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

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

[5] H. T. Hsu, An algorithm for finding a minimal equivalent graph of a digraph, *Journal of the ACM*, 22 (1), pp. 11–16, (1975).

[6] S. Khuller, B. Raghavachari and N. Young, Approximating the minimum equivalent digraph, to appear in *SIAM J. Comput.*.

[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, *Journal of the ACM*, 16 (3), pp. 455–460, (1969).

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

Clearly $\mathcal{OPT}(G) \geq n$. For $4 \leq i \leq k$, when n_i vertices remain, no cycle has more than $i-1$ edges. By Lemmas 3.1 and 3.2, $\mathcal{OPT}(G) \geq \frac{i-1}{i-2}(n_i - 1)$. Thus the number of edges returned, divided by $\mathcal{OPT}(G)$, is at most

$$\begin{aligned} & \frac{\left(1+\frac{1}{k-1}\right)n}{\mathcal{OPT}(G)} + \sum_{i=5}^{k} \frac{\frac{n_i-1}{(i-1)(i-2)}}{\mathcal{OPT}(G)} + \frac{\mathcal{OPT}(H) - \frac{4}{3}n_4}{\mathcal{OPT}(G)} \\ = \quad & \frac{1}{k-1} + 1 + \sum_{i=4}^{k-1} \frac{1}{i^2} + \frac{\mathcal{OPT}(H)}{\mathcal{OPT}(G)} - \frac{\frac{4}{3}n_4}{\mathcal{OPT}(G)}. \end{aligned}$$

Since $\mathcal{OPT}(H) \leq \mathcal{OPT}(G)$ and $\mathcal{OPT}(G) \leq 2n_4$, this is at most

$$\begin{aligned} c_k &= \frac{1}{k-1} + 1 + \sum_{i=4}^{k-1} \frac{1}{i^2} + \frac{1}{3} \\ &= \frac{1}{k-1} + \sum_{i=1}^{k-1} \frac{1}{i^2} - \frac{1}{36}. \end{aligned}$$

Using the identity (from [7, p.75]) $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$, we get

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

□

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

Acknowledgments. We thank R. Ravi and Klaus Truemper for helpful discussions.

References

[1] A. V. Aho, M. R. Garey and J. D. Ullman, The transitive reduction of a directed graph, *SIAM J. Comput.*, 1 (2), pp. 131–137, (1972).

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

with the corresponding edge in the original graph or, in the case of multi-edges, the single remaining edge is identified with any one of the corresponding edges in the original graph. To contract an edge (u, v) is to contract the pair of vertices u and v. To contract a set S of pairs of vertices in a graph G is to contract the pairs in S in arbitrary order. The contracted graph is denoted by G/S. Contracting an edge is also analogously extended to contracting a set of edges.

Definition 4 *$\mathcal{OPT}(G)$ is the minimum cardinality of any subset of the edges that strongly connects G.*

We first review some simple lemmas proven in [6].

Lemma 3.1 (Cycle Lemma) *For any directed graph G with n vertices, if a longest cycle of G has length $\mathcal{C}$, then*

$$\mathcal{OPT}(G) \geq \frac{\mathcal{C}}{\mathcal{C}-1}(n-1).$$

Lemma 3.2 (Contraction Lemma) *For any directed graph G and set of edges S,*

$$\mathcal{OPT}(G) \geq \mathcal{OPT}(G/S).$$

The algorithm and its analysis are modifications of the those presented in [6]. Fix $k \geq 4$ to be any positive integer.

CONTRACT-CYCLES$_k(G)$ —
1 **for** $i = k, k-1, k-2, ..., 4$
2 **while** the graph contains a cycle with at least i edges
3 Contract the edges on such a cycle.
4 Solve the SCSS$_3$ problem optimally.
5 **return** the edges in the optimal SCSS$_3$ together with the contracted edges.

It is easy to see that this algorithm returns an SCSS if G is a strongly connected graph.

Theorem 3.3 CONTRACT-CYCLES$_k(G)$ *returns at most $c_k \cdot \mathcal{OPT}(G)$ edges, where*

$$c_k \leq \frac{\pi^2}{6} - \frac{1}{36} + \frac{1}{(k-1)k}.$$

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

How many edges are returned? In contracting cycles with at least k edges, at most $\frac{k}{k-1}(n-n_k)$ edges are contributed to the solution. For $4 \leq i < k$, in contracting cycles with i edges, $\frac{i}{i-1}(n_{i+1}-n_i)$ edges are contributed. The number of edges returned is thus at most

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

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

union of the two branchings. Clearly there are at most $2n-2$ edges in B. It is easy to show that B is an SCSS of G and therefore all the edges of G but not in B are redundant edges (for any edge $e=(u,v)$ there is a path from u to v using edges in B). This leaves only $O(n)$ edges that need to be classified. We now show that each of these edges can be classified in $O(n)$ time (the naive method takes $O(m)$ time). Therefore the edges of G can be classified in $O(n^2)$ time.

Consider an edge (u,v). Since G is strongly connected and has no cycles of length greater than three, there must be a path from v to u of length either one or two. **Case 1** — there is an edge (v,u): If there is a path from u to v that does not use the edge (u,v) then this path has length exactly two. It is easy to check the existence of such a path in $O(n)$ time by enumeration. **Case 2** — there is no edge (v,u): If there are two paths of length two from v to u then the edge (u,v) is necessary (as in the proof of Lemma 2.1, the existence of an two u-to-v paths would imply that the graph had a cycle of length more than three). **Case 3** — the graph has a unique path of length two from v to u: Let this path be (v,w,u). In this case, if an alternate u to v path 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 can be determined and the existence of the paths ¿from u to w and w to v can be checked by enumeration in $O(n)$ time.

2.3.2 Running time of the $SCSS_3$ algorithm

We now state the main theorem of our paper.

Theorem 2.4 *Let G be a strongly connected graph with maximum cycle length bounded by three. A strongly connected subgraph of G with minimum cardinality can be computed in $O(n^2+m\sqrt{n})$ time.*

Proof. Classifying edges as necessary and redundant as described above takes $O(n^2)$ time. Using the standard techniques described in Lemma 2.1, the problem can be reduced to subproblems satisfying the conditions of Corollary 2.2. As shown in Lemma 2.3, each subproblem reduces in linear time to bipartite matching. The total number of vertices in the bipartite graphs is at most $2n$ since the total number of necessary edges is at most $2n$. The total number of edges in the bipartite graphs is at most m since the total number of sets $\{N_e\}$ is at most m. Thus, the total time required to solve all of the subproblems is $O(m\sqrt{n})$ [4]. □

3 Applications to the General SCSS Problem

Using our $SCSS_3$ algorithm, we improve the best currently known poly-time approximation algorithm for the general SCSS problem. The performance guarantee of the improved algorithm is arbitrarily close to $\pi^2/6 - 1/36 \approx 1.61$.

Definition of contraction. To *contract* a pair of vertices u,v of a digraph is to replace u and v (and each occurrence of u or v in any edge) by a single new vertex, and to delete any subsequent self-loops and multi-edges. Each edge in the resulting graph is identified

reduces to finding a smallest set of redundant edges R such that every edge in N is satisfied by an edge in R. This yields the set-cover problem described at the beginning of Section 2.2.

By the second part of the corollary, each redundant edge e satisfies at most two necessary edges. Thus, each set in the set-cover problem has size at most two, and the problem reduces to the edge-cover problem [9] in the following graph: the vertices are the elements of N and the edges are the sets $\{N_e : e \text{ redundant}\}$, with sets of size one yielding self-loops. The edge-cover problem is linear-time equivalant to maximum matching; if the graph (minus the self-loops) is bipartite, it is equivalent to maximum bipartite matching.

To finish the analysis, it remains only to show that the resulting matching problem is in fact bipartite.

Lemma 2.3 *The above maximum matching problem is bipartite.*

Proof.

Recall that N is defined to contain those edges not having a return path of necessary edges. Consider the directed graph D induced in G by the set of edges N. We will show that even if the directions of the edges in D are ignored, D is acyclic. This suffices — it implies that the vertices of D can be layered so that every edge of D goes forward one layer; consequently, for each redundant edge e such that $|N_e| = 2$, the set N_e has one edge in an even layer and one edge in an odd layer.

Assume for contradiction that a subset C of the edges of D could be redirected to form a cycle. Pick a directed edge (u, v) in C. We will show that there is a path from u to v that doesn't use the edge (u, v). This implies that (u, v) is redundant, which is a contradition.

It suffices to show that for every edge (a, b) on C, there is a path from b to a that does not use (u, v). (These paths, together with the edge set $C - \{(u, v)\}$, strongly connect the vertices of C.) Let (a, b) be any edge in $C - \{(u, v)\}$. If the return path does not contain edge (u, v), we are done. Otherwise the return path must be of length two.

Suppose that (u, v) is the first edge on the return path. Then the return path is (u, v, a). Since $b = u$, edge (a, b) is (a, u) and we wish to show the existence of a path from u to a not using edge (u, v). The edge (v, a) is redundant, for otherwise edges (u, v) and (a, u) (which are necessary) would have return paths without redundant edges and therefore would not be in N. Since (v, a) is redundant, there is an alternate path P_{va} 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.

If (u, v) is not the first edge on the return path, it must be the second. This case is similar. □

2.3 Implementation of Algorithm

2.3.1 Classifying the edges

We first consider the problem of classifying the edges as redundant or necessary. Let G have n vertices and m edges. Fix a root r, and find an incoming and an outgoing branching from root r. This can be done in $O(m)$ time using depth-first search. Let B be the edges in the

If there is only one path from v to u, then Case 1 holds, so assume otherwise. In this case there is at least one path (v, x, u) of length two. (No longer path can exist, because of the cycle it would form with (u, v).)

By assumption, 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 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). In this case, Case 2 holds.

Otherwise there is at least one more path (v, y, u) of length two. Path P_{uv} contains y for the same reason P_{uv} contains x. Thus, there is a path Q either from x to y or from y to x that does not contain u or v (see Figure 2.)

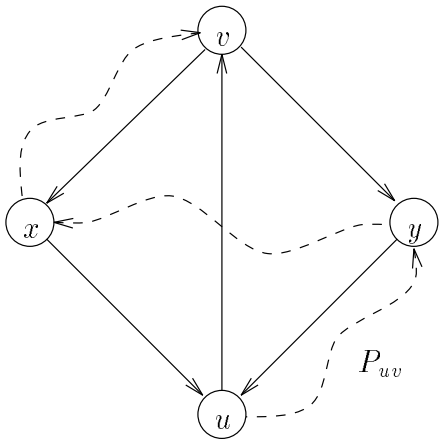

Figure 2: Any redundant edge has at most one return path of length two.

This is a contradiction, because the paths (v, x, u) and (v, y, u) and the edge (u, v) contain the paths (x, u, v, y) and (y, u, v, x), one of which would form a cycle of length at least four with the path Q. □

Corollary 2.2 *If the necessary and redundant edges are known, the $SCSS_3$ problem reduces in linear time to a restricted version such that*

1. *each cycle has at most one redundant edge and*
2. *each redundant edge lies on exactly one cycle.*

Each property follows from Lemma 2.1.

Reduction to bipartite matching. For the remainder of the section we assume that G is a strongly connected graph with maximum cycle length at most three such that every redundant edge has a unique return path. By the first part of the corollary, the set N (defined to contain those edges not having a return path of necessary edges) contains only necessary edges. Also, each return path has at most one redundant edge. Thus, the problem

Definition 3 *Let N denote those edges not having a return path of necessary edges. An edge e* satisfies *an edge f in N if f has a return path consisting of e along with a set of necessary edges. Let N_e denote the set of edges in N that e satisfies.*

Since G is strongly connected, an edge set is an SCSS iff every edge in G has a return path in the set. Since all necessary edges must be in any SCSS, our problem is to find the smallest subset R of redundant edges that provides every edge in N with a return path. In a general graph it may be that no single edge alone can satisfy an edge in N because the return paths may have many redundant edges. But in the case when G has no cycles longer than three, we show that, without loss of generality, N contains only necessary edges (although not necessarily all of them) and each return path has at most one redundant edge. This insight into the structure of the problem gives an $O(n^2)$-time reduction to maximum bipartite matching, yielding a polynomial-time algorithm for the minimum SCSS_3 problem.

2.2 Structure of the SCSS_3 problem

We begin by showing that G decomposes into independent subproblems such that, in each subproblem, (i) each cycle has at most one redundant edge and (ii) each redundant edge is on only one cycle. By (i), the set N contains only necessary edges and each return path can have at most one redundant edge. Thus, the problem reduces to finding a smallest set of redundant edges R such that $N \subseteq \cup_{e \in R} N_e$ — a set-cover problem. By (ii), each redundant edge e satisfies at most two necessary edges, so that each N_e has size at most two. Such set-cover problems are easily reducible to maximum matching.

To finish the analysis, we show that the matching instances that arise for our problem are bipartite. We then give the full algorithm and summarize the results.

Lemma 2.1 *If the necessary and redundant edges are known, the SCSS_3 problem reduces in linear time to a restricted version such that for each redundant edge the return path is unique.*

Proof. We claim that for any redundant edge (u, v), either

1. the return path from v to u is unique, or
2. for some vertex w, G contains directed cycles (u, v, x, u) and (u, x, v, u).

To see that this suffices, note that in Case 2, the vertices u, v, and x are cut vertices in the underlying undirected graph. Let V_u denote the set of vertices reachable from u without going through v or x. Let V_v and V_x be defined analogously. Then these three subsets of vertices partition the vertex set such that no edge in G crosses from one subset into another. It is well-known how to identify all such cut vertices and the resulting biconnected components of the underlying undirected graph in linear time. Each subgraph induced in G by such a component can be solved independently. In each such subgraph, Case 1 holds for every redundant edge.

To prove the claim, suppose (u, v) is redundant. We will show that one of the Cases 1 or 2 holds.

then this would further improve the approximation factor for the MEG problem. However, we suspect that this problem is NP-hard. Proving this would establish the complexity of the SCSS_k problem for all values of k.

2 The SCSS_3 Problem

Let G be a strongly-connected directed graph with maximum cycle length at most three. Our goal is to find a minimum-cardinality SCSS – a smallest subset of the edges that preserves the strong connectivity of G.

As a starting point, note that maximum bipartite matching can easily be reduced to this problem. It is well-known that maximum bipartite matching is linear-time equivalent to the edge-cover problem in bipartite graphs, which is the problem of finding a minimum-cardinality subset of edges incident to all vertices in a bipartite graph [9]. This problem can be easily transformed into our problem by directing all edges in the bipartite graph from the first part to the second part and adding an artificial source vertex with edges *to* each vertex in the first part and *from* each vertex in the second part. See Figure 1. Any SCSS in the modified graph yields consists of the edges adjacent to the source vertex, together with the edges corresponding to some edge cover in the original graph. Conversely, any edge cover in the original graph yields an SCSS in the modified graph. Thus, bipartite matching reduces to the SCSS_3 problem. In the remainder of the section, we show that the SCSS_3 problem reduces to bipartite matching.

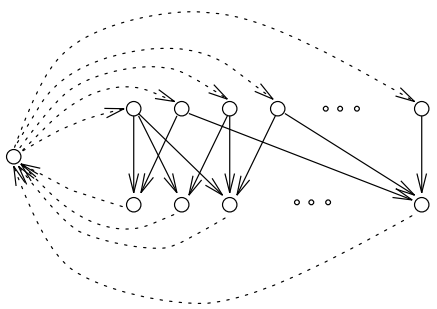

Figure 1: Maximum bipartite matching reduces to SCSS_3.

2.1 Definitions

The following definitions pertain to strongly connected digraphs.

Definition 1 *An edge $e \in E$ is* redundant *if the graph $G' = (V, E - e)$ is strongly connected, otherwise it is* necessary.

Definition 2 *For an edge $(u, v) \in E$, any path from v to u is called a* return path *for (u, v).*

Moyles and Thompson [8] observe this decomposition and give exponential-time algorithms for the restricted problems. Hsu [5] gives a polynomial-time algorithm for the acyclic MEG problem. For acyclic graphs, the MEG problem is equivalent to the *transitive reduction* problem, which is shown by Aho, Garey and Ullman to be equivalent to transitive closure [1]. Thus, the heart of the MEG problem is the minimum SCSS problem.

1.1 The bounded cycle length problem

A natural problem to consider is the SCSS problem when restricted to graphs which have bounded cycle length. In this paper we study the complexity of computing a minimum SCSS in the special case when the input graph is guaranteed to have no cycles greater than k (a fixed constant). We call this the SCSS_k problem. The SCSS_2 problem is trivial. Therefore the problem is interesting only when $k \geq 3$.

It was recently shown by Khuller, Raghavachari and Young [6] that the SCSS_5 problem is NP-hard and that the SCSS_{17} problem is MAX-SNP-hard (precluding the possibility of a polynomial-time approximation scheme, unless P=NP). The strong dependence of the complexity on the cycle length is in marked contrast to the relation of complexity and cycle length in undirected graphs. Undirected graphs with bounded cycle length have bounded tree width, allowing polynomial-time algorithms for many problems that are NP-hard in general, including the minimum 2-edge-connected subgraph problem (the natural analog of the SCSS problem in undirected graphs) [2]. This contrast makes the SCSS_k problem of interest. The problem would be completely characterized if it can be shown that the problem is polynomially solvable for $k \leq 3$ and NP-hard otherwise. We provide the next step towards proving this by showing that the problem is polynomially solvable for $k = 3$. In the process, we show that the SCSS_3 problem has a rich structure.

The study of the SCSS_3 problem is also interesting because it yields a better polynomial-time approximation algorithm for the general SCSS problem and hence for the general MEG problem. Obtaining a performance guarantee 2 for the general MEG problem is trivial — any minimal solution achieves this bound. Khuller, Raghavachari and Young [6] gave the first polynomial-time approximation algorithm that achieved a factor better than 2. Their algorithm finds a "large" cycle in G, contracts it, and recurses on the contracted graph. The set of contracted edges forms an SCSS. The cycles are chosen so that any cycle contracted either has length at least some fixed constant k or is a maximum-length cycle in the current graph. The performance guarantee is $\pi^2/6 + O(1/k^2) \approx 1.64$. A natural improvement to the algorithm is to solve the remaining problem optimally, rather than recursively, when the maximum cycle length in the current graph is constant. It can be shown that bigger the value of k for which the problem can be solved optimally, the better the performance guarantee of the improved algorithm.

In this paper, we show that the SCSS_3 problem is equivalent to maximum bipartite matching, so that it can be solved optimally in polynomial time. By modifying the previous method to solve the problem optimally in graphs with maximum cycle length three, we obtain a sequence of polynomial-time algorithms for the MEG problem with performance guarantees arbitrarily close to $\pi^2/6 - 1/36 \approx 1.61$.

The complexity of the SCSS_4 problem is still open. If it can be solved in polynomial time,

On Strongly Connected Digraphs with Bounded Cycle Length

Samir Khuller * Balaji Raghavachari † Neal Young ‡

Abstract

The MEG (minimum equivalent graph) problem is "Given a directed graph, find a smallest subset of the edges that maintains all reachability relations between nodes." We consider the complexity of this problem as a function of the maximum cycle length $\mathcal{C}$ in the graph. If $\mathcal{C} = 2$, the problem is trivial. Recently it was shown that even with the restriction $\mathcal{C} = 5$, the problem is NP-hard. It was conjectured that the problem is solvable in polynomial time if $\mathcal{C} = 3$. In this paper we prove the conjecture, showing that the problem is equivalent to maximum bipartite matching.

The strong dependence of the complexity on the cycle length is in marked contrast to the relation of complexity and cycle length in undirected graphs. Undirected graphs with bounded cycle length have bounded tree width, allowing polynomial-time algorithms for many problems that are NP-hard in general.

A consequence of our result is an improved approximation algorithm for the MEG problem in general graphs. The improved algorithm has a performance guarantee of about 1.61; the best previous algorithm has a performance guarantee of about 1.64.

1 Introduction

Let $G = (V, E)$ be a directed graph. The MEG (minimum equivalent graph) problem on G is the following: find a smallest subset S of the edges that maintains all reachability relations between nodes, i.e., for all pairs of vertices (u, v), v is reachable from u in G iff v is reachable from u using only edges in S. It is known that the problem is NP-hard [3].

Any solution to the MEG problem consists of a solution for each strongly connected component together with a solution for the acyclic graph formed by contracting each strongly connected component into a single vertex. Thus, the MEG problem reduces in linear time (preserving approximation) to the *acyclic* MEG problem and the *strongly connected* MEG problem. We call the latter problem the *minimum SCSS (strongly connected spanning subgraph) problem.*

*Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF Research Initiation Award CCR-9307462. E-mail : samir@@cs.umd.edu.

†Department of Computer Science, The University of Texas at Dallas, Box 830688, Richardson, TX 75083. E-mail : rbk@@utdallas.edu.

‡Department of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14850. E-mail : ney@@orie.cornell.edu.