ELSEVIER

Discrete Applied Mathematics 69 (1996) 281–289

DISCRETE APPLIED MATHEMATICS

On strongly connected digraphs with bounded cycle length

Samir Khuller [a,1], Balaji Raghavachari [b,2], Neal Young [c,*]

[a] *Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

[b] *Department of Computer Science, The University of Texas at Dallas, Box 830688, Richardson, TX 75083, USA*

[c] *Department of Computer Science, Dartmouth College, Hanover, NH 03755, USA*

Received 17 November 1994; revised 28 August 1995

Abstract

Given a directed graph $G = (V,E)$, a natural problem is to choose a minimum number of the edges in E such that, for any two vertices u and v, if there is a path from u to v in E, then there is a path from u to v among the chosen edges. We show that in graphs having no directed cycle with more than three edges, this problem is equivalent to Maximum Bipartite Matching. This leads to a small improvement in the performance guarantee of the previous best approximation algorithm for the general problem.

1. Introduction

Let $G = (V,E)$ be a directed graph. The minimum equivalent graph (MEG) problem is the following: find a smallest subset $S \subseteq E$ of the edges such that, for any two vertices u and v, if there is a path from u to v in E then there is a path from u to v using only edges in S. The problem is NP-hard [3]. A c-approximate solution is a subset of edges providing the necessary paths of size at most c times the minimum. A c-approximation algorithm is a polynomial-time algorithm guaranteeing a c-approximate solution.

Moyles and Thompson [8] observed that any solution to the MEG problem decomposes into solutions for each strongly connected component and a solution for the component graph (the graph obtained by contracting each strongly connected component). Thus, the problem reduces in linear time to two cases: the graph is either acyclic or strongly connected. If the graph is acyclic, the MEG problem is equivalent to the *transitive reduction* problem, which was shown by Aho et al. to be equivalent

* Corresponding author. Department of Computer Science, Dartmouth College, 6211 Sudikoff Laboratories, Hanover, NH 03755-3510, USA. E-mail: ney@dartmouth.edu. Part of this research was done while at School of ORIE, Cornell University, Ithaca NY 14853 and supported by Éva Tardos' NSF PYI grant DDM-9157199.

[1] Research supported by NSF Research Initiation Award CCR-9307462.

[2] Research supported by NSF grant CCR-9409625.

SSDI 0166-218X(95)00105-0

to transitive closure [1]. Thus, we assume the graph is strongly connected, so that the problem is to find a small subset of the edges preserving the strong connectivity. We refer to this problem as *the strongly connected spanning subgraph (SCSS)* problem.

The only known c-approximation algorithm for any $c < 2$ works by repeatedly contracting cycles [6]. Each cycle contracted is either a longest cycle in the current graph, or has length at least some constant k. The set of contracted edges yields the set S. As k grows, the performance guarantee of this algorithm rapidly tends to $\pi^2/6 \approx 1.64$.

A natural modification is to solve the problem optimally as soon as the maximum cycle length in the current graph drops below some threshold. The problem remains NP-hard even when the maximum cycle length is five, but we conjectured in [6] that it was solvable in polynomial time if the maximum cycle length is three. We use $SCSS_3$ to denote the SCSS problem with this restriction. In this paper we confirm the conjecture:

Theorem 1.1. *The $SCCS_3$ problem in n-vertex digraphs reduces in $O(n^2)$ time to Minimum Bipartite Edge Cover.*

This gives an $O(n^2 + m\sqrt{n})$-time algorithm for the $SCCS_3$ problem, since Minimum Bipartite Edge Cover is trivially equivalent to Maximum Bipartite Matching [9], which can be solved in $O(m\sqrt{n})$ time [4]. Modifying the cycle-contraction algorithm correspondingly reduces its performance guarantee by 1/36.

Corollary 1.2. *For any $c > \pi^2/6 - 1/36 \approx 1.61$, there exists a c-approximation algorithm for the MEG problem.*

(For graphs with bounded cycle size, a slightly stronger performance guarantee can be shown as described at the end of Section 4.) This corollary follows from a straightforward modification to the analysis in [6] of the algorithms described above.

Here is an overview of the reduction of $SCSS_3$ to Edge Cover. We classify each edge as either *necessary* (removal of the edge leaves the graph not strongly connected) or *redundant* (otherwise). It turns out that any SCSS consists of the necessary edges together with a set of redundant edges sufficient to ensure that each necessary edge lies on some cycle in the SCSS. We characterize the manner in which redundant edges can lie on such cycles – specifically, each cycle can have at most one redundant edge and each redundant edge lies on exactly one cycle (and thus "provides a cycle" for at most two necessary edges). This allows the reduction.

A natural question is whether $SCSS_3$ is fundamentally simpler than Bipartite Edge Cover. In Section 3 we show it is not:

Theorem 1.3. *Minimum Bipartite Edge Cover reduces in linear time to $SCSS_3$.*

Comparison to undirected graphs: When the maximum cycle length is three, the SCSS problem is as hard as Bipartite Matching. When it is five, the problem is NP-hard.

When it is seventeen, the problem is MAX-SNP-hard [6]. The latter precludes even a polynomial-time approximation scheme unless P=NP. Thus, digraphs with bounded cycle length can have rich connectivity structure.

This highlights the fundamental difference between connectivity in directed and undirected graphs. The analogous problem in undirected graphs is to find a minimum-size subset of edges preserving 2-edge connectivity. This problem (and many others that are NP-hard in general) can be solved optimally in polynomial time for graphs with bounded cycle length [2].

Other related work: Moyles and Thompson [8] gave an exponential-time algorithm for the MEG problem; Hsu [5] gave a polynomial-time algorithm for the acyclic case.

Contents: The body of the paper is organized as follows. Section 2 contains the reduction of $SCSS_3$ to Edge Cover (proving Theorem 1.1). Section 3 notes that Edge Cover reduces in linear time to $SCSS_3$, so that (with respect to quadratic time reductions) the problems are equivalent. Section 4 describes the application: the improved approximation algorithm for the general MEG problem.

2. Reduction: $SCSS_3$ to Edge Cover

Let $G = (V,E)$ be a strongly connected digraph with maximum cycle length 3 or less. Assume that G has at least four vertices, none of which are cut vertices (that is, vertices whose removal disconnects the underlying undirected graph). This is without loss of generality, because by standard techniques, in $O(n+m)$ time, the cut vertices can be found and the graph partitioned into 2-connected components. Clearly, a c-approximation for each component yields a c-approximation for G.

Definition 1. An edge is *redundant* if deleting the edge from G leaves a strongly connected graph. Otherwise it is *necessary*.

An edge (u,v) is *unsatisfied* if there is no path from v to u consisting of necessary edges.

A redundant edge e *provides a cycle* for an unsatisfied edge (u,v) if there is a path from v to u consisting of necessary edges and e.

Here is an outline of the reduction. Since the necessary edges are in any SCSS, the question is which redundant edges to add. It turns out that each redundant edge lies on exactly one cycle (Lemma 2.1) and thus provides a cycle for at most two unsatisfied edges. Further, no cycle has more than one redundant edge, so that a set of edges is an SCSS if and only if it contains the necessary edges and, for each unsatisfied edge e, a redundant edge providing a cycle for e (Lemma 2.2).

We construct an equivalent instance of Edge Cover – an undirected graph G' that has a vertex w' for each unsatisfied edge w in G and an edge r' for each redundant

edge r in G, where r' is incident to w' if r provides a cycle for w. It turns out that the graph G' is acyclic (in the undirected sense) and thus bipartite (Lemma 2.3).

Finally, the redundant edges and the graph G' can be computed in $O(n^2)$ time (Lemmas 2.4 and 2.5).

2.1. Reduction to bipartite Edge Cover

Here is the first essential fact.

Lemma 2.1. *Each redundant edge lies on exactly one cycle in G.*

Proof. We use here the assumption that G has at least four vertices, none of which are cut vertices.

Since G is strongly connected, each edge lies on at least one cycle. Suppose for contradiction that some redundant edge (u,v) lies on more than one cycle. There are (at least) two distinct paths from v to u. At least one of the paths is of length two. Denote this path (v,x,u).

Since edge (u,v) is redundant, there is a path P_{uv} from u to v other than edge (u,v). P_{uv} must contain x, for otherwise P_{uv} and the path (v,x,u) would form a cycle of more than three edges.

If the edge (v,u) is present in G, then P_{uv} is of length two (as it forms a cycle with (v,u)) and hence is the path (u,x,v). Thus, in this case, all six possible edges are present between the three vertices u, x, and v. Let V_u denote the vertices reachable from u without going through v or x. Define V_v and V_x similarly. Using the strong connectivity of the graph and its lack of long cycles, one can easily show that these sets are disjoint and have no edges between them. Thus, either at least one of u, x, or v is a cut vertex or the graph has only these three vertices. This contradicts our assumption about G.

Thus, the edge (v,u) is not present and there exists a path distinct from (v,x,u) and (v,u) from v to u. Denote this path, which must be of length two, by (v,y,u). The path P_{uv} must contain y for the same reason P_{uv} contains x. Thus, there is a path Q, without loss of generality from y to x, that does not contain u or v (see Fig. 1). This is a contradiction, because the edges (x,u), (u,v), and (v,y) would form a cycle of length at least four with the path Q. □

Lemma 2.2. *A set of edges is an SCSS iff it contains the necessary edges and, for each unsatisfied edge e, some redundant edge providing a cycle for e.*

Proof. The "if" direction is straightforward. To see the converse, first note that each cycle in G contains at most one redundant edge (otherwise each redundant edge would lie on more than one cycle, violating Lemma 2.1). In fact, this also implies that unsatisfied edges are not redundant, otherwise we would have a cycle with two redundant edges.

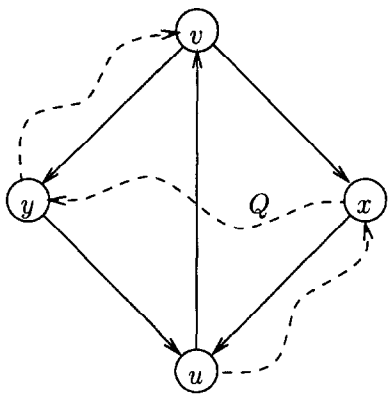

Fig. 1. No cycle has two redundant edges.

Since the SCSS strongly connects the graph, any unsatisfied edge must form a cycle with the edges in the SCSS. By the preceding observation, this cycle has one redundant edge.

By Lemma 2.2, the problem reduces to identifying a smallest set of redundant edges such that for each unsatisfied edge e in G, some redundant edge in the set provides a cycle for e. By Lemma 2.1, each redundant edge provides a cycle for at most two unsatisfied edges.

Build a graph G' whose vertices correspond to the unsatisfied edges. For each redundant edge e, if e provides a cycle for two unsatisfied edges, add an edge between the two corresponding vertices; if e provides a cycle for one unsatisfied edge, add a self-loop at the corresponding vertex.

By the above discussion, a set of edges in G' forms an edge cover if and only if the corresponding set of redundant edges in the original graph, together with the necessary edges, form an SCSS.

So far, we have reduced our problem to Minimum Edge Cover. The next lemma shows that the reduction is in fact to Minimum *Bipartite* Edge Cover.

Lemma 2.3. *G' is bipartite.*

Proof. We will show that the unsatisfied edges in G can be two-colored so that no adjacent edges have the same color. This gives the result as follows: color each vertex in G' with the color of its corresponding unsatisfied edge in G; by Lemma 2.1, vertices that share an edge in G' correspond to adjacent (and therefore differently colored) unsatisfied edges in G. Thus, no edge in G' has two vertices of the same color.

Assume for contradiction that the unsatisfied edges of G cannot be legally two-colored. Then some set C of the edges corresponds to an (odd) cycle in the underlying undirected graph. Let edge (u, v) be one of the edges in C. We will show that there is an alternate path from u to v, so that (u, v) is redundant. Since unsatisfied edges are not redundant, this is a contradiction.

It suffices to show that, for each edge (a,b) on C, there is a path from b to a that does not use (u,v). Suppose the return path for (a,b) does contain (u,v). The return path must have length two, so either $u=b$ or $v=a$.

We consider only the first case; the other is similar. Since $u=b$, this case reduces to finding a path from u to a that does not use (u,v), given that (u,v,a) is a return path for (a,u) and that (u,v) and (a,u) are unsatisfied and therefore necessary.

Suppose edge (v,a) was necessary. Then cycle (a,u,v,a) would consist of necessary edges, so none of its edges would be unsatisfied. Thus, (v,a) is redundant. Let P_{va} be an alternate path from v to a. P_{va} must go through u, for otherwise P_{va} and the edges (a,u) and (u,v) would form a cycle of length more than three. Thus, P_{va} contains a path from u to a that does not go through v. This portion of P_{va} is the desired path. □

2.2. Complexity

To finish the proof of Theorem 1.1, we show that the reduction can be computed in $O(n^2)$ time.

Lemma 2.4. *Classifying the edges as redundant or necessary requires* $O(n^2)$ *time.*

Proof. Let G have n vertices and m edges. Fix a root r and find an incoming and an outgoing branching (spanning trees rooted at r with all edges directed towards or, respectively, away from r). This can be done in $O(n+m)$ time using depth-first search. Let B be the union of the sets of edges in the two branchings. There are at most $2n-2$ edges in B and the edges not in B are redundant. This leaves $O(n)$ edges to be classified. Classify them using $O(n)$ time per edge as follows.

Consider an edge (u,v). Enumerate the other vertices to check for alternate paths from u to v of length two. If such a path exists, the edge is redundant. Otherwise, check for the edge (v,u). If it exists, the edge (u,v) is necessary, because any alternate path from u to v would have to have length two and all such paths have been checked. Otherwise, check all return paths of length two. If at least two of these paths exist, the edge is necessary by Lemma 2.1.

Otherwise, (u,v) has a unique return path (v,w,u). If an alternate path from u to v exists, then it must use w (else we get a cycle of length at least four). Because of the edge (w,u), the path from u to w can have length at most two. Similarly, the path from w to v can have length at most two. Thus, w and the existence of the paths from u to w and w to v can be determined by enumeration in $O(n)$ time. □

Lemma 2.5. *Building the graph* G' *requires* $O(n^2)$ *time.*

Proof. Once the edges have been classified as redundant or necessary, the unsatisfied edges and the return paths for the redundant edges can be identified in $O(n^2)$ time as follows. Each edge (u,v) is redundant, or necessary but *not* unsatisfied, if and only if

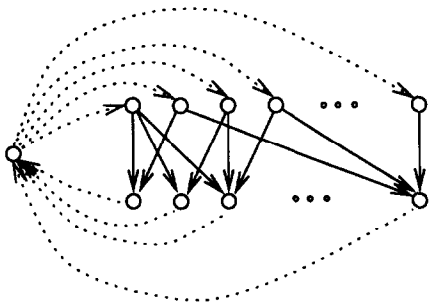

Fig. 2. Bipartite Edge Cover reduces to $SCSS_3$.

it has a return path of one or two necessary edges. Enumerate each path of one or two necessary edges; let u and v be the first and last vertices on the path; if there is an edge (v,u), then either note its return path (if it is redundant) or note that it is not unsatisfied (if it is necessary). There are $O(n^2)$ such paths, since there are $O(n)$ necessary edges. □

This proves Theorem 1.1.

3. Reduction: Edge Cover to $SCSS_3$

The proof of Theorem 1.3 (Minimum Bipartite Edge Cover reduces in linear time to $SCSS_3$) is somewhat simpler:

Proof of Theorem 1.3. Given an undirected bipartite graph, construct a directed graph as shown in Fig. 2. Direct all the edges from the first part to the second part. Add a root vertex with edges to each vertex in the first part and from each vertex in the second part. Any edge cover in the original graph (together with the added edges) yields an SCSS and vice versa. □

4. Application to the general MEG problem

Here we describe the improvement to the approximation algorithm for the general MEG problem in [6]. As usual, without loss of generality, assume the graph is strongly connected. The algorithm in [6] works by repeatedly contracting cycles. Each cycle contracted is either a longest cycle in the current graph, or has length at least some constant k. The set of contracted edges yields the set S. As k grows, the performance guarantee of the algorithm tends rapidly to $\pi^2/6 \approx 1.64$.

Assume $k \geqslant 4$. Modify the algorithm so that as soon as the current graph has maximum cycle size three or less, it solves the problem optimally (using Theorem 1.1) and returns the edges in the solution for the current graph together with the edges on previously contracted cycles.

To *contract* an edge is to identify its endpoints in the graph as a single vertex; to contract a cycle is to identify all vertices on the cycle. We use the following result from [6]:

Proposition 4.1 (Khuller et al. [6]). *If the maximum cycle length in an n-vertex graph is ℓ, then any SCSS has at least $(n-1)\ell/(\ell-1)$ edges.*

The proof is that any strongly connected graph can be contracted to a single vertex by repeatedly contracting cycles whose edges are in the SCSS; the ratio of edges contracted to vertices lost when one of these cycles is contracted is at least $\ell/(\ell-1)$.

Proof of Corollary 1.2. Initially, let the graph have n vertices. Assume n_i vertices remain in the contracted graph after contracting cycles with i or more edges ($i = k, k-1, \ldots, 4$). Finally, we get a graph H (with n_4 vertices) that has no cycles of length four or more; the algorithm solves the SCSS problem for H optimally.

How many edges are returned? Let $\mathcal{OPT}(G)$ denote the minimum size of an SCSS of G. In contracting cycles with at least k edges, at most $(k/(k-1))(n-n_k)$ edges are contributed to the solution. For $4 \leqslant i < k$, in contracting cycles with i edges, $(i/(i-1))(n_{i+1}-n_i)$ edges are contributed. The number of edges returned is thus at most

$$\frac{k}{k-1}(n-n_k)+\sum_{i=4}^{k-1}\frac{i}{i-1}(n_{i+1}-n_i)+\mathcal{OPT}(H).$$

A little work shows this is equal to

$$\frac{k}{k-1}(n-1)+\sum_{i=5}^{k}\frac{n_i-1}{(i-1)(i-2)}-\frac{4}{3}(n_4-1)+\mathcal{OPT}(H).$$

Since $\mathcal{OPT}(H) \leqslant 2(n_4-1)$, substituting for n_4 gives the upper bound

$$\frac{k}{k-1}(n-1)+\sum_{i=5}^{k}\frac{n_i-1}{(i-1)(i-2)}+\frac{1}{3}\mathcal{OPT}(H).$$

Clearly, $\mathcal{OPT}(G) > n-1$. For $4 \leqslant i \leqslant k$, when n_i vertices remain, no cycle has more than $i-1$ edges. By Proposition 4.1, any SCSS of the current graph (and therefore any SCSS of G) has at least $(n_i-1)(i-1)/(i-2)$ edges. Also $\mathcal{OPT}(G) \geqslant \mathcal{OPT}(H)$. Using these three facts, the above quantity, divided by $\mathcal{OPT}(G)$, is less than

$$\frac{k}{k-1}+\sum_{i=5}^{k}\frac{1}{(i-1)(i-1)}+\frac{1}{3}$$

$$=\frac{1}{k-1}+\sum_{i=1}^{k-1}\frac{1}{i^2}-\frac{1}{36}.$$

Using the identity (from [7, p.75]) $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$, this is equal to

$$\begin{aligned}
&\frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{k-1} - \sum_{i=k}^{\infty} \frac{1}{i^2} \\
&\leqslant \frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{k-1} - \sum_{i=k}^{\infty} \frac{1}{i(i+1)} \\
&= \frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{k-1} - \frac{1}{k} \\
&= \frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{k(k-1)}.
\end{aligned}$$

Similar to [6], standard techniques can yield more accurate estimates, e.g., $\frac{1}{6}\pi^2 - \frac{1}{36} + (1/2k^2) + O(1/k^3)$. Also following [6], if the graph initially has no cycle longer than ℓ ($\ell \geqslant k$), then the analysis can be generalized to show a performance guarantee of $(k^{-1} - \ell^{-1})/(1 - k^{-1}) + \sum_{i=1}^{k-1} 1/i^2 - \frac{1}{36}$. For instance, in a graph with no cycle longer than 5, the analysis bounds the performance guarantee (when $k = 5$) by 1.396.

Acknowledgements

We thank R. Ravi and Klaus Truemper for helpful discussions. We also thank the referees for useful comments on a preliminary draft of this paper.

References

[1] A.V. Aho, M.R. Garey and J.D. Ullman, The transitive reduction of a directed graph, SIAM J. Comput. 1 (1972) 131–137.

[2] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree-decomposable graphs, J. Algorithms 12 (1991) 308–340.

[3] M.R. Garey and D.S. Johnson, Computers and Intractability: A guide to the Theory of NP-Completeness (W. H. Freeman, New York, 1979).

[4] J.E. Hopcroft and R.M. Karp, An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, SIAM J. Comput. 2 (1973) 225–231.

[5] H.T. Hsu, An algorithm for finding a minimal equivalent graph of a digraph, J. ACM 22 (1975) 11–16.

[6] S. Khuller, B. Raghavachari and N. Young, Approximating the minimum equivalent digraph, SIAM J. Comput. 24 (1995) 859–872.

[7] D.E. Knuth, Fundamental Algorithms (Addison-Wesley, Menlo Park, CA, 1973).

[8] D.M. Moyles and G.L. Thompson, An algorithm for finding the minimum equivalent graph of a digraph, J. ACM 16 (1969) 455–460.

[9] R.Z. Norman and M.O. Rabin, An algorithm for a minimum cover of a graph, Proc. Amer. Math. Soc. 10 (1959) 315–319.