On testing single connectedness in directed graphs and some related problems

Department of Computer Science and Automation, Technische Universität Ilmenau, 98693 Ilmenau, Germany

Martin Dietzfelbinger, Raed Jaberi

Abstract

Let $G = (V, E)$ be a directed graph with n vertices and m edges. The graph G is called singly-connected if for each pair of vertices $v, w \in V$ there is at most one simple path from v to w in G. Buchsbaum and Carlisle (1993) gave an algorithm for testing whether G is singly-connected in $O(n^2)$ time. In this paper we describe a refined version of this algorithm with running time $O(s \cdot t + m)$, where s and t are the number of sources and sinks, respectively, in the reduced graph G^r obtained by first contracting each strongly connected component of G into one vertex and then eliminating vertices of indegree or outdegree 1 by a contraction operation. Moreover, we show that the problem of finding a minimum cardinality edge subset $C \subseteq E$ (respectively, vertex subset $F \subseteq V$) whose removal from G leaves a singly-connected graph is NP-hard.

Keywords: Algorithms, depth first search, unique paths, directed graphs, connectivity, NP-complete

1. Introduction

Let $G = (V, E)$ be a directed graph with n vertices and m edges. The graph G is called *singly-connected* if for every pair of vertices $v, w \in V$ there is at most one simple path from v to w in G. The problem of testing whether or not G is singly-connected was introduced by Cormen et al. in [3, Ex. 23.3-10] and [4, Ex. 22.3-13]. In [1, 2], Buchsbaum and Carlisle

presented an algorithm for solving this problem in $O(n^2)$ time. Khuller described in [5] a similar approach for solving the same problem in $O(n^2)$ time. Karlin [7] (also see [6]) also presented a simple $O(n^2)$ algorithm for solving the problem.

Let $G = (V, E)$ be a directed graph. By $G^c = (V^c, E^c)$ we denote the directed graph which is obtained by contracting every strongly connected component of G into a single vertex. The algorithms from [1] and [5] are based on reducing the problem on G to the same problem on the acyclic graph G^c. We use G^r to denote the graph obtained from G^c by eliminating all vertices of indegree or outdegree 1 by contraction operations. A vertex x of G^r is called a source if its indegree is 0 and a vertex y of G^r is called a sink if its outdegree is 0. By s and t we denote the number of sources and sinks in G^r, respectively. In this paper we improve the running time of the algorithms from [1],[5] from $O(n^2)$ to $O(s \cdot t + m)$. We also give an example for a graph where $s \cdot t$ is much bigger than m. The question posed by Khuller [5] whether the problem of testing single connectedness in directed graphs is solvable in linear time remains open.

As mentioned in [2], it is clear that a singly-connected graph can be obtained from a directed graph which is not singly-connected by removing edges, but the property of singly-connectivity can be ruined by adding edges. We also show that the problem of finding a minimum cardinality edge subset $C \subseteq E$ (respectively, vertex subset $F \subseteq V$) whose removal from G leaves a singly-connected graph is NP-hard.

Papers [1] and [5] show the existence of a procedure which has the following behavior:

Procedure 1.1.
Input: A directed graph $G = (V, E)$.
Output:
    "not singly-connected"
or
    "Test whether G^c is singly-connected".

Procedure 1.1 without testing of G^c has running time $O(m)$. Paper [1] shows that the acyclic graph G^c is singly-connected if and only if for every vertex $w \in V^c$ the vertex set of the DFS tree rooted at w has neither cross edges nor forward edges. Testing single connectedness of G^c in this way takes $O(n^2)$ time [1, 5] since there are n calls to DFS, one for each vertex of G^c, and one can stop as soon as a forward edge or cross edge appears.

The procedure leads to the situation that we only have to consider acyclic graphs. We give another reduction to be applied after Procedure 1.1 (which can be carried out in time $O(m)$) to a reduced graph G^r, so that G is singly-connected if and only if G^r is. In G^r no vertex has indegree or outdegree 1.

2. Improved handling of acyclic graphs

Assume that the input graph $G = (V, E)$ is acyclic. We propose two types of improvement over the algorithms in [1, 5] for acyclic graphs:

1. eliminating vertices with indegree or outdegree 1.

2. starting DFS only from sources.

In the first step, called preprocessing step, we modify G as follows. We consider the vertices in bottom-up order. For each vertex $v \in V$ with indegree 1 we shrink u, v, where $(u, v) \in E$, by replacing each edge $(v, w) \in E$ by (u, w) and removing the vertex v and the edge (u, v) from G (see Figure 1 for one such contraction). Of course, if v has outdegree 0, it can simply be

(a) G^c (b)

Figure 1: (a) $G = (V, E)$, vertex 2 has indegree 1. (b) The remaining graph after contracting edge $(1, 2)$.

deleted. Let G' be the resulting graph. To G' we apply a similar procedure top down to eliminate all nodes of outdegree 1. The resulting graph is called G^r.

Lemma 2.1.

(a) G^r can be computed in time $O(m)$.

(b) G is singly-connected if and only if G^r is singly-connected.

Proof (a) We represent the acyclic graph G by adjacency lists in which each vertex $v \in V$ has a circular doubly linked list L_v containing all the successors of v in G. Eliminate all vertices of indegree 1, as follows:

   (i) Treat vertices v "bottom up" (in the partial order given by G) as follows: If v has indegree 1, merge v with its predecessor u by linking L_v into L_u in constant time and deleting v. It is clear this can be done in time $O(m)$. (Actually, the merging itself takes time $O(n)$, only finding vertices with indegree 1 needs $O(m)$ time.)

   (ii) Eliminate all vertices of outdegree 1. This is done exactly as in (i), just using the reversed graph of G'.

(b) Each step preserves the property of single connectedness.   □
Note that if multiple edges arise in the preprocessing step, then G is not singly-connected. Check once at the very end if G^r has multiple edges. If so, return " not singly-connected".

   From here on we only consider the reduced graph G^r, in which all non-sources have indegree at least 2 and all non-sinks have outdegree at least 2. In the second step we perform a DFS only for the sources of G^r. The correctness of this step is based on the following lemma.

Lemma 2.2. *Let $G = (V, E)$ be a directed acyclic graph such that G does not have any multiple edges. Then G is singly-connected if and only if for every source $v \in V$ the vertex set of the DFS tree T_v with root v has only tree edges.*

Proof "⟸": Assume that G is not singly-connected. Then by definition, there are two vertices $v, w \in V$ such that there exist at least two simple paths p_1, p_2 from v to w in G. Then there is a vertex u that lies on both paths p_1, p_2 and has two different incoming edges in $p_1 \cup p_2$. There is some source s such that there is a path from s to v in G. Hence the vertex u and all the vertices on $p_1 \cup p_2$ are in T_s. Of course, it is impossible that both edges entering u are tree edges.
"⟹": Any forward or cross edge in a DFS from any vertex immediately proves that G is not singly-connected (see [1]).   □

Theorem 2.3. *It can be tested in $O(s \cdot t + m)$ time if G^r is singly-connected.*

Proof As we have ssen, the transformation from G to G^r takes time $O(m)$. In G^r, each DFS tree T has at most $2t - 1$ vertices since each vertex which

is not a leaf in T has outdegree at least 2 and all leaves of T are sinks so there cannot be more than t leaves. Each DFS requires $\Theta(t)$ time because the algorithm stops as soon as a cross edge or a forward edge appears. The total number of DFS-trees is at most s. Therefore, the total running time is $O(s \cdot t + m)$. $\qquad\square$

One may ask whether there are singly-connected graphs in which $s \cdot t$ is much bigger than m and in which our algorithm takes much longer than $O(m)$ time. Actually, a well known graph family, the butterfly graphs (or FFT graphs) B_d (see, e.g., [9]), give an example. They have $n = 2^d(d + 1)$ vertices and $m = 2^{d+1}d$ edges, are singly-connected, and we have $s = t = 2^d$ and hence $s \cdot t = t^2 = 2^{2d} \gg m$.

It is well known that the butterfly graph is singly-connected (see Figure 2). Actually, every source is connected to every sink via a unique path. This implies that all DFS calls on sources need time $\Theta(t)$, hence the total time is $\Theta(t^2)$, much bigger than m. (but still smaller than $n^2 = 2^{2d}(d + 1)^2$).

Figure 2: Butterfly graph B_3.

On Butterfly graphs our algorithm takes much more than $O(m)$ time. It remains open to find an algorithm that breaks the $O(s \cdot t)$ bound.

3. Optimization problems related to uni-connectivity

In this section we study two optimization problems related to single connectivity. The first problem is defined as follows. Given a directed graph

$G = (V, E)$, find an edge set $C \subseteq E$ of minimum size such that the directed graph $(V, E \setminus C)$ is singly-connected. This problem is denoted by ESC. The second problem is defined as follows. Given a directed graph $G = (V, E)$, find a minimum cardinality vertex set $F \subseteq V$ such that the directed graph $G \setminus F$ obtained from G by removing all the vertices in F and their incident edges is singly-connected. We denote this problem by VSC. We show that VSC and ESC are NP-hard by reducing the vertex cover problem to each of them.

The decision version of the vertex cover problem is NP-complete [8]. It is defined as follows. Given an undirected graph $G = (V, E)$ and an integer l. Is there a vertex set $U \subseteq V$ with $|U| \leq l$ such that for every edge $e = (v, w) \in E$ we have $\{v, w\} \cap U \neq \emptyset$?

We define the decision version of ESC as follows: Given a directed graph $G = (V, E)$ and an integer r. Does there exist an edge set $C \subseteq E$ of size at most r such that the directed graph $(V, E \setminus C)$ is singly-connected?

Lemma 3.1. *The decision version of ESC is NP-complete.*

Proof: It is obvious that ESC is in NP. Let $(G = (V, E), l)$ be an instance of the vertex cover problem. We construct an instance $(G' = (V', E'), r)$ of ESC as follows. For each vertex $v \in V$, we add two new vertices v', v'' to V' and a directed edge (v', v'') to E'. Furthermore, for every undirected edge $e = (v, w) \in E$ we add two vertices e', e'' to V' and four directed edges $(e', v'), (e', w'), (v'', e''), (w'', e'')$ to E'. An example is illustrated in Figure 3. Clearly, the directed graph G' has $2|V| + 2|E|$ vertices and $|V| + 4|E|$ edges. Therefore, G' can be constructed from G in polynomial time. Now we prove that G has a vertex cover of size at most l if and only if G' has an edge set $C \subseteq E'$ of size at most $r = l$ such that $(V', E' \setminus C)$ is singly-connected.
"$\Rightarrow$": Let U be a vertex cover of G such that $|U| \leq l$. Let $C = \{(v', v'') \mid (v', v'') \in E'$ and $v \in U\}$. Obviously, $|C| \leq l$. For every pair of vertices $e', e'' \in V'$ which correspond to undirected edge $e = (u, w)$ in G, there exist two simple paths $(e', u', u'', e''), (e', w', w'', e'')$ from e' to e'' in G'. Since $\{u, w\} \cap U \neq \emptyset$, we have $\{(u', u''), (w', w'')\} \cap C \neq \emptyset$. Therefore, there is at most one simple path from e' to e'' in $(V', E' \setminus C)$. Moreover, for any distinct vertices x, y with $x \neq e'$ or $y \neq e''$, there is at most one simple path from x to y in G'. Consequently, the directed graph $(V', E' \setminus C)$ is singly-connected.
"$\Leftarrow$": Let C be a subset of E' with $|C| \leq l$ such that $(V', E' \setminus C)$ is singly-connected. Let $U = \{v \mid \{(e', v'), (v', v''), (v'', e'')\} \cap C \neq \emptyset$ and $v \in V\}$. It is easy to see that $|U| \leq l$. Assume for a contradiction that U is not a vertex cover of G. Then there are two vertices u, w in G such that $\{u, w\} \cap U = \emptyset$. This implies that $\{(e', u'), (u', u''), (u'', e''), (e', w'), (w', w''), (w'', e'')\} \cap C =$

(a) $G = (V, E)$, $U = \{v_1, v_2\}$.

(b) $G' = (V', E')$, $C = \{(v_1', v_1''), (v_2', v_2'')\}$, the graph $(V', E' \setminus C)$ is singly-connected.

Figure 3: Reducing vertex cover to ESC.

$\emptyset$. Thus, there are two simple paths (e', u', u'', e'') and (e', w', w'', e'') from e' to e'' in the directed graph $(V', E' \setminus C)$, a contradiction.

Now we define the decision version of VSC. Given a directed graph $G = (V, E)$ and an integer r. Is there a vertex set $F \subseteq V$ of size at most r such that the directed graph $G \setminus F$ is singly-connected. We have the following.

Lemma 3.2. *The decision version of VSC is NP-complete.*

Proof: The proof is similar to the proof of Lemma 3.1 (see Figure 4).

4. Open Problems

We leave as an open problem whether Karlin's solution [7] (also see [6]) for the decision problem can be improved. Another open problem is whether there are approximation algorithms for the problems described in Section 3.

References

[1] A.L. Buchsbaum, M.C. Carlisle, Determining uni-connectivity in directed graphs, Information Processing Letters 48(1)(1993) 9–12.

[2] A.L. Buchsbaum, M.C. Carlisle, Determining single-connectivity in directed graphs, Research Report CS-TR-390-92, 1992.

(a) $G = (V, E)$, $U = \{v_1, v_2\}$ is a vertex cover of G.

(b) $G' = (V', E')$, $F = \{v_1', v_2'\}$, the graph $G' \setminus F$ is singly-connected.

Figure 4: Reducing vertex cover to VSC.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, The MIT Electrical Engineering and Computer Science Series, MIT Press, Cambridge, MA, 1991.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, third edition, 2009.

[5] S. Khuller, An $O(|V|^2)$ algorithm for single connectedness, Information Processing Letters, 72(3–4)(1999) 105–107.

[6] S. Khuller, Addendum to "An $O(|V|^2)$ algorithm for single connectedness", Information Processing Letters, 74(5–6) (2000) 263.

[7] A. Karlin, Solution to homework 7, CS 421, Winter 1995, Department of Computer Science, University of Washington, 1995.

[8] R.M. Karp, Reducibility Among Combinatorial Problems, Complexity of Computer Computations, 1972, 85–103.

[9] F. Thomson Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Morgan Kaufmann, San Mateo, CA, 1992, page 441.