Available online at www.sciencedirect.com
SCIENCE DIRECT®

ELSEVIER

Electronic Notes in Discrete Mathematics 17 (2004) 229–235

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

# On the Strongly Connected and Biconnected Components of the Complement of Graphs

Stavros D. Nikolopoulos[1] Leonidas Palios[2]

*Department of Computer Science, University of Ioannina, P.O.Box 1186, GR-45110 Ioannina, Greece*

**Abstract**

In this paper, we consider the problems of computing the strongly connected components and the biconnected components of the complement of a given graph. In particular, for a directed graph $G$ on $n$ vertices and $m$ edges, we present a simple algorithm for computing the strongly connected components of $\overline{G}$ which runs in optimal $O(n+m)$ time. The algorithm can be parallelized to yield an $O(\log^2 n)$-time and $O(m^{1.188}/\log n)$-processor solution. As a byproduct, we obtain a very simple optimal parallel co-connectivity algorithm.
Additionally, we establish properties which, for an undirected graph on $n$ vertices and $m$ edges, enable us to describe an $O(n+m)$-time algorithm for computing the biconnected components of $\overline{G}$, which can be parallelized resulting in an algorithm that runs in $O(\log n)$ time using $O((n+m)/\log n)$ processors.

## 1 Theoretical Framework

We consider finite (directed) undirected graphs with no (directed) loops or (directed) multiple edges. Let $G$ be an undirected graph; then, $V(G)$ and $E(G)$ denote the set of vertices and of edges of $G$ respectively.

**Lemma 1.1.**

*(i) Let $G$ be an undirected graph on $n$ vertices and $m$ edges. If $v$ is a vertex of $G$ of minimum degree, then the degree of $v$ does not exceed $\sqrt{2m}$.*

[1] Email: stavros@cs.uoi.gr
[2] Email: palios@cs.uoi.gr


doi:10.1016/j.endm.2004.03.044

*(ii) Let $G$ be a directed graph on $n$ vertices and $m$ edges. If $v$ is a vertex of $G$ of minimum sum of indegree and outdegree, then the sum of indegree and outdegree of $v$ does not exceed $2\sqrt{m}$.*

Let $G$ be a graph. We say that a set $E \subseteq E(G)$ of cardinality $\geq 2$ has the *biconnectivity property in $G$* if, for every pair of edges $e, e' \in E$, the subgraph of $G$ spanned by the edges in $E$ contains a simple cycle that passes through both $e$ and $e'$ [7].

**Lemma 1.2.** *Let $G$ be an undirected graph, let set $E \subseteq E(G)$ having the biconnectivity property in $G$ and let $V(E)$ be the set of vertices incident to at least one edge in $E$. Then,*

*(i) the edge set of the subgraph of $G$ induced by $V(E)$ also has the biconnectivity property;*

*(ii) for every edge $e \in E$ and any two vertices $x, y \in V(E)$, the subgraph of $G$ spanned by the edges in $E$ contains a simple path from $x$ to $y$ that passes along $e$.*

Due to the transitivity of the relation "to have the biconnectivity property" [7], it follows that if two edge sets $E_1$ and $E_2$ have the biconnectivity property and are not disjoint then the set $E_1 \cup E_2$ also has the biconnectivity property.

**Lemma 1.3.** *Let $G$ be an undirected graph, let $E_1, E_2 \subseteq E(G)$ be disjoint sets of edges having the biconnectivity property in $G$, and let $V(E_1), V(E_2)$ be the sets of vertices incident to at least one edge in $E_1$ and $E_2$ respectively.*

*(i) If $V(E_1) \cap V(E_2) = \emptyset$ and there exist distinct vertices $u, v \in V(E_1)$ and $x, y \in V(E_2)$ such that $ux \in E(G)$ and $vy \in E(G)$, then the edge set of the subgraph of $G$ induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property.*

*(ii) Suppose that $V(E_1) \cap V(E_2) = \{v\}$.*

*a) If there exist vertices $x \in V(E_1) - \{v\}$ and $y \in V(E_2) - \{v\}$ such that $xy \in E(G)$, then the edge set of the subgraph of $G$ induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property;*

*b) If there exist vertices $x \in V(E_1) - \{v\}$, $y \in V(E_2) - \{v\}$, and vertex $z \in V(G) - (V(E_1) \cup V(E_2))$ such that $xz, yz \in E(G)$, then the edge set of the subgraph of $G$ induced by $V(E_1) \cup V(E_2) \cup \{z\}$ has the biconnectivity property;*

*c) If there exist vertices $x \in V(E_1) - \{v\}$ and $y \in V(E_2) - \{v\}$, and edge set $E_3 \subseteq E(G)$ for which $E_3$ has the biconnectivity property and $V(E_3) \cap (V(E_1) \cap V(E_2))\emptyset$ such that $xa, yb \in E(G)$ for two distinct vertices $a, b \in V(E_3)$, then the edge set of the subgraph of $G$ induced by $V(E_1) \cup V(E_2) \cup$*

*$V(E_3)$ has the biconnectivity property.*

*(iii) If $|V(E_1) \cap V(E_2)| \geq 2$, then the edge set of the subgraph of $G$ induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property.*

## 2 Strongly Connected Components of the Complement of a Graph

Next, we present a simple optimal algorithm for computing the strongly connected components (s.c.c, for short) of the complement $\overline{G}$ of a directed graph $G$.

**Lemma 2.1.** *Let $G$ be a directed graph, and let $v$ be a vertex of $G$.*

*(i) Vertex $v$ and the vertices $x$ such that neither $vx$ nor $xv$ belongs to $E(G)$ belong to the same s.c.c of $\overline{G}$.*

*(ii) Let $G'_v$ be the directed graph where*
$V(G'_v) = \{v\} \ \cup \ \{\, x \mid vx \in E(G) \text{ or } xv \in E(G)\,\}$;
$E(G'_v) = \{\, xy \mid x, y \in V(G'_v) - \{v\} \text{ and } xy \in E(G)\,\}$
$\qquad \cup \{\, vx \,|\, x \in V(G'_v) - \{v\} \text{ and } \forall z \in V(G) - (V(G'_v) - \{v\}),\ zx \in E(G)\,\}$
$\qquad \cup \{\, yv \,|\, y \in V(G'_v) - \{v\} \text{ and } \forall z \in V(G) - (V(G'_v) - \{v\}),\ yz \in E(G)\,\}$.
*Then, two vertices $x, y \in V(G'_v)$ belong to the same s.c.c of $\overline{G}$ iff they belong to the same s.c.c of $\overline{G'_v}$.*

The algorithm takes advantage of Lemma 1.1(ii) and Lemma 2.1. It uses an array `sccc[]` of size equal to the number of vertices of the input graph $G$ in which it records the s.c.c of $\overline{G}$; in particular, `sccc[`$a$`]` $=$ `sccc[`$b$`]` iff $a, b$ belong to the same s.c.c of $\overline{G}$. In more detail, the algorithm works as follows:

*Algorithm Strong_Co-components*

1. $v \leftarrow$ a vertex of $G$ of minimum degree (sum of indegree and outdegree);
2. if the indegree and outdegree of $v$ are both equal to 0
   then {*$G$ is trivial or a disconnected graph; $\overline{G}$ is strongly connected*}
   for each vertex $w$ of $G$ do
   `sccc[`$w$`]` $\leftarrow v$; {*$v$: representative of the s.c.c of $\overline{G}$*}; stop;
3. construct the auxiliary graph $G'_v$ defined in Lemma 2.1 and, from that, its complement;
4. compute the strongly connected components of the graph $\overline{G'_v}$ and store them in the standard representative-based representation in an array `c[]`;
5. for each vertex $w$ in $V(G'_v)$ do `sccc[`$w$`]` $\leftarrow$ `c[`$w$`]`;
   for each vertex $w$ in $V(G) - V(G'_v)$ do `sccc[`$w$`]` $\leftarrow$ `c[`$v$`]`;

The above algorithm gives us a very simple s.c.co-components algorithm, which is also optimal. Indeed, because of Lemma 1.1(ii) (which implies that $\overline{G'_v}$ has $O(\sqrt{m})$ vertices, where $m$ is the number of edges of $G$) and the fact that the strongly connected components of a graph can be computed in time linear in the size of the graph, it is not difficult to see that:

**Theorem 2.1.** *Let $G$ be a directed graph on $n$ vertices and $m$ edges. Then, the algorithm Strong_Co-components computes the strongly connected components of $\overline{G}$ in $O(n+m)$ time.*

Using standard parallel algorithmic techniques and the CREW algorithm for computing the strongly connected components of a graph on $N$ vertices in $O(\log^2 N)$ time using $O(N^{2.376}/\log N)$ processors [1,13,15], we have:

**Theorem 2.2.** *Let $G$ be a directed graph on $n$ vertices and $m$ edges. Then, the strongly connected components of $\overline{G}$ can be computed in $O(\log^2 n)$ time using $O(m^{1.188}/\log n)$ processors on the CREW PRAM.*

Moreover, in light of the fact that the connected components of a graph $G$ are identical to the strongly connected components of the directed graph that results by replacing each undirected edge by two oppositely directed edges, a result similar to Lemma 2.1(ii) holds for an appropriate auxiliary graph on $O(\sqrt{m})$ vertices. Then, an algorithm similar to Strong_Co-components, along with the algorithm of Chong *et al.* [4] for computing the connected components of a graph on $N$ vertices in $O(\log N)$ time using $O(N^2/\log N)$ processors on the EREW PRAM, yield an optimal parallel co-connectivity algorithm simpler than the one in [6].

**Corollary 2.1.** *Let $G$ be an undirected graph on $n$ vertices and $m$ edges. Then, the connected components of $\overline{G}$ can be computed in $O(\log n)$ time using $O((n+m)/\log n)$ processors on the EREW PRAM.*

## 3 Biconnected Components of the Complement of a Graph

We next present an $O(n+m)$-time algorithm for computing the biconnected components of $\overline{G}$, which can be parallelized resulting in an algorithm that runs in $O(\log n)$ time using $O((n+m)/\log n)$ processors.

**Lemma 3.1.** *Let $G$ be an undirected graph on $m$ edges and $x$ be any of its vertices. If $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ are the connected components of the subgraph $\overline{G}[M(x)]$ induced by the set $M(x)$ of non-neighbors of $x$ in $G$, then* *(i)* *the vertex sets* $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ *are disjoint;*

*(ii)* *their number* $k$ *does not exceed* $2\sqrt{m}$*;*

*(iii)* *for each* $\mathcal{C}_i$*, the edge set of the subgraph* $\overline{G}[\mathcal{C}_i \cup \{x\}]$ *has the biconnectivity property in* $\overline{G}$*.*

**Lemma 3.2.** *Let* $G$ *be an undirected graph,* $v$ *a vertex of* $G$*,* $E_1, E_2, \ldots, E_\ell$ *the biconnected components of* $\overline{G}[N(v)]$ *with vertex sets* $V(E_1), \ldots, V(E_\ell)$ *respectively, and* $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ *the connected components of* $\overline{G}[M(v)]$*.*

*(i)* *If* $|E(G) \cap \{\, xy \mid x \in V(E_i),\ y \in M(v)\,\}| = |V(E_i)| \cdot |M(v)| - 1$*, then the two vertices* $u \in V(E_i)$ *and* $w \in M(v)$ *which are not adjacent in* $G$ *define a potential bridge in* $\overline{G}$*.*

*(ii)* *If there exists a vertex* $w \in M(v)$ *such that* $\{\, xy \mid x \in V(E_i),\ y \in M(v) - \{w\}\,\} \subseteq E(G)$ *and* $|\{\, xw \mid x \in V(E_i) \text{ and } xw \notin E(G)\,\}| \geq 2$*, then the edge set* $E_i \cup \{\, xw \mid x \in V(E_i) \text{ and } xw \notin E(G)\,\}$ *has the biconnectivity property in* $\overline{G}$ *and vertex* $w$ *is a potential articulation point in* $\overline{G}$*.*

*(iii)* *If there exists a vertex* $u \in V(E_i)$ *such that* $\{\, xy \mid x \in V(E_i) - \{u\},\ y \in M(v)\,\} \subseteq E(G)$ *and* $|\{\, uy \mid y \in M(v) \text{ and } uy \notin E(G)\,\}| \geq 2$*, then the edge set of the subgraph of* $\overline{G}$ *induced by* $\{v, u\} \cup \{\, \mathcal{C}_j \mid \exists y \in \mathcal{C}_j : uy \notin E(G)\,\}$ *has the biconnectivity property in* $\overline{G}$ *and vertex* $u$ *is a potential articulation point in* $\overline{G}$*.*

*(iv)* *If there exist vertices* $u, u' \in V(E_i)$ *and* $w, w' \in M(v)$ *such that* $uw,\ u'w' \notin E(G)$*, then the edge set of the subgraph of* $\overline{G}$ *induced by* $\{v\} \cup V(E_i) \cup \{\, \mathcal{C}_j \mid \exists x \in V(E_i) \text{ and } y \in \mathcal{C}_j : xy \notin E(G)\,\}$ *has the biconnectivity property in* $\overline{G}$*.*

In general terms, the algorithm works as follows: It finds a minimum-index vertex of $G$; let it be $v$. Next, it computes the biconnected components of $\overline{G}[N(v)]$ and the connected components of $\overline{G}[M(v)]$; recall that the edge set of the subgraph of $\overline{G}$ induced by each of the latter components and $v$ has the biconnectivity property in $\overline{G}$ (Lemma 3.1). Next, the algorithm takes advantage of Lemma 3.2 in order to do a first round of merging of the collected edge sets; to do that, it constructs a graph $\widetilde{G}$ in which the connected components indicate the sets to be merged. Additionally, it has collected potential articulation points and bridge endpoints of $\overline{G}$, from which it constructs another auxiliary graph $\widehat{G}$; the biconnected components of $\widehat{G}$ determine which edge sets will be merged in the second and final round of merging, which yields the biconnected components of $\overline{G}$.

The above algorithm gives us an optimal biconnected co-components algorithm, in light of Lemmas 1.1(i), 3.1, and 3.2 (which imply that the graphs

$\overline{G}[N(v)]$, $\widetilde{G}$, and $\widehat{G}$ have $O(\sqrt{m})$ vertices) and the fact that the connected and the biconnected components of a graph can be computed in time linear in the size of the graph. Thus, we have:

**Theorem 3.1.** *Let $G$ be an undirected graph on $n$ vertices and $m$ edges. Then, the algorithm Biconnected_Co-components computes the biconnected components of $\overline{G}$ in $O(n+m)$ time.*

Using standard parallel algorithmic techniques, the CREW algorithm for computing the biconnected components of a graph on $N$ vertices in $O(\log N)$ time using $O(N^2/\log N)$ processors [1,13,15], and the optimal co-connectivity algorithm of [6] (see also Corollary 2.1), we have the following theorem.

**Theorem 3.2.** *Let $G$ be an undirected graph on $n$ vertices and $m$ edges. Then, the biconnected components of $\overline{G}$ can be computed in $O(\log n)$ time using $O((n+m)/\log n)$ processors on the CREW PRAM.*

## References

[1] S.G. Akl, *Parallel Computation: Models and Methods*, Prentice Hall, 1997.

[2] B. Awerbuch and Y. Shiloach, New connectivity and MSF algorithms for ultra-computer and PRAM, *IEEE Trans. Computers* 36 (1987) 1258–1263.

[3] F.Y. Chin, J. Lam, and I. Chen, Efficient parallel algorithms for some graph problems, *Communications of the ACM* 25 (1982) 659–665.

[4] K.W. Chong, Y. Han, Y. Igarashi, and T.W. Lam, Improving the efficiency of parallel minimum spanning tree algorithms, *Discrete Applied Math.* 126 (2003) 33–54.

[5] K.W. Chong, Y. Han, and T.W. Lam, Concurrent threads and optimal parallel minimum spanning trees algorithm, *J. ACM* 48 (2001) 297–323.

[6] K.W. Chong, S.D. Nikolopoulos, and L. Palios, An optimal parallel co-connectivity algorithm, to appear in *Theory of Computing Systems*, 2004.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.

[8] E. Dahlhaus, J. Gustedt, and R.M. McConnell, Partially complemented representation of digraphs, *Descrete Math. and Theoret. Comput. Science* 5 (2002) 147–168.

[9] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[10] D.S. Hirschberg, Parallel algorithms for the transitive closure and the connected components problems, *Proc. 8th ACM Symp. on Theory of Computing (STOC'76)*, 55–57, 1976.

[11] D.S. Hirschberg, A.K. Chandra and D.V. Sarwate, Computing connected components on parallel computers, *Communications of the ACM* 22 (1979) 461–464.

[12] H. Ito and M. Yokoyama, Linear time algorithms for graph search and connectivity determination on complement graphs, *Inform. Process. Letters* 66 (1998) 209–213.

[13] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[14] D. Nath and S.N. Maheshwari, Parallel algorithms for the connected components and minimal spanning trees, *Inform. Process. Letters* 14 (1982) 7–11.

[15] J. Reif (ed.), *Synthesis of Parallel Algorithms*, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[16] Y. Shiloach and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm, *J. Algorithms* 3 (1982) 57–67.