$ [1].

Table 1: Algorithms for the minimum cost k-arc-connected orientation problem.

	Time	Space
Frank [5]	$\mathrm{O}(k^2n^3m)$	$\mathrm{O}(n^2)$
Gabow [9]	$\mathrm{O}(kn^2m)$	$\mathrm{O}(m)$
	$\mathrm{O}(knM(n))$	$\mathrm{O}(n^2)$
This paper	$\mathrm{O}(k^3n^3 + kn^2m)$	$\mathrm{O}(n^2)$

The key idea of our algorithm is an extensive use of packing arborescences. In a directed graph $G = (V, A)$, we say that $F \subseteq A$ is an *r-arborescence* for $r \in V$ if F is a spanning tree when we ignore the direction of arcs, no arc of F enters r, and exactly one arc of F enters v for each $v \in V \setminus \{r\}$. An *r-cut* is the set of arcs from X to $V \setminus X$ for a proper vertex subset X containing r. The following theorem provides a min-max characterization of packing arborescences.

Theorem 2 (Edmonds [2])**.** *The maximum number of arc-disjoint r-arborescences equals the minimum number of arcs in an r-cut.*

As a consequence of this theorem, if G is a k-arc-connected directed graph, there are k arc-disjoint r-arborescences for any vertex r of G. In Section 4, we use this fact and an efficient algorithm for packing arborescences due to Gabow [8].

Given a directed graph $G = (V, A)$, we say an arc set F is a *dijoin* if G becomes strongly connected by contracting F. Given a length for each arc, we consider to find a dijoin of minimum total length. This problem is called the *shortest dijoin problem.*

Table 2: Algorithms for the shortest dijoin problem.

	Time	Space
Frank [4]	$\mathrm{O}(n^3m)$	$\mathrm{O}(n^2)$
Gabow [9]	$\mathrm{O}(n^2m)$	$\mathrm{O}(m)$
	$\mathrm{O}(nM(n))$	$\mathrm{O}(n^2)$
Shepherd–Vetta [13]	$\mathrm{O}(n^2m)$	$\mathrm{O}(nm)$
This paper	$\mathrm{O}(n^2m)$	$\mathrm{O}(n^2)$

Previous algorithms for the shortest dijoin problem are shown in Table 2. Frank's algorithm is based on the optimality criterion for the submodular flow problem. Gabow improved Frank's algorithm to run in $\mathrm{O}(n^2m)$ time using the centroid tree, which was also used in the minimum cost k-arc-connected orientation problem. In contrast, Shepherd and Vetta [13] devised an algorithm that runs in $\mathrm{O}(n^2m)$ time without using any complex data structures. However, the algorithm of Shepherd and Vetta performs preprocessing, which requires to maintain n graphs using $\mathrm{O}(nm)$ space. Our algorithm solves the problem in $\mathrm{O}(n^2m)$ time and $\mathrm{O}(n^2)$ space without using complex data structures by reduction to the minimum cost strongly connected orientation problem.

The rest of this paper is organized as follows. Section 2 provides preliminaries on the k-arc-connected orientation problem, and Section 3 describes Frank's algorithm. Section 4 presents our method to find exchangeability arcs in Frank's algorithm. Finally, Section 5 exhibits an application of our algorithm to the shortest dijoin problem.

2 Arc-connectivity orientations and submodular flows

We consider graphs which are allowed to have multiple edges but no loops. Given an undirected graph $\overline{G} = (V, E)$ and $X \subseteq V$, we denote by $\delta_{\overline{G}}(X)$ the number of edges connecting X and $V \setminus X$. In a directed graph $G = (V, A)$, $\Delta^+_G(X)$ denotes the set of arcs of G from X to $V \setminus X$, and $\Delta^-_G(X)$ denotes the set of arcs of G from $V \setminus X$ to X. Let $\delta^+_G(X) = |\Delta^+_G(X)|$ and $\delta^-_G(X) = |\Delta^-_G(X)|$. For an arc set $F \subseteq A$, we denote by $\Delta^+_F(X)$ the set of arcs of F from X to $V \setminus X$. We also use Δ^-_F, δ^+_F, and δ^-_F in a similar way. If there is no ambiguity, we simply omit the subscript.

A pair of vertex subsets $X, Y \subseteq V$ is *crossing* if $X \cap Y \neq \emptyset$, $X \setminus Y \neq \emptyset$, $Y \setminus X \neq \emptyset$, and $X \cup Y \neq V$. A family $\mathcal{F} \subseteq 2^V$ is a *crossing family* if $X \cup Y, X \cap Y \in \mathcal{F}$ for any crossing $X, Y \in \mathcal{F}$. A function f over a crossing family $\mathcal{F}$ is *submodular on* $\mathcal{F}$ if $f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$ holds for any crossing $X, Y \in \mathcal{F}$.

Given an undirected graph $\overline{G} = (V, E)$ and costs of orientations in both directions for each edge, let $G = (V, A)$ be a directed graph obtained by orienting each edge of $\overline{G}$ in the direction of smaller cost. Let d be a function on A such that $d(e)$ represents the incremental cost by reversing the arc e. In other words, let $c(e)$ be the cost to orient the edge in the same direction of e, and let $c(\overline{e})$ be the cost to orient the edge in the reverse direction of e. Then d is given by $d(e) = c(\overline{e}) - c(e)$ for $e \in A$. Note that the incremental cost function d can be treated as a vector indexed by A. We intend to obtain a k-arc-connected directed graph by reversing some arcs of G. For a vector $x \in \{0,1\}^A$, we denote by $G_x = (V, A_x)$ the directed graph obtained from G by reversing every arc $e \in A$ with $x(e) = 1$.

Let $\mathcal{F}$ be the set of all the proper nonempty subsets of V. Then $\mathcal{F}$ forms a crossing family. Let b be a function on $\mathcal{F}$ defined by $b(X) = \delta^+(X) - k$. For any vertex subsets $X, Y \in \mathcal{F}$, if $X \cap Y \neq \emptyset$ and $X \cup Y \neq V$, then $\delta^+(X) + \delta^+(Y) \geq \delta^+(X \cup Y) + \delta^+(X \cap Y)$, and hence $b(X) + b(Y) \geq b(X \cup Y) + b(X \cap Y)$. Thus b is submodular on the crossing family $\mathcal{F}$.

The minimum cost k-arc-connected orientation problem is formulated as follows:

$$
(1) \qquad \begin{array}{lll}
\text{minimize} & dx & \\
\text{subject to} & x\left(\Delta^+(X)\right) - x\left(\Delta^-(X)\right) \leq b(X) & (\forall X \in \mathcal{F}), \\
 & x \in \{0,1\}^A, &
\end{array}
$$

where $x(F) = \sum_{e \in F} x(e)$ for any $F \subseteq A$.

Relaxing the 0-1 constraints, we obtain a linear program:

$$
(2) \qquad \begin{array}{lll}
\text{minimize} & dx & \\
\text{subject to} & x\left(\Delta^+(X)\right) - x\left(\Delta^-(X)\right) \leq b(X) & (\forall X \in \mathcal{F}), \\
 & 0 \leq x \leq 1, &
\end{array}
$$

which is a special case of the submodular flow problem. It is shown by Edmonds and Giles [3] that if (2) has an optimal solution, then it also has a 0-1 optimal solution, which provides an optimal solution to (1).

3 Tight sets and potentials

Given a feasible solution x of (1), we call $X \in \mathcal{F}$ an *x-tight set* if $x\left(\Delta^{+}(X)\right) - x\left(\Delta^{-}(X)\right) = b(X)$. In other words, X is x-tight if $\delta^{+}_{G_x}(X) = k$. The following lemma is well known.

Lemma 3. *If $X, Y \in \mathcal{F}$ are x-tight sets with $X \cup Y \neq V$ and $X \cap Y \neq \emptyset$, then $X \cup Y$ and $X \cap Y$ are x-tight sets.*

Proof. Denote $\sigma_x(X) = x\left(\Delta^{+}(X)\right) - x\left(\Delta^{-}(X)\right)$. Then we have

$$b(X) + b(Y) = \sigma_x(X) + \sigma_x(Y) = \sigma_x\left(X \cup Y\right) + \sigma_x\left(X \cap Y\right) \leq b\left(X \cup Y\right) + b\left(X \cap Y\right) \leq b(X) + b(Y),$$

which implies $\sigma_x(X \cup Y) = b(X \cup Y)$ and $\sigma_x(X \cap Y) = b(X \cap Y)$. Thus, both $X \cup Y$ and $X \cap Y$ are x-tight sets. □

Fix a feasible solution x of (1). Let $R(v)$ be the intersection of all the x-tight sets containing v. When there exists no x-tight set containing v, we set $R(v) = V$. Note that $R(v)$ is not necessarily an x-tight set. A function p defined on V is called a *potential*. For a potential p, the reduced cost $d_p(e)$ of an arc e from u to v is defined by

$$d_p(e) = d(e) - p(v) + p(u).$$

A potential p is *optimal* if p satisfies the following conditions:

(A) $$x(e) = 0 \Rightarrow d_p(e) \geq 0,$$

(B) $$x(e) = 1 \Rightarrow d_p(e) \leq 0,$$

(C) $$u \in R(v) \Rightarrow p(u) \geq p(v).$$

A potential reflects dual variables of (2). It follows from the complementarity slackness that the feasible solution x is an optimal solution if and only if there exists an optimal potential [5].

4 Frank's algorithm

In this section, we describe Frank's algorithm for the minimum cost k-arc-connected orientation problem. This algorithm starts with x and p that satisfy (A) for any $e \in A$ and (C) for any $u, v \in V$. Keeping conditions (A) and (C), it updates x and p so as to reduce the number of arcs violating (B).

First, let x be a feasible solution of (1) which can be found in $\mathrm{O}(kn^2(\sqrt{kn} + k^2 \log(n/k)))$ time [7], assign $p = 0$ so that the conditions (A) and (C) are satisfied. We then turn the feasible solution x into an initial feasible solution with at most $2kn$ arcs violating (B) as follows.

Select an arbitrary vertex r in V. Since G_x is k-arc-connected, it has k arc-disjoint r-arborescences by Theorem 2. Let F^{+} be the set of arcs used in the r-arborescences, which can be found in $\mathrm{O}(km \log(n^2/m))$ time [8]. We can also pack k arc-disjoint r-arborescences in the graph obtained by reversing all arcs. Let F^{-} be the set of arcs whose reversed arcs are used in the r-arborescences. Set x' as

$$x'(e) = \begin{cases} x(e) & (e \in F^{+} \cup F^{-}), \\ 0 & (\text{otherwise}). \end{cases}$$

Then there are at most $2kn$ arcs satisfying $x'(e) = 1$, and $G_{x'}$ determined by x' is k-arc-connected. This is because any vertex subset $X \in \mathcal{F}$ containing r satisfies $\delta^-_{G_{x'}}(X) \geq \delta^-_{F^+}(X) \geq k$ and any vertex subset $X \in \mathcal{F}$ not containing r satisfies $\delta^-_{G_{x'}}(X) \geq \delta^-_{F^-}(X) \geq k$. Thus the initial conditions (A) and (C) are satisfied, and there are at most $2kn$ arcs violating (B).

The main algorithm is described as follows.

An algorithm for minimum cost k-arc-connected orientation

Input: A directed graph $G = (V, A)$, incremental cost d, a feasible solution x of (1), and a potential p, satisfying (A) and (C) with at most $2kn$ arcs violating (B).

Output: A minimum cost k-arc-connected orientation G_x.

Step 1.

1.0. For each pair of $u, v \in V$, determine whether $u \in R(v)$ or not.

1.1. If all arcs satisfy (B), then go to Step 4. Otherwise, select $a \in A$ violating (B). Let s and t be the initial and terminal vertices of a, respectively.

1.2. For the reverse arc $\overline{e}$ of each $e \in A$, let $d_p(\overline{e}) = -d_p(e)$. Construct an auxiliary graph $G^*_x = (V, A^*_x \cup D^*_x)$ with arc sets

$$A^*_x = \{e \mid e \in A_x,\ d_p(e) \leq 0\},$$
$$D^*_x = \{uv \mid u \in R(v),\ p(u) = p(v)\}.$$

Arcs in D^*_x are called *exchangeability arcs.*

1.4. Find a directed path from s to t in G^*_x by a labeling technique. If such a directed path exists, then let P be the one with the minimum number of arcs and go to Step 3.

Step 2.

2.0. Let S be the set of vertices reachable from s in G^*_x. Compute $\varepsilon = \min(\alpha, \beta, \gamma)$, where

$$\alpha = d_p(a),$$
$$\beta = \min\left\{d_p(e) \mid e \in \Delta^+_{G_x}(S)\right\},$$
$$\gamma = \min\left\{p(u) - p(v) \mid u \in S,\ v \notin S,\ u \in R(v)\right\}.$$

When the minimum is taken over the empty set, it is defined to be ∞. Set $p(v) := p(v) - \varepsilon$ for $v \in S$.

2.1. If $\varepsilon = \alpha$, then delete all the labels and go to Step 1.1 (Note that $R(v)$ does not change). Otherwise, go to Step 1.2.

Step 3.

Change x along P so that every arc in $A_x \cap P$ gets reversed. Set $x(a) := 0$. Go to Step 1.0.

Step 4.

The current directed graph G_x is a minimum cost k-arc-connected orientation.

In this algorithm, Step 3 is iterated at most $2kn$ times. In each iteration, Step 2 is done until all vertices are labeled, so it takes $O(n^2)$ time. Thus, if Step 1.0 runs in $g(n, m, k)$ time, the entire algorithm takes $O(kn(n^2 + g(n, m, k)))$ time in total. This algorithm needs $O(n^2)$ space to keep the auxiliary graph G^*_x.

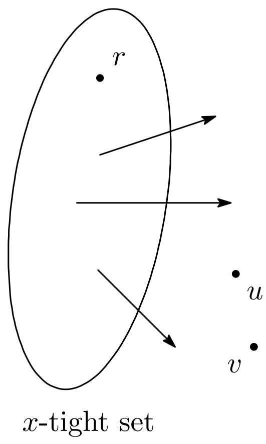

Figure 1: A member of $\mathcal{F}_u^-$ and v.

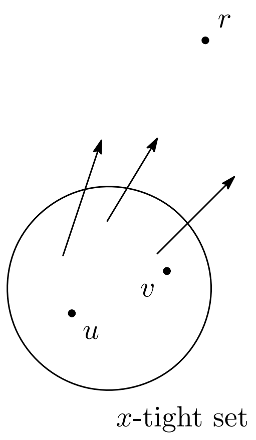

Figure 2: A member of $\mathcal{F}_v^+$ and u.

5 Finding exchangeability arcs

In Frank's algorithm for the minimum cost k-arc-connected orientation problem, we need to find exchangeability arcs. In this section, we describe a method to find all the exchangeability arcs in $\mathrm{O}(k^2n^2+nm)$ time so that our algorithm runs in $\mathrm{O}(kn(k^2n^2+nm))$ time. Gabow's algorithm computes them by using a data structure called *centroid tree* which represents all the x-tight sets. In contrast, our method finds all the exchangeability arcs without using such a sophisticated data structure. The key observation is that we do not need all x-tight sets.

Note that X is an x-tight set if and only if $\delta^+_{G_x}(X) = k$. Fix a certain vertex $r \in V$. First, we determine whether $u \in R(v)$ or not for $u, v \in V \setminus \{r\}$. Let $\mathcal{F}_u^-$ be the collection of all x-tight sets which contain r and exclude u. Let $\mathcal{F}_v^+$ be the collection of all x-tight sets which contain v and exclude r.

If $X, Y \in \mathcal{F}_u^-$, it is obvious that $X \cup Y, X \cap Y$ contain r and exclude u and that both $X \cup Y$ and $X \cap Y$ are x-tight sets by Lemma 3. Thus, if $\mathcal{F}_u^- \neq \emptyset$, then there is a unique maximal set in $\mathcal{F}_u^-$, to be denoted by R_u^-. If $\mathcal{F}_u^- = \emptyset$, let $R_u^- = \emptyset$. Similarly, $\mathcal{F}_v^+$ is closed with respect to union and intersection. So, if $\mathcal{F}_v^+ \neq \emptyset$, then there is a unique minimal set in $\mathcal{F}_v^+$, to be denoted by R_v^+. If $\mathcal{F}_v^+ = \emptyset$, let $R_v^+ = V$.

Lemma 4. *We have $u \in R(v)$ if and only if $v \notin R_u^-$ and $u \in R_v^+$ hold.*

Proof. If $u \in R(v)$, there are neither x-tight sets which contain r, v and exclude u nor x-tight sets which contain v and exclude r, u. In other words, no members of $\mathcal{F}_u^-$ contain v (Fig. 1), and all members of $\mathcal{F}_v^+$ contain u (Fig. 2). Thus $u \in R(v)$ is equivalent to $v \notin R_u^-$ and $u \in R_v^+$. $\square$

We now describe how to compute R_u^- by using Theorem 2. This procedure works with the graph G_x.

First, we pack k arc-disjoint r-arborescences in G_x. Since G_x is k-arc-connected, it follows from Theorem 2 that there are k arc-disjoint r-arborescences. The packing can be done in $\mathrm{O}(k^2n^2)$ time [8]. After this preprocessing, we can compute R_u^- easily for each vertex $u \in V$ as follows.

For any $u \in V$, we can find an r-u path consisting of arcs of an r-arborescence. Hence we can find k arc-disjoint r-u paths. Let T_u be the set of arcs used in the paths, and let G_{T_u} be an auxiliary graph obtained from G_x by reversing arcs of T_u. If there is a directed path P_u from r to u in G_{T_u}, the symmetric difference $T'_u = T_u \triangle P_u$ includes $k+1$ arc-disjoint r-u paths of G_x, which implies $\mathcal{F}_u^- = \emptyset$, $R_u^- = \emptyset$. If there is no directed path from r to u, we can compute R_u^- by the following lemma.

Lemma 5. *Let S_u be the set of vertices from which we can reach u in G_{T_u}. Then we have $R_u^- = V \setminus S_u$.*

Proof. Obviously, we have $u \in S_u$ and $r \notin S_u$. Since T_u consists of k arc-disjoint r-u paths, we have $\delta^-_{T_u}(S_u) - \delta^+_{T_u}(S_u) = k$. By the definition of S_u, we have $\delta^+_{T_u}(S_u) + \delta^-_{G_x \setminus T_u}(S_u) = 0$, and hence

$$\delta^+_{G_x}(V \setminus S_u) = \delta^-_{G_x}(S_u) = \delta^-_{T_u}(S_u) + \delta^-_{G_x \setminus T_u}(S_u) = \delta^+_{T_u}(S_u) + k + \delta^-_{G_x \setminus T_u}(S_u) = k.$$

Thus we have $V \setminus S_u \in \mathcal{F}^-_u$.

By the definition of S_u, we have $\delta^+_{T_u}(X) + \delta^-_{G_x \setminus T_u}(X) \geq 1$ for any vertex subset X with $u \in X \subsetneq S_u$. Then we have

$$\delta^+_{G_x}(V \setminus X) = \delta^-_{G_x}(X) = \delta^-_{T_u}(X) + \delta^-_{G_x \setminus T_u}(X) = \delta^+_{T_u}(X) + k + \delta^-_{G_x \setminus T_u}(X) \geq k + 1,$$

which means that $V \setminus X$ is not an x-tight set. Thus we obtain $R^-_u = V \setminus S_u$. □

We now analyze the running time of finding exchangeability arcs. Packing k arc-disjoint r-arborescences requires $\mathrm{O}(k^2n^2)$ time, and computing R^-_u requires $\mathrm{O}(m)$ time for each vertex $u \in V$. Thus we can compute R^-_u for all u in $\mathrm{O}(k^2n^2 + nm)$ time. We can also compute $V \setminus R^+_v$ by reversing all arcs and executing the same procedure. This also takes $\mathrm{O}(k^2n^2 + nm)$ time. We have $r \in R(v)$ if and only if $R^+_v = V$, and $v \in R(r)$ if and only if $R^-_v = \emptyset$. So we can determine whether $u \in R(v)$ or not by computing R^+_v and R^-_v for all $v \in V$. Thus for all $u, v \in V$ we can determine whether $u \in R(v)$ or not in $\mathrm{O}(k^2n^2 + nm)$ time, which implies the following theorem.

Theorem 6. *The total running time of our algorithm is* $\mathrm{O}(kn(k^2n^2 + nm))$.

6 Application to shortest dijoins

A directed graph $G = (V, A)$ is *weakly connected* if G is a connected graph when we ignore the orientations of the arcs. Given a weakly connected directed graph $G = (V, A)$, if proper nonempty vertex subset $S \subsetneq V$ satisfies $\Delta^+(S) = \emptyset$, then we say $\Delta^-(S)$ is a *directed cut* (*dicut*). An arc subset $F \subseteq A$ is a *dijoin* (*directed cut cover*) if F meets every dicut. Thus G can be made strongly connected by contracting (or adding the reverse arc of) every arc of a dijoin. The following theorem establishes a min-max relation between dijoins and packing dicuts.

Theorem 7 (Lucchesi and Younger [11])**.** *The minimum cardinality of a dijoin is equal to the maximum number of disjoint dicuts.*

It is clear that the cardinality of a dijoin is greater than or equal to the number of disjoint dicuts. This theorem guarantees that there must exist a pair that attains the equality. A simple proof of this theorem is given in [10].

Given a length function $l : A \to \mathbf{Z}_+$, we consider finding a dijoin F of minimum length $\sum_{e \in F} l(e)$. This is called the *shortest dijoin problem.* The optimal value is characterized by the following generalization of Theorem 7.

Theorem 8 (Lucchesi and Younger [11])**.** *The minimum total length of a dijoin is equal to the maximum size of a collection $\mathcal{C}$ of dicuts such that at most $l(e)$ dicuts of $\mathcal{C}$ contain e for every $e \in A$, where $\mathcal{C}$ allows duplication, i.e.,*

$$\min \left\{ \sum_{e \in F} l(e) \;\middle|\; F : \textit{dijoin} \right\} = \max \left\{ |\mathcal{C}| \;\middle|\; \textit{at most } l(e) \textit{ dicuts of } \mathcal{C} \textit{ contain } e \textit{ for each } e \in A \right\}.$$

The shortest dijoin problem can be reduced to the minimum cost strongly connected orientation problem as follows [6]. Suppose we are given a directed graph $G = (V, A)$ and $l(e)$ for each $e \in A$. Let $\overline{G'}$ be the undirected graph made by reduplicating each arc e as edges e_1, e_2. Since G is weakly connected, undirected graph $\overline{G'}$ is 2-edge-connected. So $\overline{G'}$ has a 1-arc-connected (strongly connected) orientation. We consider a strongly connected orientation problem with costs for orienting each edge as follows: $c(e_1) = c(e_2) = 0$, $c(\overline{e_1}) = l(e)$, $c(\overline{e_2}) = +\infty$ where $c(e_1)$ means the cost for orienting e_1 in the same direction as e, $c(\overline{e_1})$ means the cost for orienting e_1 in the reverse direction of e, and the same for $c(e_2)$ and $c(\overline{e_2})$. There exists an orientation whose cost is finite, so the set of edges oriented in the reverse direction of G corresponds to a shortest dijoin.

When applied to the minimum cost strongly connected orientation problem, our algorithm for arc-connectivity orientation runs in $O(n^2m)$ time. Furthermore, packing k arborescences is easy for $k = 1$. Thus, we can solve the shortest dijoin problem in $O(n^2m)$ time and $O(n^2)$ space by reduction to the minimum cost strongly connected orientation problem.

We have implemented our algorithm, and applied it to some instances of the shortest dijoin problem. Our experiments were conducted on the PC with an Intel Pentium4, CPU 1.60GHz, 1GB of memory. All programs are written in C++ with C++ class library LEDA. We generated random directed graphs for given n, m, and when the graph is weakly connected we applied our algorithm to the graph. All the running times reported here are in seconds, and we only report the user CPU time, excluding the time of constructing graphs. We made five instances for each n, m, and each number in the table is the time averaged over five runs. *Ratio* in the table, meaning the time divided by n^2m, are almost constant. Thus the running time of our algorithm is confirmed to be $O(n^2m)$.

Table 3: Running time for instances of the shortest dijoin problem.

n	m	time [s]	ratio [μs]
15	30	0.03	4.74
15	45	0.05	4.94
15	90	0.08	3.95
30	60	0.20	3.74
30	90	0.31	3.83
30	180	0.53	3.28
50	100	0.84	3.34
50	150	1.32	3.53
50	300	2.33	3.11
100	200	6.89	3.44
100	300	10.70	3.57
100	600	18.35	3.06
200	400	58.19	3.64
200	600	92.63	3.86
200	1200	166.75	3.47
300	600	202.27	3.75
300	900	347.80	4.29
300	1800	661.02	4.08

References

[1] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symbolic Computation*, 9 (1990), pp. 251–280.

[2] J. Edmonds, Edge-disjoint branchings, *Combinatorial Algorithms*, R. Rustin, ed., Algorithmics Press, New York, 1973, pp. 91–96.

[3] J. Edmonds and R. Giles, A min-max relation for submodular functions on graphs, *Annals of Discrete Mathematics*, 1 (1977), pp. 185–204.

[4] A. Frank, How to make a digraph strongly connected, *Combinatorica*, 1 (1981), pp. 145–153.

[5] A. Frank, An algorithm for submodular functions on graphs, *Annals of Discrete Mathematics*, 16 (1982), pp. 97–120.

[6] S. Fujishige and N. Tomizawa, An algorithm for finding a minimum-cost strongly connected reorientation of a directed graph, manuscript, 1982.

[7] H. N. Gabow, A framework for cost-scaling algorithms for submodular flow problems, *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, 1993, pp. 449–458.

[8] H. N. Gabow, A matroid approach to finding edge connectivity and packing arborescences, *J. Computer and System Sciences*, 50 (1995), pp. 259–273.

[9] H. N. Gabow, Centroids, representations, and submodular flows, *J. Algorithms*, 18 (1995), pp. 586–628.

[10] L. Lovász, On two minimax theorems in graph, *J. Combinatorial Theory* (B), 21 (1976), pp. 96–103.

[11] C. L. Lucchesi and D. H. Younger, A minimax theorem for directed graphs, *J. London Mathematical Society* (2), 17 (1978), pp. 369–374.

[12] C. St. J. A. Nash-Williams, On orientations, connectivity and odd-vertex-pairings in finite graphs, *Canadian J. Mathematics*, 12 (1960), pp. 555–567.

[13] F. B. Shepherd and A. Vetta, Visualizing, finding, and packing dijoins, *Graph Theory and Combinatorial Optimization*, D. Avis, ed., Springer-Verlag, 2005, pp. 219–254.