

# A FAST & EFFECTIVE HEURISTIC FOR THE FEEDBACK ARC SET PROBLEM

**Peter Eades**

*Department of Computer Science  
University of Newcastle*

**Xuemin Lin**

*Department of Computer Science  
University of Queensland*

**W. F. Smyth**

*Department of Computer Science & Systems  
McMaster University*

*School of Computing Science  
Curtin University of Technology*

## ABSTRACT

Let  $G = (V, A)$  denote a simple connected directed graph, and let  $n = |V|$ ,  $m = |A|$ , where  $n - 1 \leq m \leq \binom{n}{2}$ . A *feedback arc set (FAS)* of  $G$ , denoted  $R(G)$ , is a (possibly empty) set of arcs whose reversal makes  $G$  acyclic. A *minimum feedback arc set* of  $G$ , denoted  $R^*(G)$ , is a FAS of minimum cardinality  $r^*(G)$ ; the computation of  $R^*(G)$  is called the *FAS problem*. For every  $n$ , let  $\rho(n)$  denote the maximum, over all digraphs  $G$  of order  $n$ , of  $|R^*(G)|$ . Berger and Shor have recently published an algorithm which, for a given digraph  $G$ , computes a FAS whose cardinality is  $O(\rho(n))$ . Thus the Berger/Shor algorithm provides, in a certain asymptotic sense, an optimal solution to the FAS problem. Unfortunately, the Berger/Shor algorithm is complicated and requires running time  $O(mn)$ . In this paper we present a simple FAS algorithm which guarantees a good (though not optimal) performance bound and executes in time  $O(m)$ . Further, for the sparse graphs which arise frequently in graph drawing and other applications, our algorithm achieves the same asymptotic performance bound that Berger/Shor does.

## 1 INTRODUCTION

The FAS problem defined in the Abstract was apparently first studied by Slater [S61] in the special case  $m = \binom{n}{2}$  that  $G$  is a *tournament*. Interpreting the direction of each arc  $xy$  as a “win” for  $x$  over  $y$  in a round-robin tournament of  $n$  players, Slater looked for a minimum set of results (arcs) whose reversal would make the tournament consistent (acyclic). Much of the subsequent research was devoted to estimating  $\rho_T(n)$ , the restriction of  $\rho(n)$  to tournaments  $T$ . Then

$$\rho_T(n) = \binom{n}{2} / 2 - \phi_T(n) \leq \rho(n),$$

Typeset by  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\texttt{TEX}}$

where  $\phi_T(n) \in O(n^2)$ .

In 1965 Erdős and Moon [EM65] showed that

$$\phi_T(n) \in \Omega(n), \quad \phi_T(n) \in O(n^{3/2} \log^{1/2} n).$$

The lower bound was sharpened to

$$\phi_T(n) \in \Omega(n \log n)$$

by Jung [J71], and Spencer [S71, S80] was able to improve both bounds by showing that

$$\phi_T(n) \in \Theta(n^{3/2}).$$

Then the upper bound was further refined by de la Vega [d83], who proved that

$$\text{prob}[\phi_T(n) \leq 1.73n^{3/2}] \longrightarrow 1$$

as  $n \rightarrow \infty$ , where all labelled tournaments on  $n$  vertices are regarded as equally likely.

Berger and Shor [BS90] provide a basis for both strengthening and generalizing these results. Given a digraph  $G$  of maximum degree  $\Delta$ , their algorithm computes  $R(G)$  of cardinality  $r(G)$  satisfying

$$r^*(G) \leq r(G) \leq m/2 - c_1 m / \Delta^{1/2},$$

where  $c_1$  is a positive constant. On the other hand, they exhibit an infinite class  $\mathcal{G}$  of digraphs with the property that for every  $G \in \mathcal{G}$ ,

$$r^*(G) \geq m/2 - c_2 m / \Delta^{1/2},$$

for some positive constant  $c_2$ . Since  $\rho(n) \geq r^*(G)$ , it follows that their algorithm computes  $r(G) \in O(\rho(n))$ . It follows also that, over all digraphs  $G$  such that  $r^*(G) = \rho(n)$ ,

$$r^*(G) - m/2 \in \Theta(m / \Delta^{1/2}).$$

When  $G$  is a tournament ( $m = \binom{n}{2}$ ,  $\Delta = n - 1$ ), this reduces to the form established by Spencer for  $\rho_T(n)$ .

Since the FAS problem is NP-hard [K72], it is likely that no polynomial-time algorithm computes  $r^*(G)$  exactly; hence the asymptotic performance bound of the Berger/Shor algorithm can be regarded as best possible. At the same time, Berger/Shor requires  $O(mn)$  execution time, and it may be that, in many circumstances (for example, for on-line processing), a much faster algorithm with a suboptimal but good performance bound would be preferred. Such an algorithm, with performance bound

$$r(G) \leq m/2 - n/6$$

and  $O(m)$  execution time, is presented in Section 2 of this paper. Observe that, over digraphs satisfying  $m \in O(n)$ , this performance bound is at least as good

as that of Berger/Shor, and is in fact better over digraphs for which, in addition,  $\Delta \notin O(1)$ .

Section 3 briefly discusses other FAS algorithms and makes suggestions for future work.

## 2 A NEW FAS ALGORITHM

Suppose that the vertices of  $G$  are arranged in some order on a horizontal line and labelled  $v_1, v_2, \dots, v_n$  from left to right. We call such an arrangement a *vertex sequence* and denote it by  $s = v_1 v_2 \dots v_n$ . Clearly each vertex sequence  $s$  induces a feedback arc set  $R(s)$  consisting of all the leftward arcs  $v_j v_i$ ,  $j > i$ ; moreover, every feedback arc set corresponds to some vertex sequence. Thus the FAS problem is equivalent to the problem of determining a vertex sequence  $s^*$  such that  $R(s^*) = R^*(G)$ .

In this section we present a greedy algorithm, called Algorithm GR, which computes a “good” vertex sequence  $s$  (one corresponding to a “small” feedback arc set  $R(s)$ ). The algorithm greedily removes from  $G$  vertices (and their incident arcs) which are sinks or sources, or which satisfy a property that we now specify. For any vertex  $u \in V$ , let  $d(u)$  denote the degree of  $u$ ,  $d^+(u)$  its outdegree, and  $d^-(u)$  its indegree: then  $d(u) = d^+(u) + d^-(u)$ . At each step of Algorithm GR, after sinks and sources have been removed, a single vertex  $u$  is removed for which  $\delta(u) = d^+(u) - d^-(u)$  is currently a maximum. If a vertex  $u$  removed from  $G$  is a sink, it is concatenated with a vertex sequence  $s_2$ ; otherwise,  $u$  is concatenated with a vertex sequence  $s_1$ . When  $G$  has been reduced to an empty graph by successive removals, the output vertex sequence  $s$  is computed by forming the concatenation  $s = s_1 s_2$ . A formal statement of the new algorithm is given below:

```

procedure GR ( $G$  : DiGraph; var  $s$  : VertexSequence);
 $s_1 \leftarrow \emptyset$ ;  $s_2 \leftarrow \emptyset$ ;
while  $G \neq \emptyset$  do
    { while  $G$  contains a sink do
        {choose a sink  $u$ ;  $s_2 \leftarrow us_2$ ;  $G \leftarrow G - u$ };
    while  $G$  contains a source do
        {choose a source  $u$ ;  $s_1 \leftarrow s_1 u$ ;  $G \leftarrow G - u$ };
    choose a vertex  $u$  for which  $\delta(u)$  is a maximum;
     $s_1 \leftarrow s_1 u$ ;  $G \leftarrow G - u$ ;
 $s \leftarrow s_1 s_2$ .

```

We consider first the performance of this algorithm.

**Theorem 2.1** Algorithm GR computes either an empty vertex sequence or a vertex sequence  $s$  for which  $|R(s)| \leq m/2 - n/6$ .

**Proof** For  $n \leq 5$ , GR computes  $s = \emptyset$ . Otherwise, GR induces a partitioning  $V = \bigcup_{i=1}^5 V_i$  where

\*  $V_1$  consists of removed vertices  $u$  which at the time of removal were sinks of indegree  $d^-(u) > 0$ ;

- \*  $V_2$  consists of removed vertices  $u$  which were isolated ( $d(u) = 0$ );
- \*  $V_3$  consists of removed vertices  $u$  which were sources of outdegree  $d^+(u) > 0$ ;
- \*  $V_4$  consists of removed vertices  $u$  such that  $d^+(u) = d^-(u) > 0$ ;
- \*  $V_5$  consists of removed vertices  $u$  such that  $d^+(u) > d^-(u) > 0$ .

For  $1 \leq i \leq 5$ , let  $n_i = |V_i|$  and let  $m_i$  denote the number of arcs removed from  $G$  as a result of the removal of the vertices of  $V_i$ . Then  $n = \sum_{i=1}^5 n_i$ ,  $m = \sum_{i=1}^5 m_i$ , and  $m_2 = 0$ .

Since  $G$  is connected and  $n > 1$ , there is initially no isolated vertex. Suppose that removal of a vertex  $u \in V_4 \cup V_5$  induces an isolated vertex  $v$ . Then  $v$  must result from removal of either  $uv$  or  $vu$ , but not both. Hence  $v$  would have been either a sink before the removal of  $uv$  or a source before the removal of  $vu$ . In either case  $v$  would already have been removed, and so removal of  $u$  cannot give rise to a vertex  $v \in V_2$ . A similar argument shows that removal of  $u \in V_3$  cannot induce an isolated vertex  $v$ , and we see then that  $v$  can only be induced by prior removal of a sink  $u$  and an incident arc  $vu$ . It follows that

$$n_2 \leq m_1. \quad (2.1)$$

Observe that if GR removes a vertex  $u \in V_4$ , the resulting digraph  $G - u$  contains exactly  $d^+(u)$  vertices  $v$  for which  $d^+(v) = d^-(v) + 1$  and exactly  $d^+(u)$  vertices  $w$  for which  $d^-(w) = d^+(w) + 1$ . Since as we have seen  $G - u$  contains no isolated vertex, it follows that the next removed vertex  $x \in V_1 \cup V_3 \cup V_5$ . That is,

$$n_4 \leq n_1 + n_3 + n_5,$$

which, by substitution for  $n_4$ , becomes

$$\begin{aligned} n &\leq 2n_1 + n_2 + 2n_3 + 2n_5 \\ &\leq 2n_1 + n_2 + 3n_3 + 3n_5, \end{aligned}$$

so that, applying (2.1) and observing that  $n_1 \leq m_1$ , we find

$$n \leq 3(m_1 + n_3 + n_5). \quad (2.2)$$

Consider now the number of leftward arcs induced by the removal from  $G$  of a vertex  $u$  and its insertion in the vertex sequence  $s$ . If  $u \in V_4$ , then the number of leftward arcs to  $u$  at the time of its removal is exactly  $d(u)/2$ ; if  $u \in V_5$ , then the number of leftward arcs to  $u$  at the time of its removal is at most  $(d(u) - 1)/2$ . A removed vertex  $u \in V_1 \cup V_2 \cup V_3$  contributes no leftward arcs. Thus,

$$\begin{aligned} r(s) = |R(s)| &\leq m_4/2 + (m_5 - n_5)/2 \\ &= m/2 - (m_1 + m_3 + n_5)/2 \\ &\leq m/2 - (m_1 + n_3 + n_5)/2, \end{aligned}$$

since  $n_3 \leq m_3$ . Applying (2.2), we obtain the desired result.  $\square$

The upper bound of Theorem 2.1 is sharp, since it is attained when  $G$  is a directed cycle of length 3. However, for tournaments, a slightly sharper bound can be stated, as follows:

**Theorem 2.2** Given a tournament  $G$ , Algorithm GR computes a vertex sequence  $s$  for which  $|R(s)| \leq m/2 - \lfloor n/2 \rfloor / 2$ .

**Proof** Suppose that  $n$  is odd and observe that removing the first vertex  $u$  from  $G$  contributes at most  $(n-1)/2$  arc reversals to  $r(s)$ .  $G-u$  will then be a tournament of even order  $n-1$ , and removing a vertex  $v$  from  $G-u$  contributes at most  $\lfloor (n-2)/2 \rfloor$  arc reversals. In general,

$$\begin{aligned} r(s) &\leq (n-1)/2 + \lfloor (n-2)/2 \rfloor + \cdots + 4/2 + \lfloor 3/2 \rfloor + 2/2 \\ &= (n-1)^2/4. \end{aligned}$$

A similar calculation yields

$$r(s) \leq n(n-2)/4$$

when  $n$  is even. Both these inequalities reduce to the required form.  $\square$

We now consider the implementation of Algorithm GR. For this, it is convenient to partition the vertex set of  $G$  into sources, sinks, and  $\delta$ -classes, as follows:

$$V_d = \{u \in V \mid d = \delta(u); d^+(u) > 0; d^-(u) > 0\}, -n+3 \leq d \leq n-3;$$

$$V_{-n+2} = \{u \in V \mid d^+(u) = 0\};$$

$$V_{n-2} = \{u \in V \mid d^-(u) = 0; d^+(u) > 0\}.$$

It is clear that every vertex  $u \in V$  falls into exactly one of these  $2n-3$  classes. To compute these classes for a given digraph  $G$ , Algorithm GR needs to perform a bin sort as an initialization phase. The bin sort implements each vertex class as a bin, with the vertices in each class linked together by a doubly linked list. This structure can be formed in  $O(m)$  time and uses  $O(m)$  space.

Using the bins, it becomes trivial to recognize a sink  $u$ , a source  $u$ , or a vertex  $u$  for which  $\delta(u)$  is a maximum. Furthermore, having determined  $u$ , it becomes trivial to compute  $s_1$  or  $s_2$  and, at the same time, to output the leftward arcs, if any, currently incident at  $u$ .

It remains only to be seen how to form  $G-u$ . The vertex  $u$  itself can be removed in  $O(1)$  time by eliminating it from its bin list. As a result, every vertex  $v$  adjacent to  $u$  will either become a sink, a source, or an element of an adjacent bin: to perform these updates requires  $O(1)$  time for each arc incident at  $u$ . We have proved

**Theorem 2.3** Algorithm GR can be implemented in  $O(m)$  time and  $O(m)$  space.

### 3 REMARKS

In Section 2 we described Algorithm GR, a simple new FAS algorithm with a good (though not optimal) performance bound and  $O(m)$  (linear) execution time. For sparse digraphs this performance bound is at least as good as that of Berger and Shor, whose algorithm requires execution time  $O(mn)$ . However, for dense digraphs, the Berger/Shor algorithm has a performance bound which is, in a certain sense, optimal.

Our new algorithm is the only known FAS algorithm which executes in linear time and has a performance bound less than  $m/2$  for all digraphs. Trivial linear algorithms [ELS89,L92] have been known for some time, but these do not even guarantee good performance bounds for tournaments. [ELS89] describes new greedy and divide-and-conquer algorithms which execute in linear time and guarantee good performance bounds for tournaments; but, as shown in [L92], there nevertheless exist digraphs for which these algorithms yield feedback arc sets of cardinality greater than  $m/2$ . ([L92] also describes an  $O(mn)$  FAS algorithm (a greedier version of Algorithm GR) whose performance bound is not as good as Berger/Shor in general, but which nevertheless yields slightly better results when applied to oriented cubic graphs.)

In a practical context, the real challenge is to find a technique for expressing performance bounds, not just in terms of  $\rho(n)$ , but rather in terms of  $r^*(G)$ . For many digraphs  $G$ ,  $r^*(G)$  is much smaller than  $\rho(n)$ , and in such cases there is persuasive computational evidence [ELS89] that even simple linear algorithms will “usually” yield feedback arc sets of cardinality not much greater than  $r^*(G)$ . Thus, to find a means of expressing performance bounds of certain algorithms (such as Algorithm GR and the Berger/Shor algorithm) in terms of  $r^*(G)$  would provide a much sounder basis for evaluating their practical effectiveness.

### REFERENCES

- [BS90] Bonnie Berger & Peter W. Shor, **Approximation algorithms for the maximum acyclic subgraph problem**, *Proc. First ACM-SIAM Symposium on Discrete Algorithms* (1990) 236-243.
- [d83] W. F. de la Vega, **On the maximum cardinality of a consistent set of arcs in a random tournament**, *J. Combin. Theory, Series B* 35 (1983) 328-332.
- [ELS89] Peter Eades, Xuemin Lin & W. F. Smyth, *Heuristics for the Feedback Arc Set Problem*, Technical Report No. 1, Curtin University of Technology, School of Computing Science (1989).
- [EM65] P. Erdős & J. W. Moon, **On sets of consistent arcs in tournaments**, *Canadian Mathematical Bulletin* 8 (1965) 269-271.
- [J70] H. A. Jung, **On subgraphs without cycles in tournaments**, *Combinatorial Theory & Its Applications II*, edited by P. Erdős, A. Rényi & Vera T. Sós, North-Holland (1970) 675-677.
- [K72] Richard M. Karp, **Reducibility among combinatorial problems**, *Com-*

*plexity of Computer Computations*, Plenum Press (1972) 85-103.

[L92] Xuemin Lin, *Analysis of Algorithms for Drawing Graphs*, Ph.D. thesis, University of Queensland, Department of Computer Science (1992).

[S61] P. Slater, **Inconsistencies in a schedule of paired comparisons**, *Biometrika* 48 (1961) 303-312.

[S71] J. Spencer, **Optimal ranking of tournaments**, *Networks* 1 (1971) 135-138.

[S80] J. Spencer, **Optimally ranking unrankable tournaments**, *Period. Math. Hungar.* 11-2 (1980) 131-144.

## ACKNOWLEDGEMENT

The work of the third author was supported in part by Grant No. A8180 of the Natural Sciences and Engineering Research Council of Canada.