A NOTE ON ESWARAN AND TARJAN'S ALGORITHM FOR THE STRONG CONNECTIVITY AUGMENTATION PROBLEM

S. Raghavan
The Robert H. Smith School of Business
University of Maryland, College Park

Abstract In a seminal paper Eswaran and Tarjan [1] introduced several augmentation problems and presented linear time algorithms for them. This paper points out an error in Eswaran and Tarjan's algorithm for the strong connectivity augmentation problem. Consequently, the application of their algorithm can result in a network that is not strongly connected. Luckily, the error can be fixed fairly easily, and this note points out the remedy yielding a "corrected" linear time algorithm for the strong connectivity augmentation problem.

1. Introduction

Approximately 30 years ago Eswaran and Tarjan introduced the strong connectivity augmentation problem that can be described as follows. Let $D = (N, A)$ be a directed graph with node set N and arc set A. The strong connectivity augmentation problem is the problem of finding a minimum cardinality set of arcs A^{AUG} such that $D = (N, A \cup A^{\text{AUG}})$ is strongly connected.

Eswaran and Tarjan also describe an elegant linear time algorithm for the problem. Their algorithm consists of three steps. It first condenses the directed graph by shrinking every strongly connected component of the directed graph to obtain an acyclic digraph. A node with no incoming arc in this acyclic graph is called a source, and a node with no outgoing arc in this acyclic graph is called a sink. The second step of their algorithm constructs a particular ordering of sources and sinks with a *desired* set of properties. Their third step then adds arcs to strongly connect this acyclic digraph. (They show that it suffices to solve the augmentation problem on the condensed graph.) The correctness of their procedure relies on the second step of their algorithm, where an ordering of sources and sinks with a set of desired properties is constructed.

In this note we point out an error in Eswaran and Tarjan's strong connectivity augmentation algorithm. Specifically, we show that the algorithm described in their paper *does not* provide an ordering of sources and sinks with the desired set of properties. Consequently, the application of their augmentation algorithm (as will be shown with a counterexample) can lead to a directed graph that is not strongly connected. We also provide a corrected procedure for the second step of their algorithm that runs in linear time.

2. The Algorithm and the Error

We now review Eswaran and Tarjan's algorithm and elaborate on the error within it. We note that our notation differs slightly from Eswaran and Tarjan.

Given a directed graph $D = (N, A)$ the first step of their procedure is to create its condensation $D^{\mathrm{SCC}} = (N^{\mathrm{SCC}}, A^{\mathrm{SCC}})$, obtained by shrinking every strongly connected component of D. D^{SCC} contains one node for every strong component of D, and there is an arc (i, j) in D^{SCC} if there is an arc in D from any node in the strong component corresponding to node $i \in D^{\mathrm{SCC}}$ to any node in the strong component corresponding to node $j \in D^{\mathrm{SCC}}$. For notational convenience they define the two mappings α and β as follows. For every $v \in N$, let $\alpha(v)$ be the node in D^{SCC} corresponding to the strong component in D that contains node v. For every $v \in N^{\mathrm{SCC}}$, $\beta(v)$ defines any node in the strongly connected component of D corresponding to node v. Eswaran and Tarjan show the following lemma, proving that it suffices to solve the augmentation problem on D^{SCC}.

LEMMA 1 *Let X be an augmenting set of arcs which strongly connects D. Then $\alpha(X) = \{(\alpha(v), \alpha(w)) | (v, w) \in X, \alpha(v) \neq \alpha(w)\}$ is a set of arcs which strongly connects D^{SCC}. Conversely, let Y be an augmenting set of arcs which strongly connects D^{SCC}. Then $\beta(Y) = \{(\beta(x), \beta(y)) | (x, y) \in Y\}$ is a set of arcs which strongly connects D.*

In the acyclic digraph D^{SCC}, a source is defined to be a node with outgoing but no incoming arcs, a sink is defined to be a node with incoming but no outgoing arcs, and an isolated node is defined to be a node with no incoming and no outgoing arcs. Let s, t and q denote the number of source nodes, sink nodes, and isolated nodes respectively in D^{SCC}, and assume without loss of generality $s \leq t$.

The second step of the algorithm finds an index p and an ordering $v(1), \ldots,$ $v(s)$ of sources of D^{SCC}, $w(1), \ldots, w(t)$ of sinks of D^{SCC} with the following properties:

1 there is a path from $v(i)$ to $w(i)$ for $1 \leq i \leq p$;

2 for each source $v(i)$, $p + 1 \leq i \leq s$ there is a path from $v(i)$ to some $w(j)$, $1 \leq j \leq p$; and

3 for each sink $w(j)$, $p+1 \leq j \leq t$, there is a path from some $v(i)$, $1 \leq i \leq p$, to $w(j)$.

Let $x(1), \ldots, x(q)$ denote the set of isolated nodes of D^{SCC}. They show that a minimal augmentation of D^{SCC} is obtained from the arc set.[1]

$$\begin{aligned} A^{\text{ASC}} &= \{(w(i), v(i+1)) | 1 \leq i < p\} \cup \{(w(i), v(i)) | p+1 \leq i \leq s\} \\ &\cup \begin{cases} (w(p), v(1)) & \text{if } q = 0 \text{ and } s = t; \\ (w(p), w(s+1)) \cup \{(w(i), w(i+1)) | s+1 \leq i < t\} & \\ \cup (w(t), v(1)) & \text{if } q = 0 \text{ and } s < t; \\ (w(p), w(s+1)) \cup \{(w(i), w(i+1)) | s+1 \leq i < t\} & \\ \cup (w(t), x(1)) \cup \{(x(i), x(i+1)) | 1 \leq i < q\} & \\ \cup (x(q), v(1)) & \text{otherwise.} \end{cases} \end{aligned}$$

Eswaran and Tarjan show that $\max(s,t) + q$ is a lower bound on the number of arcs needed to augment D^{SCC} so that it is strongly connected.[2] Note that the augmenting set A^{ASC} contains $t+q$ arcs. To see that the addition of these arcs strongly connects D^{SCC} observe that by construction the nodes $v(1), \ldots, v(p), w(1), \ldots, w(p)$, $w(s+1), \ldots, w(t), x(1), \ldots, x(q)$ are on a directed cycle (denoted by C) and thus strongly connected. For $v(p+1), \ldots,$ $v(s)$, $w(p+1), \ldots, w(s)$, the correctness of their procedure relies on Properties (2) and (3). Due to Property (2) there is a path from each $v(i)$, $p+1 \leq i \leq s$, to some node $w(j)$, $1 \leq j \leq p$, and thus by construction to every node in the cycle C. From Property (3), and the addition of the arcs $(w(i), v(i))$, for $p+1 \leq i \leq s$, there is a path from every node in the cycle C to each $v(i)$, $p+1 \leq i \leq s$. A similar argument shows that there is a directed path from the nodes in the cycle to each $w(i)$, $p+1 \leq i \leq s$, and from each $w(i)$, $p+1 \leq i \leq s$ to the nodes in the cycle C.

The Eswaran and Tarjan paper provides the algorithm ST shown in Figure 1. We note that for any ordering of sources and sinks satisfying Properties (1)–(3), arbitrarily permuting the ordering of sources $v(p+1), \ldots, v(s)$, sinks $w(p+1), \ldots, w(t)$, and isolated nodes $x(1), \ldots, x(q)$ results in an ordering that continues to satisfy Properties (1)–(3). Consequently the algorithm focuses on obtaining p and the ordering $v(1), \ldots, v(p)$ of sources and $w(1), \ldots, w(p)$ of sinks satisfying Property (1), while ensuring that the remaining sources and sinks satisfy the desired Properties (2) and (3). The authors state that it is obvious that the algorithm ST finds a sequence of sources

[1]There is a typographical error in the first line of the equation shown on page 657 of [1] that is corrected here.

[2]Since there are $s+q$ nodes with no incoming arcs, at least $s+q$ arcs are needed to augment D^{SCC} so that it is strongly connected. Similarly, as there are $t+q$ nodes with no outgoing arcs, at least $t+q$ arcs are needed to augment D^{SCC} so that it is strongly connected.

```
1.  algorithm ST: begin
2.    procedure SEARCH($x$);
3.      if $x$ is unmarked then
4.        begin
5.          if $x$ is a sink and $(w = 0)$ then $w := x$;
6.          mark $x$;
7.          for each $y$ such that $(x, y)$ is an edge do SEARCH($y$);
8.        end SEARCH
9.    initialize all nodes to be unmarked;
10.   $i := 0$;
11.   while some sink is unmarked do begin
12.     choose some unmarked source $v$;
13.     $w := 0$;
14.     SEARCH($v$);
15.     if $w \neq 0$ then begin
16.       $i := i + 1$;
17.       $v(i) := v$;
18.       $w(i) := w$;
19.   end end
20.   $p := i$;
21. end ST;
```

Figure 1. Eswaran and Tarjan's algorithm to find an ordering of sources and sinks that satisfies Properties (1)–(3).

and sinks satisfying Properties (1)–(3). We now show that this statement is not true, and the algorithm can produce an ordering of sources and sinks that does not satisfy Property (2). For convenience, we also display in Figure 2 Eswaran and Tarjan's algorithm for the strong connectivity augmentation problem.

Consider the example shown in Figure 3. The acyclic digraph shown in Figure 3(a) has two sources nodes a and c, and two sinks nodes b and d. There is a directed path from source node a to sink node b and the first arc on this path is (a, k). There is also a directed path from source node a to sink node d and the first arc on this path is (a, l). There is a directed path from source node c to sink node d, and the paths from a to d and c to d are identical from node m onwards. Suppose the search starts from a (i.e., a is the first unmarked source selected in line 12), and suppose that in line 7 of the algorithm arc (a, k) is considered before arc (a, l). Then the procedure will first find sink b and set

```
algorithm STRONGCONNECT: begin
SC1: Use depth-first search to form the condensation $D^{SCC}$ of $D$,
     identifying the sources, sinks, and isolated nodes of $D^{SCC}$;
SC2: Apply algorithm ST to $D^{SCC}$ to find a set of sources and sinks
     satisfying (1)–(3);
SC3: Construct the corresponding augmenting set of arcs $A^{ASC}$;
SC4: Convert $A^{ASC}$ into an augmenting set of arcs $A^{AUG}$ for $D$, using
     Lemma 1;
end STRONGCONNECT;
```

Figure 2. Eswaran and Tarjan's algorithm to solve the strong connectivity augmentation problem.

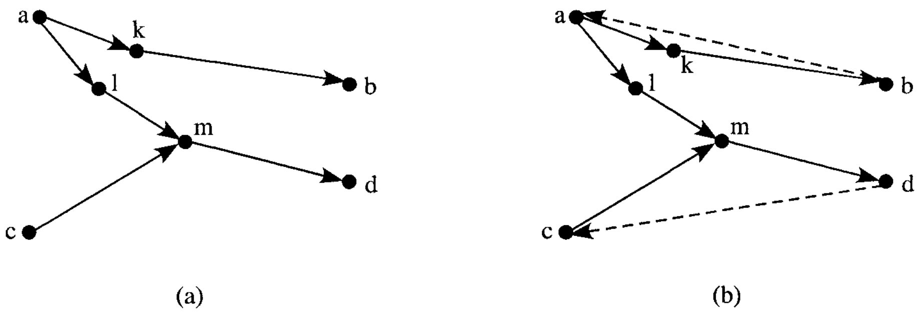

Figure 3. Counterexample that shows algorithm ST does not find an ordering of sources and sinks that satisfies Property (2). (a) Running algorithm ST gives $p = 1$, $v(1) = a$, $w(1) = b$, $v(2) = c$, $w(2) = d$. (b) Augmenting by adding arcs $(w(1), v(1)) = (b, a)$ and $(w(2), v(2)) = (d, c)$ results in a digraph that is not strongly connected.

$w := b$ (in line 5). It will then continue the search from a considering arc (a, l) in line 7 of the algorithm. This will result in traversing the path from a to d and marking nodes m and d. It will then set $v(1) := a$ and $w(1) := b$ in lines 17 and 18. Since all sinks are marked the procedure stops. Eswaran and Tarjan's procedure will also set $p := 1$, $v(2) := c$ and $w(2) := d$. Observe now that there is no path from c to node b, and thus this ordering does not satisfy Property (2). The augmentation procedure adds the arcs $(w(1), v(1)) = (b, a)$ and $(w(2), v(2)) = (d, c)$. The resulting digraph shown in Figure 3(b) is obviously not strongly connected.

3. The Correction

As the example indicates algorithm ST fails to find an ordering that satisfies Property (2). The problem within the algorithm is that the search continues from a source that has found an unmarked sink. This search may mark unmarked sinks (and thus these sinks would be ordered with an index of $p + 1$ or greater) that are the only sinks an unmarked source has a directed path to, leading to a violation of Property (2).

We now show how to rectify the problem by modifying algorithm ST so that it obtains an ordering of sources and sinks that satisfies Properties (1)–(3). The corrected algorithm is called STCORRECT and is displayed in Figure 4. It is identical to ST except for the following modest changes. A boolean variable *sinknotfound* is added that is true if the search from a source node has not yet encountered an unmarked sink. Further, in line 11 the search continues until all sources are marked (as opposed to ST where the search continues until all sinks are marked). At the start of a search from an unmarked source *sinknotfound* is set to true (line 13a). Within the procedure SEARCH, if an unmarked sink is found then $w := x$, and the boolean variable *sinknotfound* is set to false (lines 5, 5a, 5b). This has the effect of stopping the search (in line 7a) as soon as an unmarked sink is found in a search from an unmarked source node. We now prove the correctness of algorithm STCORRECT.

THEOREM 1 *The algorithm STCORRECT finds an ordering of sources and sinks satisfying Properties (1)–(3) in $\mathcal{O}(|A|)$ time.*

Proof:
Observe that in line 7 an arc is examined no more than once proving that the algorithm runs in time proportional to the number of arcs in the acyclic digraph. We now show a very useful property of algorithm STCORRECT that will be invaluable in our proof.

LEMMA 2 *All nodes marked by STCORRECT have a directed path to some sink node in $w(1), \ldots, w(p)$.*

Proof:
We show this by induction. Since the digraph is acyclic, the first time STCORRECT searches from an unmarked source it marks all nodes on a unique path to an unmarked sink node. At that point the search initiated from the unmarked source stops, and the unmarked source is set to $v(1)$ and the unmarked sink just found is set to $w(1)$. Observe that all marked nodes have a directed path to $w(1)$. Consider what happens at any later point in the algorithm when the search is initiated from an unmarked source. The search marks nodes along a path until it encounters a marked node x (in which case we do not search from x but the search from the unmarked source continues) or encounters an unmarked

```
1.  algorithm STCORRECT: **begin**
2.      procedure SEARCH($x$);
3.          **if** $x$ is unmarked **then**
4.              **begin**
5.                  **if** $x$ is a sink **then begin**
5a.                     $w := x$;
5b.                     *sinknotfound*:=false;
5c.                 **end**;
6.                  mark $x$;
7.                  **for** each $y$ such that $(x, y)$ is an arc **do**
7a.                     if (*sinknotfound*) **do** SEARCH($y$);
8.              **end** SEARCH
9.      initialize all nodes to be unmarked;
10.     $i := 0$;
11.     **while** some source is unmarked **do begin**
12.         choose some unmarked source $v$;
13.         $w := 0$;
13a.        *sinknotfound*:=true;
14.         SEARCH($v$);
15.         **if** $w \neq 0$ **then begin**
16.             $i := i + 1$;
17.             $v(i) := v$;
18.             $w(i) := w$;
19.     **end end**
20.     $p := i$;
21. **end** STCORRECT;
```

Figure 4. A "corrected" linear time algorithm to find an ordering of sources and sinks that satisfies Properties (1)–(3).

sink (in which case the search stops). Assume the inductive argument is true at the conclusion of the search from the previous unmarked source in the algorithm. By induction the marked node x has a path to a sink in $w(1), \ldots, w(p)$, thus all nodes in the path to x have a path to a sink in $w(1), \ldots, w(p)$. In the case the procedure encounters an unmarked sink all nodes on the path to the unmarked sink are marked in the search from the unmarked source, and since the unmarked source and sink are now marked and added as $v(i)$ and $w(i)$, with $i \leq p$, the marked nodes on the path from $v(i)$ to $w(i)$ have a path to a sink in $w(1), \ldots, w(p)$. □

Consider the search from any unmarked source. The search successfully finds a directed path from the unmarked source to an unmarked sink (and these are added as $v(i)$ and $w(i)$ with $i \leq p$), or it fails to find a path to an unmarked sink (in which case the source has an index $i > p$). Failure occurs only if there is a marked node on every path from the unmarked source to every unmarked sink. By Lemma 2 these marked nodes have a path to some sink in $w(1), \ldots, w(p)$, proving that the ordering STCORRECT provides satisfies Property (2).

Suppose the ordering STCORRECT provides does not satisfy Property (3). Then there is an unmarked sink node with no directed path from any source in $v(1), \ldots, v(p)$. The unmarked sink node must then have a path from some source $v(j)$, for $p+1 \leq j \leq s$. Consider this path, and consider the last marked node y on this path to the unmarked sink. Node y could not have been marked by any source $v(i)$, with $p + 1 \leq i \leq s$. If it had been, then the search from $v(i)$ would have found the unmarked sink and both the source $v(i)$ and the unmarked sink would have been added to the ordering with an index less than or equal to p. But that means that y must have been marked in the search from one of the sources in $v(1), \ldots, v(p)$. Consequently, there is a directed path from a source in $v(1), \ldots, v(p)$ to the unmarked sink yielding a contradiction to our assumption. □

References

[1] K. P. Eswaran and R. E. Tarjan, *Augmentation problems*, SIAM Journal on Computing, 5 (1976), pp. 653–665.