AN IMPROVEMENT OF AN ALGORITHM FOR
CONSTRUCTION OF EDGE DISJOINT
BRANCHINGS

by

S. Moran

Technical Report #341
October 1984
An Improvement of an Algorithm for Construction of
Edge Disjoint Branchings

by

Shlomo Moran

Department of Computer Science
The Technion, Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT Let $G = (V, E)$ be a directed graph such that for some $r \in V$, and for every $u \in V - \{r\}$, there are $k$ edge disjoint directed paths from $r$ to $u$. We present an $O(k \cdot |V| \cdot |E| + k^3 \cdot |V|^2)$ algorithm to construct $k$ edge disjoint branchings rooted at $r$. This algorithm is a modification of an $O(k^2 \cdot |V| \cdot |E|)$ algorithm of Shiloach for the same task, which is based on Lovász' proof of a Theorem of Edmonds that implies the existence of such edge disjoint branchings.
1. Introduction

Let \( G = (V, E) \) be a directed graph (digraph), and let \( s, t \in V \). An \((s, t)\) path in \( G \) is a directed path from \( s \) to \( t \), and an \((s, t)\) cut is a set of edges that intersects every \((s, t)\) path. It is easily observed that if \( C \) is an \((s, t)\) cut then there is an \( S \subseteq V \) such that \( s \in S \), \( t \in V - S \) and \( C \) contains all the edges in \( G \) which are directed from a vertex in \( S \) to a vertex in \( V - S \). \( C \) is a minimum \((s, t)\) cut if it is an \((s, t)\) cut of minimum possible cardinality; this cardinality is denoted by \( \delta_C(s, t) \), and \( \min_{C} \delta_C(s, t) \) is denoted by \( K(G, s) \).

Let \( r \) be a vertex of a digraph \( G = (V, E) \). A branching (or a directed spanning tree) rooted at \( r \) is a subgraph of \( G \) in which there is exactly one (directed) path from \( r \) to every vertex of \( V \). Edmonds had proved in [1] that if \( K(G, r) = k \) then \( G \) contains \( k \) edge disjoint branchings rooted at \( r \). In [3] Lovász gave an alternative proof of Edmonds theorem, and remarked that his proof provides an algorithm which constructs the \( k \) disjoint branchings in \( O(fk \cdot |V| \cdot |E|) \) time, where \( f \) is time needed to decide for \( s, t \in V \) whether \( \delta_C(s, t) \geq k \). It is pointed out in [4] that if Ford and Fulkerson maximum flow algorithm is used for this latter task, then it can be solved in \( O(k \cdot |E|) \) time. It is also shown there that the number of such problems that have to be solved can be reduced to \( O(k \cdot |V|) \), which yields an \( O(k^2 \cdot |V| \cdot |E|) \) algorithm. (This algorithm improves an earlier \( O(k^2 \cdot |E|^2) \) algorithm [5]). In the next section we describe a modification of the Lovász - Shiloach algorithm, which is slightly simpler than the original one. This simplification is based on a simple property of edges in minimum cuts, which is given there. This property is later used to reduce the complexity to \( O(k \cdot |V| \cdot |E| + k^2 \cdot |V|^2) \). (To see that this is indeed an improvement, note that if \( K(G, r) = k \) then \( G \) has at least \( k \) edges entering each node except \( r \), and hence at least \( O(k \cdot |V|) \) edges).

2. The Algorithms of Lovász and Shiloach

To simplify the presentation, we use the following definitions:

**Definition 1:** Let \( G = (V, E) \) be a digraph, and let \( T \) be a subgraph of \( G \) which is a directed tree rooted at \( r \). Then \( T \) is economic if \( K(G(E - E(T)), r) = K(G, r) - 1 \).
Definition 2: Let $T = (V_1, E_1)$ be an economic subgraph of $G$. An edge $e$ in $E - E_1$ is available for $T$ in $G$ if $G(E_1 \cup \{e\})$ is an economic subgraph of $G$ (i.e., if $e$ can be added to $T$ such that the resulted graph remains an economic subgraph of $G$).

The following Lemma is implicit in Lovász' proof of Edmonds' Theorem ([3], see also [4], [2], Theorem 6.9):

**Lemma 2.1.** (Lovász): Let $T$ be an economic subgraph of a digraph $G$ which is not a branching of $G$, and let $\mathcal{A}$ be a maximal subset of $V$ satisfying the following properties:

(i) $T \subseteq \mathcal{A}$

(ii) $\mathcal{A} \subseteq V(T)$

(iii) There are exactly $k - 1$ edges in $G(E - E(T))$ which are directed from a vertex in $\mathcal{A}$ to a vertex in $V - \mathcal{A}$.

Then there exists an edge in $E$ which is directed from a vertex in $V(T) - \mathcal{A}$ to a vertex in $V - V(T) - \mathcal{A}$, and every such edge is available for $T$. Moreover, if there is no such a set $\mathcal{A}$, then every edge directed from a vertex in $V(T)$ to a vertex in $V - V(T)$ is available for $T$.

Lovász describes in [3] a polynomial time algorithm based on Lemma 2.1 which constructs $k = K(G,r)$ edge disjoint branchings rooted at $r$, and his algorithm was later improved in [4]. This improvement is described below:

In its first step, the algorithm computes $k = K(G,r)$ by solving simultaneously $|V| - 1$ zero-one flow problems from $r$ to $v$ (denoted $(r,v)$ flows), one flow problem for each $v \in V - \{r\}$. This is done so that if a maximum $(r,v)$ flow of $k$ units is found for some $v$, then no flow of $k + 1$ units is computed (see [4]). Since the computation of a flow of $k$ units can be done in $O(k |E|)$ time, the complexity of this step is $O(k |V| |E|)$. In the second step the algorithm constructs $k$ edge disjoint economic spanning trees of $G$, one tree at a time. Each such tree is constructed by starting with a tree consisting of the node $r$, and repeatedly adding available edges to it. The naive approach for this is to test each edge in its turn for availability (in a way to be described later), and to add it to the tree if it is available.
This may require \(|E|\) such tests. Shiloach had pointed out in [4] that if the test fails, then there exists a set of vertices \(A\) satisfying (i)-(iii) of Lemma 2.1, and it can be found by solving one flow problem (see below). Hence, an available edge can be found by finding such a set. This reduces the number of tests needed for the construction to \(|V-1|\) (actually, to \(|V|-2\), since by Lemma 2.1 the first edge in the tree can be taken to be any edge leaving \(r\)). When the economic spanning tree is completed, its edges are deleted from \(G\), and if in the remaining graph, \(G'\), \(K(G',r) > 0\), the next tree is constructed by repeating this procedure.

Since deleting the edges of an economic spanning tree rooted at \(r\) from \(G\) reduces \(K(G,r)\) by one, the algorithm does not terminate before \(k\) edge disjoint spanning trees rooted at \(r\) are constructed.

The complexity of the above algorithm depends on the complexity of determining for a given edge \(e\) and an economic subgraph \(T\) whether \(e\) is available for \(T\). The straightforward approach suggested in [3], [2] is to solve \(|V|-1\) \((r,v)\) flow problems over \(G(E-E(T)-\{e\})\). In [4] it is shown that the solution of one flow problem is sufficient for this task; we present below an alternative, more direct proof (and algorithm) of this result.

In the following lemmas \(T\) is an economic subgraph of \(G\) which is not a branching of \(G\), \(G_1 = G(E-E(T))\), and \(e\) is an edge directed from \(u \in V(T)\) to \(v \in V-V(T)\).

**Lemma 2.2.** \(e\) is not available for \(T\) iff \(e\) participates in some \((r,t)\) cut \(C\) in \(G_1\), where \(|C| = K(G_1,r) = k-1\).

**Proof:** By the definitions, \(K(G_1,r) = k-1\), and \(e\) is not available for \(T\) iff \(K(G_1-\{e\},r) < k-1\). Thus \(e\) is not available for \(T\) iff the removal of \(e\) from \(G_1\) decreases the cardinality of a certain \((r,t)\) cut in \(G_1\) from \(k-1\) to \(k-2\). The Lemma follows.

**Lemma 2.3.** If \(e\) is not available for \(T\), then \(e\) participates in an \((r,v)\) cut \(C\) in \(G_1\), where \(|C| = K(G_1,r) = k-1\).

\(^1\) For a graph \(G=(V,E)\) and \(F \subseteq E\), \(G-F\) denotes \(G(E-F)\).
4. **Proof**: By Lemma 2.2, \( e \) participates in some \((r,t)\) cut \( C \) of cardinality \( k-l \). We shall prove that \( C \) is also an \((r,v)\) cut. For contradiction, let \( G_2 = G_1 - C \) and assume that there is an \((r,v)\) path in \( G_2 \). Then since there is no \((r,t)\) path in \( G_2 \), this implies that there is no \((v,t)\) path in \( G_2 \). Hence \( e \), which is directed into \( v \), can be added to \( G_2 \) without creating any path from \( r \) to \( t \). This implies that \( C_1 = C - \{e\} \) is also an \((r,t)\) cut in \( G_1 \). But \( |C_1| = k-2 \), which contradicts the assumption that \( K(G_1,r) = k-1 \).

**Corollary 2.4.** (Shiloach): The algorithm described above can be implemented in \( O(k^2 |V| |E|) \) time.

**Proof**: The complexity of the first step of the algorithm (computing \( k = K(G,r) \)) was shown to be \( O(k |V| |E|) \). In the construction of an economic spanning tree in the second step, at most \( |V|-2 \) edges are tested for availability. By Lemma 2.3 this can be done by determining whether \( \delta_{G_1-\{e\}}(r,v) < k-1 \), where \( G_1, e \) and \( v \) are as defined above. This can be done by solving one \((r,v)\) flow problem, in which at most \( k \) augmenting paths are constructed. If the test fails, then a set \( A \) that satisfies (i)-(iii) of Lemma 2.1 is found by solving a \((v,r)\) flow problem on the graph obtained by reversing the directions of \( G_1 \)'s edges (see [4]). It follows that the total number of flow problems that have to be solved is \( O(k |V|) \), and each of those problems can be solved in \( O(k |E|) \) time. The corollary follows.

3. **A further improvement**

We conclude by showing how the complexity of the algorithm can be reduced by another application of Lemma 2.3.

**Theorem 3.1.** Let \( G = (V,E) \) be a digraph and let \( r \in V \). There is an \( O(k |V| |E| + k^3 |V|^2) \) algorithm which constructs \( k = K(G,r) \) edge disjoint branchings rooted at \( r \).

**Proof** (An outline): Modify the algorithm described in the previous section as follows: After \( k = K(G,r) \) is computed, repeatedly remove from \( E \) edges whose removal does not decrease \( K(G,r) \). By Lemma 2.3 (with \( G_1 = G \) and "available" means
"removable"), an edge directed from \( u \) to \( v \) is removable if it does not participate in an \((r,v)\) cut of cardinality \( k \), thus these edges can be found by computing an \((r,v)\) flow of size \( k \), and removing from the graph the edges entering \( v \) whose flow function is zero in this flow. Since there is a zero-one \((r,v)\) flow of \( k \) units which does not use any of the removed edges, none of them participate in an \((r,v)\) cut of cardinality \( k \). Hence, by Lemma 2.3, none of them participate in an \((r,t)\) cut of cardinality \( k \) for any \( t \in V \), and hence their removal does not decrease \( K(G,r) \).

This step requires the computation of \(|V|-1\) flows, each of at most \( k \) units, and hence can be done in \( O(k|V||E|) \) time. In the remaining graph there are exactly \( e' = k(|V|-1) \) edges. Hence, the construction of the \( k \) edge disjoint spanning trees on this graph can be done in \( O(k^2|V|e') = O(k^3|V|^2) \) steps.

REFERENCES: