

Packing Feedback Arc Sets in Reducible Flow Graphs

Han Xiao

Department of Mathematics, The University of Hong Kong,
Hong Kong, China
`hxiao.math@connect.hku.hk`

Abstract

In this paper we establish a min-max relation in arc-weighted reducible flow graphs. In particular, we prove that the maximum cardinality of feedback arc set packings equals the minimum total weight of cycles. We also present an $O(n^2m)$ algorithm for finding a maximum feedback arc set packing in reducible flow graphs. Our result relies heavily on Ramachandran's [10, 11] structural analysis of reducible flow graphs.

Keywords. Min-max relation, feedback arc set, packing, reducible flow graph, algorithm

1 Introduction

A reducible flow graph is a special flow graph which corresponds to the flowchart of computer programs via structured programming. As described in [6], the “divide-and-conquer” approach applies to reducible flow graphs, which makes the analysis easier on such graphs. Reducible flow graphs also occur frequently in practice. Statistics suggest that most flow graphs associated with computer programs are reducible flow graphs. Therefore, reducible flow graphs have attracted many research efforts over the past few decades. Various characterizations of reducible flow graphs can be found in [7]. The first efficient algorithm for recognizing reducible flow graphs was given by Hopcroft and Ullman [9]. Based on their work, Tarjan [13] derived a more efficient algorithm for testing flow graphs reducibility.

In this paper, by a cycle (*resp.* path) in a digraph we mean a directed one. A digraph is called *acyclic* if it contains no cycles, and is abbreviated a *DAG* (directed acyclic graph). Let $G = (V, A)$ be a digraph with a nonnegative integral function w defined on arcs in A . A set $F \subseteq A$ is a *feedback arc set* (*FAS*) for G if $G' = (V, A \setminus F)$ is acyclic. Let $H = (A, \mathcal{C})$ be a hypergraph whose vertex set is the arc set of G and edges (with repetition allowed) are the arc sets of cycles in G . A subset \mathcal{C}' of edge set \mathcal{C} in H is called a *cycle packing* if each arc $a \in A$ occurs at most $w(a)$ times in cycles of \mathcal{C}' . A *transversal* of a hypergraph is a subset of its vertex

set which intersects each edge of the hypergraph. Note that a transversal of $H = (A, \mathcal{C})$ is a feedback arc set of G . Let ν_w denote the maximum cardinality of cycle packings and τ_w denote the minimum weight of feedback arc sets. It is easy to prove that $\nu_w \leq \tau_w$. When equality $\nu_w = \tau_w$ holds for all nonnegative integral function w , the hypergraph H is said to satisfy the min-max relation and the graph G is said to be *cycle Mengerian (CM)*. In [10], Ramachandran gave an $O(n^2 m \log(n^2/m))$ algorithm for finding a minimum weight feedback arc set in arc-weighted reducible graphs. She [11] also established a conjecture of Frank and Gyarfás [5] by showing that the minimum cardinality of feedback arc sets is equal to the maximum cardinality of cycle packings in unweighted reducible flow graphs. Her proof yielded an $O(\min\{mn^{5/3}, m^2\})$ algorithm for finding a maximum collection of arc disjoint cycles. Chen and Zang [3] pointed out that although Ramachandran's proof [11] was intended for unweighted reducible flow graphs, it could be adapted to handle the weighted case. Based on Ramachandran's work [10, 11], Chen and Zang proved that every reducible flow graph is CM. They also reduced the time complexity of finding a maximum cycle packing in weighted reducible flow graphs to $O(n^2 m \log(n^2/m))$ by avoiding using augmenting path method for finding the maximum flow as a subroutine in their algorithm.

A *clutter* is a hypergraph no edge of which is contained in another. Note that hypergraph $H = (A, \mathcal{C})$ is a clutter. The hypergraph $H^\perp = (A, \mathcal{F})$ is called the *blocker* of H , where vertex set of H^\perp is the arc set of G and edges (with repetition allowed) are the collection of feedback arc sets. Analogous to the definition of cycle packing, we call a subset \mathcal{F}' of edge set \mathcal{F} in H^\perp a *feedback arc set packing* if each arc a occurs at most $w(a)$ times in feedback arc sets of \mathcal{F}' . Note that a transversal of H^\perp is a cycle in G . Let λ_w denote the maximum cardinality of feedback arc set packings and μ_w denote the minimum weight of cycles. Obviously, $\lambda_w \leq \mu_w$. When equality $\lambda_w = \mu_w$ holds for all nonnegative integral function w , the hypergraph H^\perp is said to satisfy the min-max relation and the graph G is said to be *feedback arc set Mengerian (FASM)*. It has been proved that hypergraph $H = (A, \mathcal{C})$ derived from reducible flow graph $G = (V, A)$ satisfies the min-max relation. Hence every reducible flow graph is CM. A natural question is whether the blocker $H^\perp = (A, \mathcal{F})$ of hypergraph H satisfies the min-max relation as well. The main contribution of this paper is twofold. First, we give an affirmative answer to this question, hence every reducible flow graph is FASM. Second, we present an $O(n^2 m)$ algorithm for finding a maximum feedback arc set packing in reducible flow graphs. Our result relies heavily on the structural analysis for reducible flow graphs of Ramachandran's work [10, 11].

The remainder of this paper is organized as follows. In section 2, we introduce some preliminary knowledge on reducible flow graphs. In section 3, we first prove that every reducible flow graph is FASM, then we present our algorithm and establish its correctness.

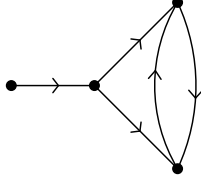


Figure 1: The obstruction F

2 Preliminaries

A *flow graph* is a digraph with a root such that the root reaches all other vertices via paths. Let $G = (V, A; r)$ be a flow graph with root r . A *DAG* of G is a maximal acyclic subgraph of G which remains a flow graph with root r . We call G *reducible* if the DAG of G is unique. A DAG of G obtained from a depth-first spanning tree rooted at r by adding a maximal subset of arc set A is called a *DFS DAG* of G . For $u, v \in V$, we say u *dominates* v if every $r - v$ path passes through u . Clearly, every vertex dominates itself. For a subgraph G' of G , a vertex v of G' is called an *entry vertex* if $v = r$ or there is an arc (u, v) of G with u outside G' . Observe that arcs in a reducible flow graph can be uniquely partitioned into the *forward* (or *DAG*) arc set and the *back arc* set.

Following are some useful characterizations of reducible flow graphs.

Theorem 2.1. *Let $G = (V, A; r)$ be a flow graph and let D be a DFS DAG of G . Then the following statements are equivalent:*

- (a) G is a reducible flow graph.
- (b) F is the unique obstruction for the transitive closure of G (see Figure 1).
- (c) D is the unique DAG of G .
- (d) The arc set A can be partitioned into A_1 and A_2 such that $(V, A_1; r)$ is a DAG of G and u dominates v in G for each arc (v, u) in A_2 .
- (e) Every cycle C in G contains exactly one back arc $a \in A \setminus A(D)$. Moreover, the head of back arc $a \in A \setminus A(D)$ is the entry vertex of C which dominates other vertices on C .

Proof. The equivalence of (a) – (d) can be found in [7] and [8]. The implication (a) \Rightarrow (e) is proved in [12] and the converse direction (e) \Rightarrow (a) is contained in [3]. \square

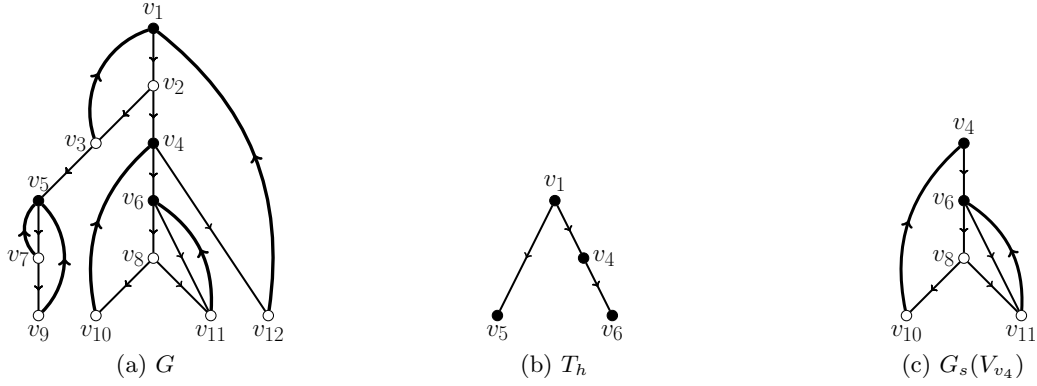


Figure 2: Reducible flow graph G and corresponding T_h , $G_s(V_{v_4})$

We follow some notations introduced by Hecht, Ullman [7, 8] and Ramachandran [10, 11]. Here V_h denotes the set of heads of all back arcs in G , and T_h denotes the *head dominator tree* of G which demonstrates the domination relation on $V_h \cup \{r\}$: the descendants of a vertex u in T_h are precisely the vertices in V_h dominated by u in G . For each $u \in V_h \cup \{r\}$, let $(u_1, v_1), (u_2, v_2), \dots, (u_k, v_k)$ be all the back arcs in G whose heads are dominated by u . We use V_u to denote the *dominated back arc vertex set* of u which consists of all the vertices in V lying on some DAG path from u to u_i , for $1 \leq i \leq k$. Ramachandran [10, 11] proved that u dominates all vertices in V_u . We use $G_s(V_u)$ to denote the subgraph of G induced by V_u (see Figure 2).

The following properties on dominated back arc vertex sets were established by Chen and Zang [3].

Theorem 2.2. *Let $G = (V, A; r)$ be a reducible flow graph and let u, v be two vertices in $V_h \cup \{r\}$. Then the following statements hold:*

- (a) $G_s(V_u)$ is a reducible flow graph rooted at u .
- (b) u is the unique entry vertex of $G_s(V_u)$ in G .
- (c) If $G_s(V_u)$ and $G_s(V_v)$ have a common vertex, then either u dominates v or v dominates u .

3 Maximum feedback arc set packings in reducible flow graphs

3.1 Min-max relation

Let $G = (V, A; r)$ be a reducible flow graph with a nonnegative integral weight $w(a)$ on each arc $a \in A$. We construct a network $N(r, t)$ from G as follows. Add a new vertex t . For $v \in V_h$, make a copy v' , redirect back arcs from v to v' and connect v' to t by a *newly added arc* (v', t) . We also define a length function l on arcs in N from the weight function w on arcs in G : first, set

As we shall see, initializing \mathcal{C} with $\{C_v : v \in V_h\}$, recursively choosing some quasi-FAS $\partial^+(U_i)$ of current \mathcal{C} and deleting C_v that are broken by $\partial^+(U_i)$ from \mathcal{C} always end with an empty set \mathcal{C} . When \mathcal{C} becomes empty, the union of all the chosen quasi-FASs breaks all $v - v'$ paths for $v \in V_h$, which yields a FAS of G naturally. For clarity, we call an execution of choosing a quasi-FAS of \mathcal{C} and deleting C_v that are broken from \mathcal{C} an *iteration* and call the process of constructing a FAS of G a *stage*.

To make sure \mathcal{C} becomes empty, we show that in each iteration, there exists a quasi-FAS of current \mathcal{C} which breaks some $C_v \in \mathcal{C}$. Notice that in the worst case, each C_v in \mathcal{C} needs a FAS. Hence it suffices to show there are enough FASs for $C_v \in \mathcal{C}$ throughout the stage.

Lemma 3.2. *Let \mathcal{C} be the collection of C_v for $v \in V_h$. At each stage, we start with \mathcal{C} , recursively choose the first unused quasi-FAS of \mathcal{C} in $N(r, t)$ and delete C_v that are broken from \mathcal{C} , then we always end with an empty set \mathcal{C} . Moreover, at the end of stage k , the number of unused FAS of C_v for $v \in V_h$ is at least $K - k$.*

Our proof of Lemma 3.2 is based on the following observation.

Lemma 3.3. *In each iteration of a stage, let \mathcal{C}' be the subset of \mathcal{C} such that the first unused quasi-FAS of \mathcal{C} is the first unused FAS of C_v for $C_v \in \mathcal{C}'$. Then there exists a $C_u \in \mathcal{C}'$ such that the number of unused FAS of C_u decreases by precisely one in this stage.*

Proof. At the end of each iteration, all cycle classes in \mathcal{C}' will be deleted from \mathcal{C} . No quasi-FAS of \mathcal{C}' is chosen in the succeeding iterations of this stage. If for $C_v \in \mathcal{C}'$, there exists an FAS of C_v chosen before this iteration, then the first unused quasi-FAS of \mathcal{C} no longer depends on \mathcal{C}' , as all cycle classes in \mathcal{C}' are deleted from \mathcal{C} before this iteration, a contradiction. \square

Notice that each $r - t$ cut $\partial^+(U_i)$ of N can only be used once in an FAS packing. Therefore, at the end of each stage, we decrease the value of $l(a)$ by one for $a \in \partial^+(U_i)$ if $\partial^+(U_i)$ is chosen during this stage. The distance $d(v)$ for $v \in V(N)$ with respect to the latest length function l provides some useful information in each stage. For $v \in V_h$, $\min\{d(t), d(v')\} - d(v)$ is the number of unused FASs of C_v . Besides, the first unused quasi-FAS of \mathcal{C} in each iteration of a stage is exactly the first unused FAS of C_v with the largest $d(v)$ in \mathcal{C} .

Now we are ready to present a proof of our key lemma.

Proof of Lemma 3.2. We apply induction on the number of stages.

Observe that the initial state, viewed as stage 0, automatically satisfies the induction hypothesis. Therefore, the proof of basis and the proof of induction step are essentially the same.

So we proceed to the induction step. Suppose the statement in Lemma 3.2 holds for stage $k \geq 0$. We aim to establish the statement for stage $k + 1$.

We initially start with \mathcal{C} which consists of C_v for every $v \in V_h$. By induction hypothesis, at the beginning of stage $k + 1$, the number of available FAS of C_v for every $v \in V_h$ is at least $K - k > 0$. Therefore, every C_v for $v \in V_h$ has available FAS. Let C_u be an arbitrary cycle class in \mathcal{C} . Let $\partial^+(U_{i_\tau}), \dots, \partial^+(U_{i_1})$ with $i_\tau < \dots < i_1$ be all the FASs of C_u used in stage $k + 1$. Assume these $r - t$ cuts are chosen as the first unused quasi-FAS of a series of nested subsets $\mathcal{C}^{(\tau)} \subset \dots \subset \mathcal{C}^{(1)} := \mathcal{C}$ respectively. If $\tau = 1$, the number of available FAS of C_u decreases by one. By induction hypothesis, the number of available FAS of C_u is at least $K - k - 1$ at the end of stage $k + 1$. So assume $\tau \geq 2$. Let $u_j \in V_h$ be a vertex with the largest $d(v)$ among all $C_v \in \mathcal{C}^{(j)}$ for $j = \tau, \dots, 1$. Since every C_v for $v \in V_h$ has available FASs, the first available quasi-FAS $\partial^+(U_{i_j})$ of $\mathcal{C}^{(j)}$ is essentially the first available FAS of $C_{u_j} \in \mathcal{C}^{(j)}$. Without loss of generality, assume C_{u_j} is a cycle class in $\mathcal{C}^{(j)}$ whose number of available FASs decreases by one in this stage. By Lemma 3.3, such C_{u_j} exists. It follows that $d(u) \leq d(u_\tau) < d(u'_\tau) \leq \dots \leq d(u_1) < \min\{d(t), d(u')\}$. Obviously, C_u is not the cycle class with the least number available FASs among all vertices in V_h , since $\min\{d(t), d(u')\} - d(u) > d(u'_\tau) - d(u_\tau)$. Moreover, by Lemma 3.3, the number of available FASs of C_{u_τ} decreases by one in this stage. Then inequality $\min\{d(t), d(u')\} - d(u) > d(u'_\tau) - d(u_\tau)$ together with the induction hypothesis implies that the number of available FASs of C_u decreases by $\tau \geq 2$, but is no less than $K - k - 1$ at the end of stage $k + 1$.

Therefore, we always have available FAS of C_v for $v \in V_h$ in each iteration of a stage. And the first available quasi-FAS of current \mathcal{C} is always a available FAS of some $C_v \in \mathcal{C}$. Hence the cardinality of \mathcal{C} decreases in each iteration till \mathcal{C} becomes empty. \square

Recall that there are K quasi-FASs of $\mathcal{C} = \{C_v : v \in V_h\}$, and we use precisely one in a stage. Hence there are K stages in all. Moreover, Lemma 3.2 guarantees each stage returns a FAS of G . Therefore we can attain an FAS packing of cardinality K . Since K is also the minimum total weight of cycles in G , the following theorem follows directly.

Theorem 3.4. *Every reducible flow graph is FASM.*

3.2 Algorithm and complexity

The constructive proof of Theorem 3.4 naturally yields an algorithm for finding a maximum FAS packing in a reducible flow graph. However, regardless of other operations, we need to repeat stage K times before we arrive at a maximum FAS packing, which makes the algorithm pseudo-polynomial. Observe that consecutive stages might return the same FAS because chosen

quasi-FASs in consecutive stages might be the same. Hence a natural idea is to merge stages returning the same FAS into one. Suppose we have merged the first $k - 1$ stages returning the same valid FAS into $\kappa - 1$ new STAGES respectively and updated the length function l accordingly. Now we are at stage k , which cannot be merged into STAGE $\kappa - 1$. We start our construction with \mathcal{C} which initially consists of C_v for every $v \in V_h$. Then we recursively choose the first available quasi cut of current \mathcal{C} in N , delete every cycle class C_v from \mathcal{C} that is broken by the chosen quasi cut, and update d by $\max_{C_v \in \mathcal{C}} d(v)$. After $j \geq 0$ iterations, we arrive at a nonempty set \mathcal{C} , and $\partial^+(U_{d+1})$ is the first available quasi cut of current \mathcal{C} . Observe that the following $\min_{(u,w) \in \partial^+(U_{d+1})} \{d(w) - d\}$ quasi cuts of current \mathcal{C} including $\partial^+(U_{d+1})$ are actually the same. If current \mathcal{C} occurs in succeeding stages, we might use the same $r - t$ cut again. In each iteration of stage k , we update α (initialized with a value large enough) by the smaller value of current α and $\min_{(u,w) \in \partial^+(U_{d+1})} \{d(w) - d\}$. When \mathcal{C} becomes empty, quasi cuts chosen in the succeeding α stages are actually the same. It follows that the succeeding α consecutive stages including stage k give rise to the same valid FAS of G . Therefore, we merge these α stages including stage k into new STAGE κ . Besides, observe that arcs might be chosen repeatedly in more than one iterations of a stage. Let \mathcal{T} be the collection of all the chosen quasi cuts of a stage. At the end of STAGE κ , instead of updating the length function l separately, we go through every $r - t$ cut T in \mathcal{T} and decrease $l(a)$ by α for each arc $a \in T$.

The following lemma guarantees that, by making the biggest jump on stages in exchange for efficiency, our construction ends in $O(n)$ STAGES, where n is number of vertices in G .

Lemma 3.5. *By merging consecutive stages returning the same valid FAS into a new STAGE, the construction ends in $O(n)$ STAGES, where n is the number of vertices in G .*

Proof. At the beginning of our algorithm, we may decompose all vertices of $V(N)$ into at most $|V(N)|$ subsets of $V(N)$ according to the distance $d(v)$ with respect to initial length function l . At each STAGE, there exists a chosen quasi cut S satisfying equality $\alpha = \min_{(u,w) \in S} d(w) - \max_{(u,w) \in S} d(u)$. At the end of this STAGE, the length of each arc in quasi cut S decreases by α . It follows that the vertex set of distance $\min_{(u,w) \in S} d(w)$ and the vertex set of distance $\max_{(u,w) \in S} d(u)$ will merge into one vertex set of the same distance. Therefore, the number of decompositions decreases by at least one in each STAGE. After $O(|V(N)|)$ STAGES, all vertices in $V(N)$ will belong to the same distance class, implying the termination of our algorithm. Also notice that $|V(N)| \leq 2n + 1$, where n is the number of vertices in G . Hence the construction ends in $O(n)$ STAGES. \square

Now we are ready to present our algorithm.

Algorithm 1: FAS packing algorithm

Input : An arc-weighted reducible flow graph $G = (V, A; r)$

Output: A maximum FAS packing \mathcal{F} of G

Construct $N(r, t; l)$ from $G = (V, A; r)$;

Find $d(v)$ for every $v \in V(N)$ with respect to l in N ;

$d \leftarrow \max_{v \in V_h} d(v)$;

$\mathcal{F} \leftarrow \emptyset$;

STAGE: **while** $d < d(t)$ **do**

$\mathcal{T} \leftarrow \emptyset$;

$F \leftarrow \emptyset$;

$\alpha \leftarrow d(t)$;

Augment $(\partial^+(U_{d+1}))$;

 Delete all newly added arcs from F ;

 Replace remaining arcs in F by corresponding arcs in G ;

$\mathcal{F} \leftarrow \mathcal{F} \cup \{(F, \alpha)\}$;

Update (l) ;

 Find $d(v)$ for every $v \in V(N)$ with respect to current l in N ;

$d \leftarrow \max_{v \in V_h} d(v)$;

Return \mathcal{F} ;

procedure **Augment** (S)

$\mathcal{T} \leftarrow \mathcal{T} \cup \{S\}$;

$F \leftarrow F \cup S$;

$\alpha \leftarrow \min\{\alpha, \min_{(u,w) \in S} \{d(w) - d\}\}$;

if $\exists (v', t) \in S$ **then**

$d \leftarrow \max_{(v', t) \in S} d(v)$;

Augment $(\partial^+(U_{d+1}))$;

procedure **Update** (l)

for $T \in \mathcal{T}$ **do**

for $a \in T$ **do**

$l(a) \leftarrow l(a) - \alpha$;

In our algorithm, we need to repeatedly find all the $d(v)$ with respect to latest length function l in N . The following lemma ensures that this task can always be finished efficiently due to the special structure of N , a DAG.

Lemma 3.6. *All the single source shortest distances in an acyclic digraph $D = (V, A)$ can be found in $O(n + m)$, where $n = |V|$ and $m = |A|$.*

Proof. Let $l : A \rightarrow \mathbb{R}$ be the length function defined on arcs of D and $r \in V$ be the source given. Denote the distance of v from r by $d(v)$ for every $v \in V$. Initially, set $d(r) = 0$ and $d(v) = +\infty$ for $v \in V \setminus \{r\}$, which takes $O(n)$ time. Then topologically sort all vertices and denote the ordering by π . This process can be finished in $O(m + n)$ time [1]. Visit every vertex v in topological order and update $d(v)$ by $\min_{u: \pi(u) < \pi(v)} \{d(u) + l(u, v)\}$. This step returns the

global shortest distance $d(v)$ because vertices behind v in a topological order cannot be on any $r-v$ paths. Throughout the updating process, each arc is only visited once which requires $O(m)$ time. Hence the time complexity of finding all the shortest distances from r is the same as the time complexity of topological sort, which is $O(m+n)$. \square

Theorem 3.7. *In an arc-weighted reducible flow graph $G = (V, A; r)$, a maximum FAS packing \mathcal{F} can be found in $O(n^2m)$ time, where $n = |V|$ and $m = |A|$.*

Proof. By the construction of network $N(r, t; l)$, $|V(N)| \leq 2n + 1$ and $|A(N)| \leq m + n$. It follows that $O(|V(N)|) = O(n)$. Since G is connected, $O(|A(N)|) = O(m)$. Notice that the construction of network N involves finding the shortest distance $d(v)$ for $v \in V_h$. By Lemma 3.6, we can find all the distance $d(v)$ from r in $O(m+n) = O(m)$ time. So the construction of network N requires $O(m)$ time. It is easy to see that all operations outside STAGEs take $O(m)$ time. By Lemma 3.5, our algorithm ends in $O(n)$ STAGEs. It remains to show that the time complexity of each STAGE is $O(nm)$. Notice that each STAGE contains a recursive function $Augment(S)$ and a function $Update(l)$. Each invocation of function $Augment(S)$ in a STAGE decreases the number of newly added arcs in S by at least one. So $Augment(S)$ in a STAGE ends in $O(n)$ invocations. Function $Augment(S)$ itself involves finding $\min_{(u,w) \in S} \{d(w) - d\}$ and $\max_{(v',t) \in S} d(v')$, both of which can be done in $O(n)$ time. So the complexity of function $Augment(S)$ in one STAGE is $O(n^2)$. As to function $Update(l)$, it requires $O(nm)$ time in one STAGE. Hence the time complexity of each STAGE is $O(nm)$. The theorem follows. \square

Acknowledgments

The author would like to thank Prof. Wenan Zang for his invaluable suggestions.

References

- [1] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows - Theory, Algorithms, and Applications. Prentice Hall Inc., Englewood Cliffs, NJ (1993)
- [2] Chen, X., Ding, G., Hu, X., Zang, W.: A min-max relation on packing feedback vertex sets. Math. Oper. Res. **31**(4), 777–788 (2006)
- [3] Chen, X., Zang, W.: An efficient algorithm for finding maximum cycle packings in reducible flow graphs. Algorithmica **44**(3), 195–211 (2006)
- [4] Ding, G., Zang, W.: Packing cycles in graphs. J. Combin. Theory Ser. B **86**(2), 381–407 (2002)

- [5] Frank, A., Gyárfas, A.: Directed graphs and computer programs. In: *Problèmes Combinatoires et Théorie des Graphes*, 260, pp. 157–158 (1976)
- [6] Hecht, M.S.: *Flow analysis of computer programs*. North-Holland, New York (1977)
- [7] Hecht, M.S., Ullman, J.D.: Flow graph reducibility. *SIAM J. Comput.* **1**(2), 188–202 (1972)
- [8] Hecht, M.S., Ullman, J.D.: Characterizations of reducible flow graphs. *J. Assoc. Comput. Mach.* **21**(3), 367–375 (1974)
- [9] Hopcroft, J.E., Ullman, J.D.: An $n \log n$ algorithm for detecting reducible graphs. In: *Proc. 6th Annual Princeton Conference on Information Sciences and Systems*, pp. 119–122. Princeton, NJ (1972)
- [10] Ramachandran, V.: Finding a minimum feedback arc set in reducible flow graphs. *J. Algorithms* **9**(3), 299–313 (1988)
- [11] Ramachandran, V.: A minimax arc theorem for reducible flow graphs. *SIAM J. Discrete Math.* **3**(4), 554–560 (1990)
- [12] Shamir, A.: A linear time algorithm for finding minimum cutsets in reducible graphs. *SIAM J. Comput.* **8**(4), 645–655 (1979)
- [13] Tarjan, R.E.: Testing flow graph reducibility. *J. Comput. System Sci.* **9**(3), 355–365 (1974)