August 1988

UILU-ENG-88-2247
ACT-97

COORDINATED SCIENCE LABORATORY
College of Engineering
Applied Computation Theory

VERTEX CONNECTIVITY OF GRAPHS: ALGORITHMS AND BOUNDS

Arkady Kanevsky

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-88-2247 (ACT-97)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION National Science Foundation Office of Naval Research Semiconductor Research Corporation
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 1800 G. St., Washington D.C. 10552 800 N. Quincy St., Arlington, VA 22217 P.O. Box 12053, Research Triangle Park 27709
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NSF, JSEP, SRC 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER ECS 8404866 , N00014-84-C-0149, SRC-87-DP-109PRE
8c. ADDRESS (City, State, and ZIP Code) 1800 G. St., Washington D.C. 10552 800 N. Quincy St., Arlington, VA 22217 P.O. Box 12053, Research Triangle Park, NC	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Vertex Connectivity of Graphs: Algorithms and Bounds

12. PERSONAL AUTHOR(S) Kanevsky, Arkady

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	August, 1988	130

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	graph theory, combinatorial algorithms, parallel algorithms, vertex connectivity, data structures

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report considers several problems concerning vertex connectivity of undirected graphs and presents new bounds and algorithms for these problems.

We have proved that the upper bound for the number of separating triplets of a triconnected graph is $\frac{(n-1)(n-4)}{2}$, and it exactly matches the lower bound, which is achieved by the wheel graph. This result has been generalized to an $O(2^k \frac{n^2}{k})$ upper bound on the number of separating k-sets in a k-connected graph. We have also obtained a new $\Omega(2^k \frac{n^2}{k^2})$ lower bound.

(over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

19. Abstract (continued)

Even though the upper bound for the number of separating k-sets is not linear but quadratic in n, we have obtained a linear space representation for the separating k-sets of a k-connected graph. For k=3 this representation is a collection of wheels, where every nonadjacent pair on the cycle of a wheel gives a separating triplet of a triconnected graph. For general k, we have obtained an $O(k^2n)$ representation.

We have designed a new sequential $O(n^2)$ algorithm for the problem of determining if the graph is four-connected or not. Consequently, we find all separating triplets of the graph if it is not four-connected. The algorithm has a parallel version which runs in $O(\log^2 n)$ time using $O(n^2)$ processors, which is also an improvement over $O(nm)$ processor count of the best previously known parallel algorithm.

We have designed algorithms for generating all separating k-sets of a k-connected graph. The sequential algorithm runs in $O(2^k n^3)$ time and parallel one runs in $O(k \log n)$ deterministic parallel time or in $O(\log^2 n)$ randomized time using $O(4^k \frac{n}{k^2})$ processors on a CRCW PRAM.

VERTEX CONNECTIVITY OF GRAPHS: ALGORITHMS AND BOUNDS

BY

ARKADY KANEVSKY

B.S., Univerisity of Illinois at Chicago, 1983
M.S., Univerisity of Illinois at Urbana-Champaign, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1988

Urbana, Illinois

VERTEX CONNECTIVITY OF GRAPHS: ALGORITHMS AND BOUNDS

Arkady Kanevsky, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1988

This thesis considers several problems concerning vertex connectivity of undirected graphs and presents new bounds and algorithms for these problems.

We have proved that the upper bound for the number of separating triplets of a triconnected graph is $\frac{(n-1)(n-4)}{2}$, and it exactly matches the lower bound, which is achieved by the wheel graph. This result has been generalized to an $O(2^k \frac{n^2}{k})$ upper bound on the number of separating k-sets in a k-connected graph. We have also obtained a new $\Omega(2^k \frac{n^2}{k^2})$ lower bound.

Even though the upper bound for the number of separating k-sets is not linear but quadratic in n, we have obtained a linear space representation for the separating k-sets of a k-connected graph. For $k=3$ this representation is a collection of wheels, where every nonadjacent pair on the cycle of a wheel gives a separating triplet of a triconnected graph. For general k, we have obtained an $O(k^2 n)$ representation.

We have designed a new sequential $O(n^2)$ algorithm for the problem of determining if the graph is four-connected or not. Consequently, we find all separating triplets of the graph if it is not four-connected. The algorithm has a parallel version which runs in $O(\log^2 n)$ time using $O(n^2)$ processors, which is also an improvement over $O(nm)$ processor count of the best previously known parallel algorithm.

We have designed algorithms for generating all separating k-sets of a k-connected graph. The sequential algorithm runs in $O(2^k n^3)$ time and parallel one runs in $O(k \log n)$ deterministic parallel time or in $O(\log^2 n)$ randomized time using $O(4^k \frac{n^6}{k^2})$ processors on a CRCW PRAM.

ACKNOWLEDGEMENTS

I express my deepest gratitude to my advisor, Professor Vijaya Ramachandran, who has had a profound influence on the development of my work, including this thesis.

Many thanks to the rest of my Ph. D. preliminary and final exam committee members: Professors H. Edelsbrunner, C. L. Liu, M. C. Loui, and E. Reingold for their constructive criticism.

At different stages of this thesis, I have benefited from many useful discussions with Rene Cruz, Dan Gusfield, Nany Hasan, Scot Hornick, David Luginbuhl, Sanjeev Maddila, David Muller, Franco Preparata, Harold Rosenberger, Steven Skiena, Roberto Tamassia, Prasoon Tiwari, Ioannis Tollis, Pravin Vaidya, Douglas West, and Dian Zhou.

This work has been partially supported, at various stages, by the National Science Foundation under Grant ECS 8404866, by the JSEP under Contract N00014-84-C-0149, and by Semiconductor Research Corporation under Contract ECS 87-DP-109.

I would like to thank my parents for their encouragement and support in my education. Finally, to my wife, [redacted] and daughter, [redacted] owe a heartfelt thanks for their constant support, understanding and unfaltering faith without which this thesis would have been impossible.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1. Motivations

Connectivity is one of the fundamental graph properties, and there has been a considerable amount of work on algorithms and structural aspects of this property. Applications of graph connectivity arise in operation research for scheduling problems, network analysis in electrical engineering, and many other real-life problems.

The most direct application of this problem is for the reliability of networks [3, 4, 40, 7, 8, 42, 43]. A fundamental criterion for evaluating performance of a communications network is its ability to withstand the failure of its components [3]. Two most important measures of network reliability are *reachability* and *connectedness* [4]. Usually, a network is viewed as an undirected connected graph in which both vertices and edges have some probability of failure. Most of the analyses of a network reliability are concerned with link (edge) failure, rather than vertex failure. The reason for ignoring vertex failure is not only real-life models but nonexistence of good upper bounds for the number of minimum size separating vertex sets of graphs. On the other hand it has been known that the number of separating edge sets is upper bounded by $\begin{bmatrix} n \\ 2 \end{bmatrix}$ [6, 30]. The list of all minimum size separating vertex sets and their number play a fundamental role in analyzing the connectedness of a network.

Another important measure of network reliability is to determine the subgraphs which are "highly" connected and to decompose the network into them. The results in all of these measures help in the design of optimum communication networks.

In recent years more and more network designs are based upon distributed networks (e.g. ARPANet) rather than tree type networks with multiedges [16, 2, 53, 54, 17]. These networks

are more reliable for node failure and use less hardware than tree type networks. Until recently most networks designed had small connectivity (1 or 2) [12, 2, 37, 38]. So the network (graph) can be easily decomposed into connected, biconnected or strongly connected components (directed case) [3, 28, 12, 37, 38]. This is needed in order to find which parts of the network can still communicate (with high probability) in case of edge and vertex failures. So in addition to connectedness, a relevant measure of network reliability is also what percent of nodes can communicate in case of failures of edges and vertices [38]. For more complex networks we need to find connected components with higher connectivity, along with all the separating vertex sets and edge sets of the network [1, 12, 37, 38, 32, 3, 34].

1.2. Previous Results

There are well-known sequential linear-time algorithms for determining vertex connectivity and biconnectivity (see e.g., [11]), as well as triconnectivity [16, 27]. These algorithms use either the depth-first search technique [11, 16, 35] or the ear-decomposition technique [39, 24, 25, 26]. The best deterministic sequential algorithms for testing graph 4-connectivity had time complexity $O(nm)$, where n is the number of vertices in the input graph and m is the number of edges. There are two such algorithms. One is based on a reduction to a network flow [9, 10, 14, 15]. The other uses the $O(m)$ algorithm for testing triconnectivity [16, 27] to test four-connectivity in a triconnected graph in $O(mn)$ time by deleting each vertex of the graph in turn, and testing triconnectivity in the resulting graph; this algorithm also finds all separating triplets in the graph, if the graph is not four-connected. The best sequential algorithm for general k is $O(\max(k, n^{1/2})kmn^{1/2})$ algorithm for determining the connectivity of a graph which is based upon a network flow [9, 10, 14, 15].

Efficient parallel algorithms were designed for determining graph connectivity for small k Clearly, there are NC algorithms for testing graph k-connectivity for any fixed k. Simply,

remove all k vertex subsets of the graph and test for graph connectivity, simultaneously on a CRCW PRAM. The best parallel algorithms for graph k-connectivity for $k = 1,2$ are the efficient $O(\log n)$ parallel time algorithms using $O(m+n)$ processors on a CRCW PRAM [45, 50], for $k = 3$ an $O(\log n)$ parallel time algorithm using $O((m+n)\log n)$ processors on a CRCW PRAM [36, 41] and for $k = 4$ an $O(\log n)$ parallel time algorithm using $O(nm)$ processors on a CRCW PRAM by using the triconnectivity algorithm by deleting each vertex of a graph in turn in parallel [41]. There is no efficient deterministic parallel algorithm for determining the connectivity of a graph for general k. We also note there are some randomized algorithms for testing k-connectivity for $k > 3$ [5, 29]; the running times of these algorithms are $O(n^{5/2} + nk^{5/2})$ [29], and $O(n^{3/2}m)$ [5].

The other question which often raised with connectivity is to find all minimum size separating vertex sets. This idea lies in the heart of the algorithms for determining graph k-connectivity for $k = 1,2,3,4$. The algorithms for graph (one)-connectivity, biconnectivity, triconnectivity and 4-connectivity find all articulation points, separating pairs, separating triplets of a graph in order to determine that a graph has a higher connectivity.

1.3. Main Results and Organization of the Thesis

Chapter 2 presents several definitions which will be used in the later chapters.

In chapters 3 and 4 we address the following question: what is the maximum number of separating k-sets in a k-connected undirected graph?

An undirected graph G on n vertices and m edges has for any k a trivial upper bound of $\binom{n}{k}$ on the number of separating k-sets. The graph on n vertices with no edges achieves this bound. However, the more interesting question is the one raised above, namely the bound when the graph is k-connected. For $k=1$ the maximum number of articulation points in an undirected connected graph is $(n-2)$, and a path on n vertices achieves this bound.

Chapter 3 presents a lower bound on the maximum number of separating k-sets in a k-connected undirected graph. It also presents the cycle and the wheel, the graphs that achieve the lower bounds for $k = 2,3$. For general k generalizations of cycle and wheel are presented. The lower bound obtained on the maximum number of separating k-sets of an undirected k-connected graph is $\Omega(2^k \frac{n^2}{k^2})$.

Chapter 4 presents the upper bound on the maximum number of separating k-sets in a k-connected undirected graph. For $k=2$ the maximum number of separating pairs in an undirected biconnected graph is $\frac{n(n-3)}{2}$, and a graph that achieves it is a cycle on n vertices [23]. For $k=3$ the maximum number of separating triplets in an undirected triconnected graph is $\frac{(n-1)(n-4)}{2}$, and a graph that achieves it is a wheel on n vertices [23].

Chapter 5 presents the $O(n^2)$ bound on the number of separating k-sets in a k-connected graph for any fixed k [24, 26]. The exact bound is $O(2^k \frac{n^2}{k})$. Furthermore, it presents a linear representation of separating k-sets in k-connected undirected graphs. There is an $O(n)$ representation for separating pairs in a biconnected graph [23] and there is an $O(k^2 n)$ compact representation for separating k-sets in a k-connected graph [26]. Table 1 summarizes these results.

Chapter 6 presents new sequential and parallel algorithms for graph four-connectivity. The new best deterministic sequential algorithm for testing graph four-connectivity has time complexity $O(n^2)$ [25] and is based upon ear-decomposition technique. The new efficient parallel algorithm for testing graph 4-connectivity runs in $O(\log^2 n)$ time using $O(n^2)$ processors on a CRCW PRAM [25]. Table 2 summarizes these results and their relationships to earlier results.

Chapter 7 presents sequential and parallel algorithms for finding all minimum size vertex separating sets for general k [27]. Table 3 presents the current time complexities of algorithms

Bounds for the number of separating k-sets in a k-connected graph on n vertices				
	Lower	Upper	Representation	
$k=1$	$n-2$	$n-2$	$O(n)$	
$k=2$	$\frac{n(n-3)}{2}$	$\frac{n(n-3)}{2}$	$O(n)$	Kanevsky Ramachandran [25] [Chapter 4]
$k=3$	$\frac{(n-1)(n-4)}{2}$	$\frac{(n-1)(n-4)}{2}$	$O(n)$	Kanevsky Ramachandran [25] [Chapter 4]
general k	$\Omega(2^k \frac{n^2}{k^2})$	$O(2^k \frac{n^2}{k})$	$O(k^2 n)$	Kanevsky [26] [Chapter 5]

Table 1.

Algorithms for determining graph connectivity in a k-connected n-node, m-edge graph					
	Sequential		Parallel		
	Time		Time	Processors	
$k=1$	$O(m+n)$		$O(\log^2 n)$ $O(\log n)$	CREW $O(m+n)$ CRCW	Hirschberg, Chandra,Sarwate [21] Shiloach, Vishkin [45]
$k=2$	$O(m+n)$	Tarjan [48]	$O(\log^2 n)$ $O(\log n)$	CREW $O(m+n)$ CRCW	Tsin, Chin [51] Tarjan, Vishkin [50]
$k=3$	$O(m+n)$	Hopcroft, Tarjan [22] Miller, Ramachandran [36]	$O(\log^2 n)$ $O(\log n)$	$O(m+n)$ CRCW $O((m+n)\log n)$	Miller, Ramachandran [36] Ramachandran, Vishkin [41]
$k=4$	$O(n^2)$	Kanevsky, Ramachandran [24] [Chapter 6]	$O(\log n)$ $O(\log^2 n)$	$O(mn)$ CRCW $O(n^2)$	Ramachandran, Vishkin [41] Kanevsky, Ramachandran [24] [Chapter 6]
	$O(\max(k,n^{1/2})kmn^{1/2})$	Galil [18]; Girkar,Sohoni [19]			

Table 2.

Algorithms for finding all separating k-sets

in a k-connected n-node, m-edge graph

(M is the number of separating k-sets)

	Sequential		Parallel		
	Time		Time	Processors	
$k=1$	$O(m+n)$	Tarjan [48]	$O(\log^2 n)$	CREW	Tsin,Chin [51]
			$O(m+n)$		
			$O(\log n)$	CRCW	Tarjan, Vishkin [50]
$k=2$	$O(m+n)$	Hopcroft, Tarjan [22]	$O(\log^2 n)$	$O(m+n)$	Miller, Ramachandran [36]
		Miller, Ramachandran [36]	$O(\log n)$	CRCW	Ramachandran, Vishkin [41]
				$O((m+n)\log n)$	
$k=3$	$O(n^2)$	Kanevsky, Ramachandran [24] [Chapter 6]	$O(\log n)$	$O(mn)$	Ramachandran, Vishkin [41]
				CRCW	
			$O(\log^2 n)$	$O(n^2)$	Kanevsky, Ramachandran [24] [Chapter 6]
general k	$\Theta(\min(Mnk + kmn\min(\sqrt{n},k)), (Mn + k^2n^3)) = O(2^k n^3)$	Kanevsky [27] [Chapter 7]	$O(k\log n)$ NC	$\Theta(M^2n^2 + kn^{3.376})$	Kanevsky [27]
				$O(4^k \frac{n^6}{k^2})$ CRCW	
			$O(\log^2 n)$ RNC	$\Theta(M^2n^2 + kn^{4.376})$	

Table 3.

for finding all minimum size separating vertex sets in an undirected k-connected graph.

Finally, in Chapter 8 we give the conclusion of the Thesis and state several open problems.

CHAPTER 2

DEFINITIONS

2.1. Graph Connectivity

An *undirected graph* $G=(V,E)$ consists of a *vertex set* V and an *edge set* E containing unordered pairs of distinct elements from V. A *path* P in G is a sequence of vertices $<v_0, \cdots, v_k>$ such that $(v_{i-1},v_i)\in E, i=1, \cdots, k$. The path P *contains* the vertices $v_0, \cdots, v_k$ and the edges $(v_0,v_1), \cdots, (v_{k-1},v_k)$ and has *endpoints* v_0, v_k, and *internal vertices* $v_1, \cdots, v_{k-1}$. The path P is a *simple path* if $v_0, \cdots, v_{k-1}$ are distinct and $v_1, \cdots, v_k$ are distinct. P is a *simple cycle* if it is a simple path and $v_0=v_k$. A single vertex is a trivial path with no edges. We denote by $|P|$, the number of vertices contained in path P.

Let $P=<v_0, \cdots, v_{k-1}>$ be a simple path. The path $P(v_i,v_j), 0\le i,j\le k-1$ is the simple path connecting v_i and v_j in P, i.e., the path $<v_i,v_{i+1}, \cdots, v_j>$, if $i\le j$ or the path $<v_j,v_{j+1}, \cdots, v_i>$, if $j<i$. Analogously, $P[v_i,v_j]$ consists of the path segments obtained when the edges and internal vertices of $P(v_i,v_j)$ are deleted from P.

Let $G=(V,E)$ be an undirected graph and let $V'\subseteq V$. A graph $G'=(V',E')$ is a *subgraph* of G if $E'\subseteq E\cap\{(v_i,v_j)\,|\,v_i,v_j\in V'\}$. The *subgraph of G induced by V'* is the graph $G''=(V',E'')$ where $E''=E\cap\{(v_i,v_j)\,|\,v_i,v_j\in V'\}$.

We will sometimes specify a graph G structurally without explicitly defining its vertex and edge sets. In such cases, $V(G)$ will denote the vertex set of G and $E(G)$ will denote the edge set of G. Also, if $V'\subseteq V$ and $v\in V$ we will use the notation $V'\cup v$ to represent $V'\cup\{v\}$.

An undirected graph $G=(V,E)$ is connected if there exists a path between every pair of vertices in V. For a graph G that is not connected, a *connected component* of G is an induced subgraph of G which is maximally connected.

A vertex $v \in V$ is an *articulation point* (a.p.) of a connected undirected graph $G=(V,E)$ if the subgraph induced by $V-\{v\}$ is not connected. G is *biconnected* if it contains no articulation point.

Let $G=(V,E)$ be a biconnected undirected graph. A pair of vertices $v_1, v_2 \in V$ is a *separating pair* for G if the induced subgraph on $V-\{v_1,v_2\}$ is not connected. G is *triconnected* if it contains no separating pair.

A triplet (v_1,v_2,v_3) of distinct vertices in V is a *separating triplet* of a triconnected graph if the subgraph induced by $V-\{v_1,v_2,v_3\}$ is not connected. G is *four-connected* if it contains no separating triplets.

In general, an undirected graph is k-connected if and only if between every pair of vertices there are k vertex disjoint paths, or alternatively, removal of any $k-1$ vertices leaves a graph connected [14]. The equivalence of these two definitions is the well-known Menger theorem [34].

Let $G=(V,E)$ be a k-connected undirected graph. A set V' of k distinct vertices of G is a separating k-set if the subgraph induced on $V-V'$ is not connected.

2.2. Ear Decomposition of Graphs

An *ear decomposition* [31, 55] $D=[P_0, \cdots, P_{r-1}]$ of an undirected graph $G=(V,E)$ is a partition of E into an ordered collection of edge disjoint simple paths $P_0, \cdots, P_{r-1}$ such that P_0 is a simple cycle and each endpoint of $P_i, i=1, \cdots, r-1$ is contained in some $P_j, j<i$, while none of the internal vertices of P_i are contained in any $P_j, j<i$. The P_i's are called the *ears* of D. D is an *open ear decomposition* if none of the $P_i, i=1, \cdots, r-1$ is a simple cycle. A *trivial ear* is an ear consisting of a single edge. A graph has an open ear decomposition if and only if it is biconnected [55].

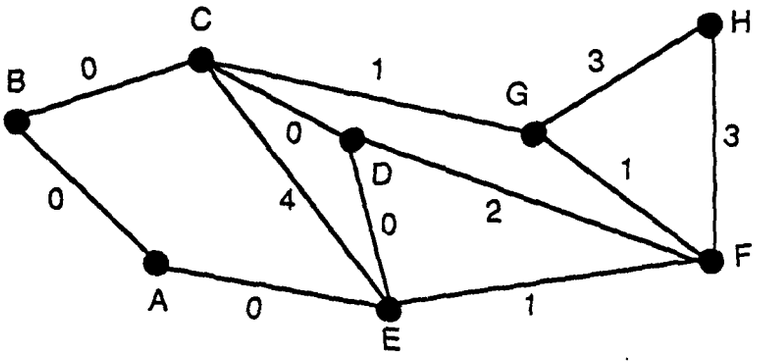
Let $G=(V,E)$ be a biconnected graph, and let Q be a subgraph of G. We define the *bridges of Q in G as follows* (see also [14]): Let V' be the vertices in $G-Q$, and consider the partition of V' into classes such that two vertices are in the same class if and only if there is a path connecting them which does not use any vertex of Q. Each such class K defines a *(nontrivial) bridge* $B=(V_B,E_B)$ of Q, where B is the subgraph of G with $V_B=K\cup$ {vertices of Q that are connected by an edge to a vertex in K}, and E_B containing the edges of G incident on a vertex in K. The vertices of Q which are connected by an edge to a vertex in K are called the *attachments* of B. An edge (u,v) in $G-Q$, with both u and v in Q, is a *trivial bridge* of Q, with attachments u and v. The nontrivial and trivial bridges together form the bridges of Q in G. In general, wherever we use the term bridge, we mean nontrivial bridge.

Let $G=(V,E)$ be a biconnected graph, and let Q be a subgraph of G. We define the *bridge graph of Q, $S=(V_S,E_S)$* as follows: Let the bridges of Q in G be $B_i, i=1,\cdots,k$. Then $V_S=V(Q)\cup\{B_1,\cdots,B_k\}$ and $E_S=E(Q)\cup\{(v,B_i)\mid v\in V(Q), 1\le i\le k$, and v is an attachment of $B_i\}$.

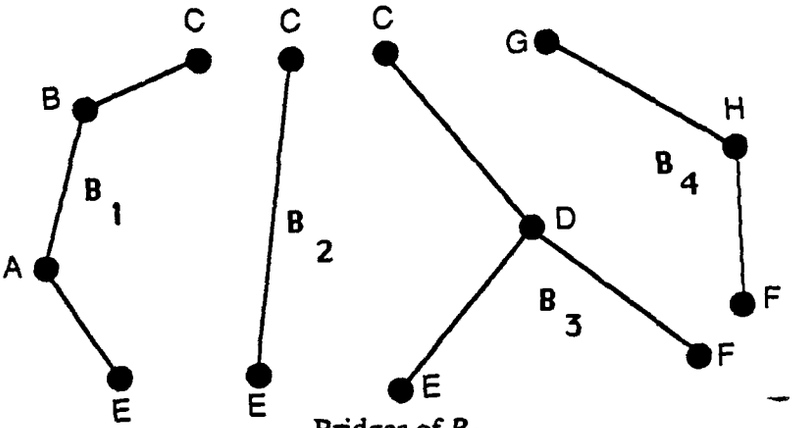
Figure 2.1 illustrates some of our definitions relating to bridges.

Let $G=(V,E)$ be a graph and let P be a simple path in G. If each bridge of P in G contains exactly one vertex not on P, and there is a bridge B of P with the endpoints of P as attachments, then we call G the *star graph of P* and denote it by $G(P)$. We denote the bridges of $G(P)$ by *stars*. The unique vertex of a star that is not contained in P is called its *center*. Note that, in a connected graph G, the bridge graph of any simple path in G is a star graph. Let $G(P)$ be a star graph, and let $S_1,\cdots,S_k$ be some of the stars in $G(P)$. The operation of *coalescing* the stars $S_i, i=1,\cdots,k$ removes these stars and replaces them by a new star S whose attachments are the union of the attachments of $S_1,\cdots,S_k$.

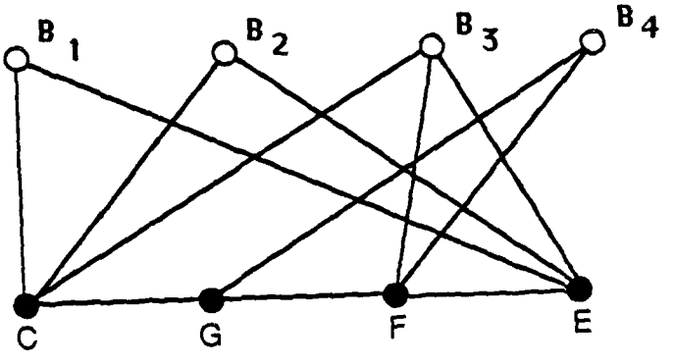
Let G be a biconnected graph with an open ear decomposition $D=[P_0,\cdots,P_{r-1}]$. Let the bridges of P_i in G that contain vertices on ears numbered lower than i be $B_{r_1},\cdots,B_{r_t}$. We shall



G with open ear decomposition $D = [P_0, P_1, P_2, P_3, P_4]$;
P_0=<1,2,3,4,5,1>, P_1=<3,7,6,5>, P_2 = <6,4>, P_3=<7,8,6>, P_4=<3,5>.



Bridges of P_1.



Bridge graph G_1 of P_1.

Figure 2.1.
Illustrating the ear decomposition

call these the *anchor bridges of* P_i. The *ear graph of* P_i, denoted by $G_i(P_i)$ is the graph obtained from the bridge graph ot P_i by coalescing all stars corresponding to anchor bridges. We will call

this coalesced star, the *anchoring star* of $G_i(P_i)$. For any two vertices x,y on P_i, we denote by $V_i(x,y)$, the internal vertices of $P_i(x,y)$; we denote by $V_i[x,y]$, the vertices in $P_i[x,y]-\{x,y\}$ together with the vertices in anchor bridges. For a star graph $G(P)$, the set $V(x,y)$ denotes the vertices in $P(x,y)-\{x,y\}$, and the set $V[x,y]$ denotes the vertices in $P[x,y]-\{x,y\}$.

2.3. Network Flows and Graph Connectivity

A *network* consists of the following data:

(1) A finite directed graph $G(V,E)$ with no self-loops and no multiple edges.

(2) Two specified vertices s and t; s is called *source* and t, the *sink*

(3) Each edge $e\in E$ is assigned a non-negative number $c(e)$ called the *capacity* of e.

A *flow function* f is an assignment of a real number $f(e)$ to each edge e, such that the following two conditions hold:

(1) For every edge $e\in E$, $0\le f(e)\le c(e)$

(2) Let $\alpha(v)$ and $\beta(v)$ be the sets of edges incoming to vertex v and outgoing from v, respectively. For every vertex $v\in V-\{s,t\}$

$$0=\sum_{e\in\alpha(v)} f(e)-\sum_{e\in\beta(v)} f(e).$$

The *total flow* F of f is defined by

$$F=\sum_{e\in\alpha(t)} f(e)-\sum_{e\in\beta(t)} f(e).$$

The *maximum flow problem* is to find an f for which the total flow is maximum.

Let S be a subset of vertices such that $s\in S$ and $t\notin S$. Let $\bar{S}$ be the complement of S, i.e. $\bar{S}=V-S$. Let $(S,\bar{S})$ be the set of edges of G whose start vertex is in S and end vertex is in $\bar{S}$. The set $(\bar{S},S)$ is defined similarly. The set of edges connecting vertices of S with $\bar{S}$ (in both directions) is called the *cut* defined by $(S,\bar{S})$.

Lemma 2.1. [15] For every S

$$F = \sum_{e \in (S,\bar{S})} f(e) - \sum_{e \in (\bar{S},S)} f(e).$$

Let us denote by $c(s)$ the *capacity of the cut* determined by $(S,\bar{S})$ which is defined as follows:

$$c(S) = \sum_{e \in (S,\bar{S})} c(e).$$

Lemma 2.2. [15] For every flow function f, with total flow F, and every S,

$$F \leq c(S).$$

By the capacity constraint, the flow across any cut cannot exceed the capacity of the cut. Thus the value of the maximum flow is no greater than the capacity of a minimum cut. The *max–flow min–cut theorem* states that these two numbers are equal.

Theorem 2.1. [15] If F and S are such that $F = c(S)$ then F is the maximum and the cut defined by S is of the minimum capacity.

The *residual capacity* for a flow f is the function on vertex pairs given by $r(v,w) = c(v,w) - f(v,w)$. We can push up to $r(v,w)$ additional units of flow from v to w by increasing $f(v,w)$ and correspondingly decreasing $f(w,v)$. The *residual graph* R for a flow f is the directed graph with vertex set V, source s, sink t, and an edge (v,w) of capacity $r(v,w)$ for every pair v,w such that $r(v,w) > 0$. An *augmenting path* for f is a simple directed path from s to t in R. There are several algorithms for finding the network flows in a digraph [14, 49, 32, 20] using augmenting path to increase flow. But for faster algorithms for network flows *blocking flow* technique is used [10, 49].

There are special types of networks which are often used: *a unit network* is the network with all edge capacities integers and each vertex v other than s and t has either a single incoming edge, of capacity one, or a single outgoing edge, of capacity one.

Theorem 2.2. [49] On a unit network, Dinic's algorithm finds a blocking flow in $O(m)$ time and a maximum flow in $O(\sqrt{n}m)$ time.

There is a direct connection between a network flows and connectivity of an undirected graph [14]. Let $N(v,w)$ be the least cardinality vertex separator between v and w (a smallest cardinality vertex set S such that there are no path from v to w in $V-S$), and let $p(v,w)$ be the maximum number of pairwise vertex disjoint paths connecting v and w in G.

Lemma 2.3. [14, 34] (Menger's Theorem) If $(v,w) \notin E$ then $N(v,w) = p(v,w)$.

Based upon this lemma we construct a directed graph $\overline{G}$ from undirected G and find a maximum flow between kn pairs of vertices in order to determine the minimum $p(v,w)$ in G [12].

Theorem 2.3. [14] Connectivity k of G is equal to $\min_{v,w} p(v,w)$.

First, we construct a digraph $\overline{G} = (\overline{V},\overline{E})$ as follows. For every vertex $v \in V$ there are two vertices v' and v'' in $\overline{V}$ with a directed edge $(v',v'') \in \overline{E}$. For every edge $(u,v) \in E$, there are two edges $(u'',v') \in \overline{E}$ and $(u',v'') \in \overline{E}$. Define now a network with digraph $\overline{G}$, source s'', sink t', unit capacity for all *internal* edges (edges of the form (v',v'')) and infinite capacity edges for all other edges (external) of $\overline{G}$ [14] (see figure 2.2).

In order to find connectivity of G maximum flow has to be found between kn pairs of vertices of $\overline{G}$ [14]. Thus gives $O(k^2mn)$ time algorithm for finding the connectivity of a graph [14]. More refined algorithms for this problem have $O(max(k,\sqrt{n})km\sqrt{n})$ time [18, 19], where the algorithm in [19] has the best space bound. For more information on the use of network flows for graph connectivity see [14].

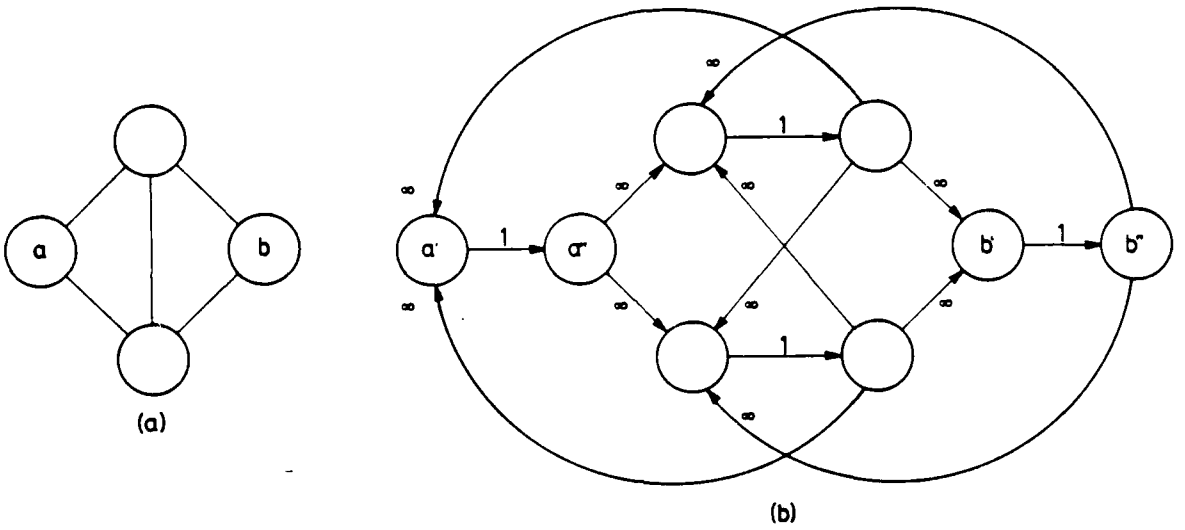


Figure 2.2.
Illustrating the reduction from G to $\overline{G}$.

2.4. Models of Parallel Computation

The model of parallel computation that we will be using is the *PRAM* model, which consists of several independent sequential processors, each with its own private memory, communicating with one another through a global memory. In one unit of time, each processor can read one global or local memory, execute a single RAM operation, and write into one global or local memory location in that order.

PRAMs are classified according to restrictions on global memory access. An EREW PRAM is a PRAM for which simultaneous access to any memory location by different processors is forbidden for both reading and writing. In a CREW PRAM simultaneous reads are allowed but no simultaneous writes. A CRCW PRAM allows simultaneous reads and writes. In this case we have to specify how to resolve write conflicts. We will use the ARBITRARY model in which any one processor participating in a concurrent write may succeed, and the algorithm should

work correctly regardless of which one succeeds. Of the three PRAM models we have listed, the EREW model is the most restrictive, and the ARBITRARY CRCW model is the most powerful. It is not difficult to see that any algorithm for the ARBITRARY CRCW PRAM that runs in parallel time T using P processors can be simulated by an EREW PRAM (and hence by a CREW PRAM) in parallel time $T\log P$ using the same number of processors, P (see e.g., [28]).

Let S be a problem which, on an input of size n, can be solved on a PRAM by a parallel algorithm in parallel time $t(n)$ with $p(n)$ processors. The quantity $w(n)=t(n)\cdot p(n)$ represents the *work* done by the parallel algorithm. Any PRAM algorithm that performs work $w(n)$ can be converted into a sequential algorithm running in time $w(n)$ by having a single processor simulate each parallel step of the PRAM in $p(n)$ time units. More generally, a PRAM algorithm that runs in parallel time $t(n)$ with $p(n)$ processors also represents a PRAM algorithm performing $O(w(n))$ work for any processor count $P<p(n)$.

Define $polylog(n)=\bigcup_{k>0} O(\log^k n)$. Let S be a problem for which currently the best sequential algorithm runs in time $T(n)$. A PRAM algorithm A for S, running in parallel time $t(n)$ with $p(n)$ processors is *efficient* if

a) $t(n)=polylog(n)$; and

b) the work $w(n)=p(n)\cdot t(n)$ is $T(n)\cdot polylog(n)$.

An efficient parallel algorithm is one that achieves a high degree of parallelism and comes to within a polylog factor of optimal speed-up. A major goal in the design of parallel algorithms is to find efficient algorithms with $t(n)$ as small as possible. The simulations between the various PRAM models make the notion of an efficient algorithm invariant with respect to the particular PRAM model used. For more on the PRAM model and PRAM algorithms, see [28].

CHAPTER 3

LOWER BOUNDS FOR THE NUMBER OF SEPARATING K-SETS

3.1. Cycle and Wheel for k=2,3

The n-node graph that achieves the maximum number of articulation points of a connected graph is a *path* P_n on n vertices. The *cycle* C_n is a simple path $P_n = <v_0, \cdots, v_n, v_0>$ on n vertices such that its endpoints coincide (see Figure 3.1). It has $\frac{n(n-3)}{2}$ separating pairs, which is a lower bound for the maximum number of separating pairs for a biconnected graph on n vertices.

The *wheel* W_n [52] is a cycle C_{n-1} together with a vertex v and an edge between v and every vertex on C_{n-1} (see Figure 3.2). After removal of v and all edges adjacent to it we get C_{n-1}, which yields $\frac{(n-1)(n-4)}{2}$ separating triplets. And removal of any three vertices of C_{n-1}

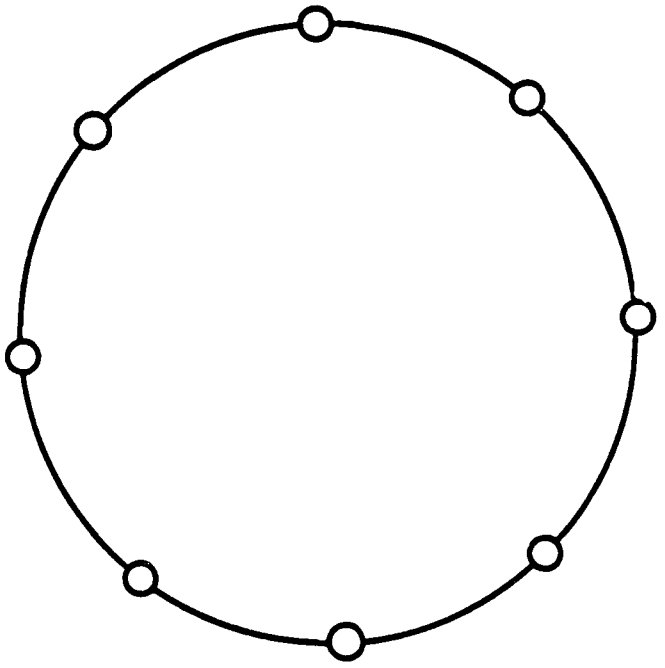


Figure 3.1.
Cycle.

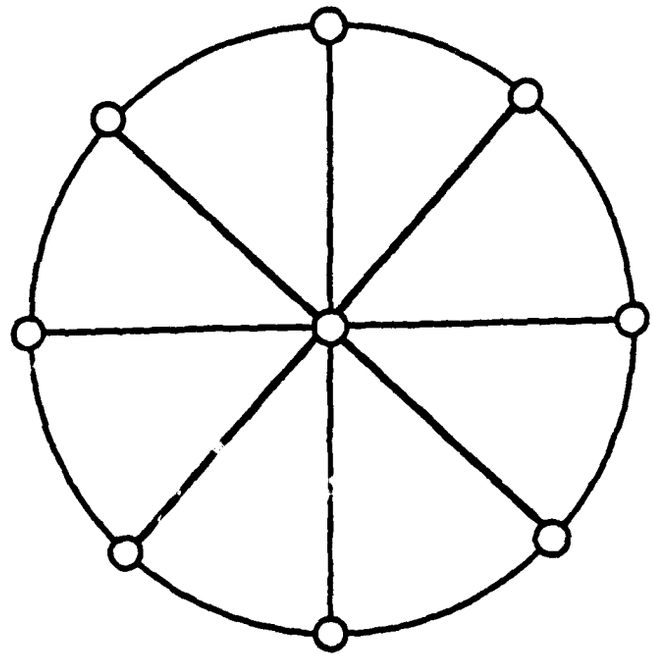


Figure 3.2.
Wheel

does not disconnect the wheel. Hence the number of separating triplets of a wheel is $\frac{(n-1)(n-4)}{2}$. This is a lower bound for the maximum number of separating triplets of a tricon-nected undirected graph on n vertices.

3.2. Generalized Cycle and Wheel for General $k>3$

Let us now generalize the *wheel* graph and the *cycle* graph to achieve lower bounds for the number of separating k-sets for odd and even k, respectively. (see Figure 3.3).

For even k take $\frac{n}{k}$ complete graphs K_k on k vertices, arranged in a cycle. Take $\frac{k}{2}$ vertices of each K_k. Two adjacent complete graphs are connected via $\frac{k}{2}$ edges, one edge per vertex, such that every vertex of K_k has one and only one edge outside K_k. Removal of these $\frac{k}{2}$ edges and analogous $\frac{k}{2}$ edges which connect two other adjacent complete graphs on the 'cycle' will

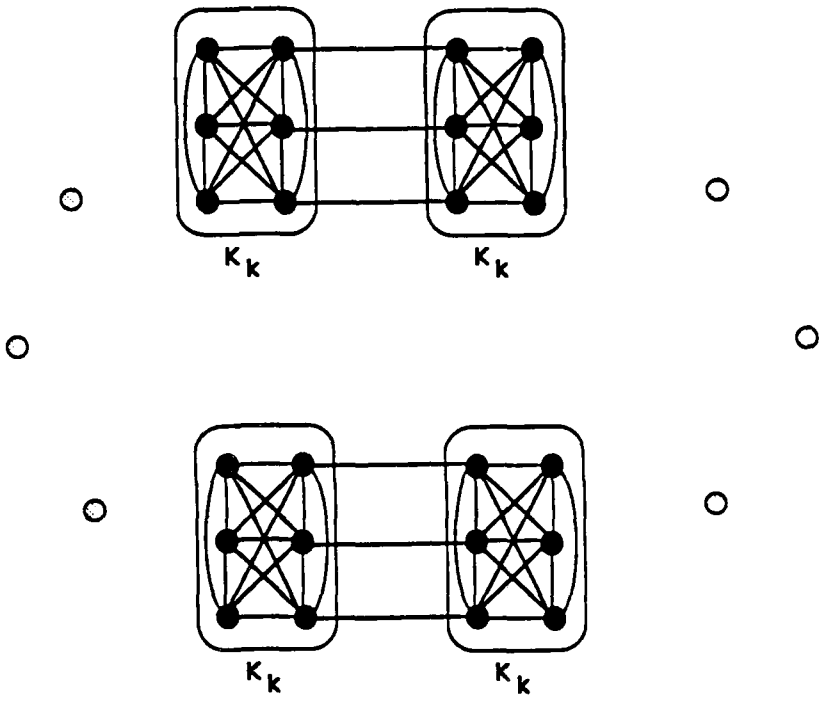


Figure 3.3.
Generalization of cycle for even k ($k = 6$).

separate the graph. Since for removal of each edge we can use either of its endpoints we get 2^k factor for each such separating edge set. Since a cycle has $\frac{n\,(n-3)}{2}$ separating pairs, we get $\Omega(\frac{n^2}{k^2})$ separating edge sets of cardinality k. Hence, the number of separating k-sets for the *generalized cycle* graph is $\Omega\left(2^k \frac{n^2}{k^2}\right)$. If n is not divisible by k then one complete graph will be bigger than k in order to take extra (n *mod* k) vertices. But the number of the edges between two adjacent complete graphs on the cycle still remains $\frac{k}{2}$.

The generalized cycle is a k-connected graph. To see this consider any two vertices v_1 and v_2 of it. There are two cases: they belong to two different complete subgraphs on k vertices (K_1

and K_2), or they belong to the same one K_3. If it is the first case, then clearly there are $\frac{k}{2}$ pairwise disjoint paths between v_1 and v_2 going clockwise on the generalized cycle, and there are $\frac{k}{2}$ paths between v_1 and v_2 going counterclockwise on the generalized cycle. If it is the second case, then there is one path which is just a single edge, $k-2$ paths inside K_3 and one path which connects v_1 and v_2 via the rest of the generalized cycle.

Analogously, we have the lower bound for odd k, which is achieved by a *generalized wheel*. A generalized wheel is a generalized cycle on $n-1$ vertices and one vertex in a center which is connected to every vertex on a generalized cycle (see Figure 3.4).

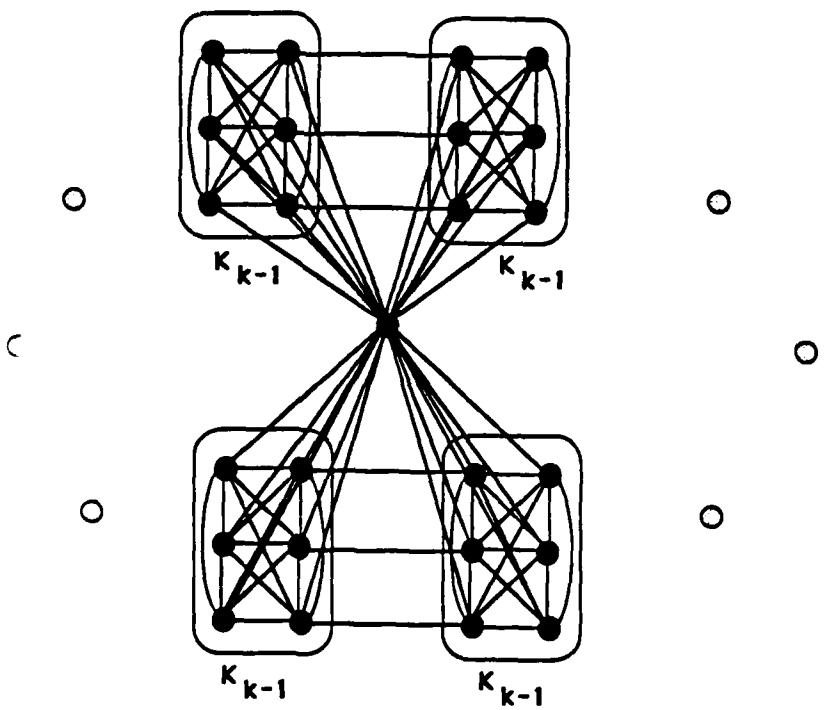


Figure 3.4.
Generalization of wheel for odd k ($k = 7$).

CHAPTER 4

UPPER BOUNDS AND REPRESENTATIONS FOR SEPARATING PAIRS AND TRIPLETS

In this and the following Chapter we present compact representations for separating k-sets of a graph as well as upper bounds for the maximum number of separating k-sets in an n-node k-connected graph. By *representation of separating k-sets* we mean a data structure that takes less space than the input graph itself and for which there is a fast procedure to list all separating k-sets of the graph; when computing the space required for a representation, we assume that we are using one unit of space for each vertex, and for each edge.

In this chapter we provide upper bounds and representations for $k = 2$ and $k = 3$. In Chapter 5 we generalize these techniques for general k.

4.1. Separating Pairs

4.1.1. $\frac{n(n-3)}{2}$ Upper Bound

Theorem 4.1. The maximum number of separating pairs in an undirected biconnected n-node graph is $\frac{n(n-3)}{2}$.

Proof: Let $\{v_1, v_2\}$ be a separating pair of a biconnected graph G on n vertices and m edges, whose removal separates G into nonempty G_1 and G_2 (see Figure 4.1).

Then we can divide all separating pairs of G into four types:

1). Separating pairs completely inside $G_1 \cup \{v_1, v_2\}$,

2). Separating pairs completely inside $G_2 \cup \{v_1, v_2\}$,

3). Separating pairs with one vertex from G_1 and one vertex from G_2,

4). The separating pair $\{v_1, v_2\}$.

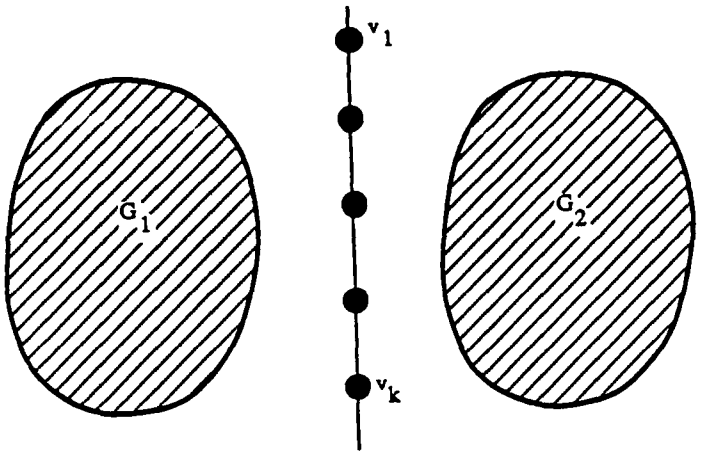


Figure 4.1.
Separating G into nonempty G_1 and G_2 by separating pair $\{v_1, v_2\}$

Let $f(n)$ be the maximum number of separating pairs in a biconnected graph on n vertices. The number of separating pairs of type one and type two are upper bounded by $f(l+2)$ and $f(n-l)$, respectively, where l is the cardinality of $V(G_1)$ and $n-l-2$ is the cardinality of $V(G_2)$. Let us denote the separating pairs of type third as the *cross separating pairs*. The number of separating pairs of type three is trivially upper bounded by $l(n-l-2)$. Hence, any function $f(n)$ that satisfies the recurrence

$$f(n) \leq \max_{1 \leq l \leq n-3} \left\{ f(l+2) + f(n-l) + l(n-l-2) + 1 \right\}.$$

is an upper bound on the number of separating pairs in a biconnected graph on n vertices.

We note that this is the recurrence for the cycle and that the recurrence implies that $f(n) \leq \frac{n(n-3)}{2}$. Combining this upper bound with the lower one found in Chapter 3, we get that $f(n) = \frac{n(n-3)}{2}$.

□ Theorem 4.1.

4.1.2. O(n) Representation by Cycles

Even though the number of separating pairs in a biconnected n-node graph $G = (V,E)$ can be as large as $\Theta(n^2)$, we observe that there are more succinct representations for them.

1). The *tree of triconnected components* of a biconnected graph has size $O(m+n)$, where $|E| = m$ [22, 36], and this is a representation for all separating pairs together with the triconnected components of the graph.

2). The algorithm in [36] enumerates the separating pairs as a collection $C = \{V_1, \cdots, V_t\}$ of subsets of V, with the interpretation that any pair of vertices within a single V_i is either a separating pair for G or the endpoints of an edge in a specified 'ear' in G, and further, every separating pair for G appears in at least one of the V_i's. We show below that $\sum_{i=1}^{t} |V_i| = O(n)$; thus this gives an $O(n)$ representation for separating pairs.

Let us first look at the subsets V_i (cycles) which are the representation of separating pairs. Look at the star embedding which we get for each nontrivial ear of an open ear decomposition of a biconnected graph (see Figure 4.2). If there is a separating pair of G then it belongs to some nontrivial ear [36].

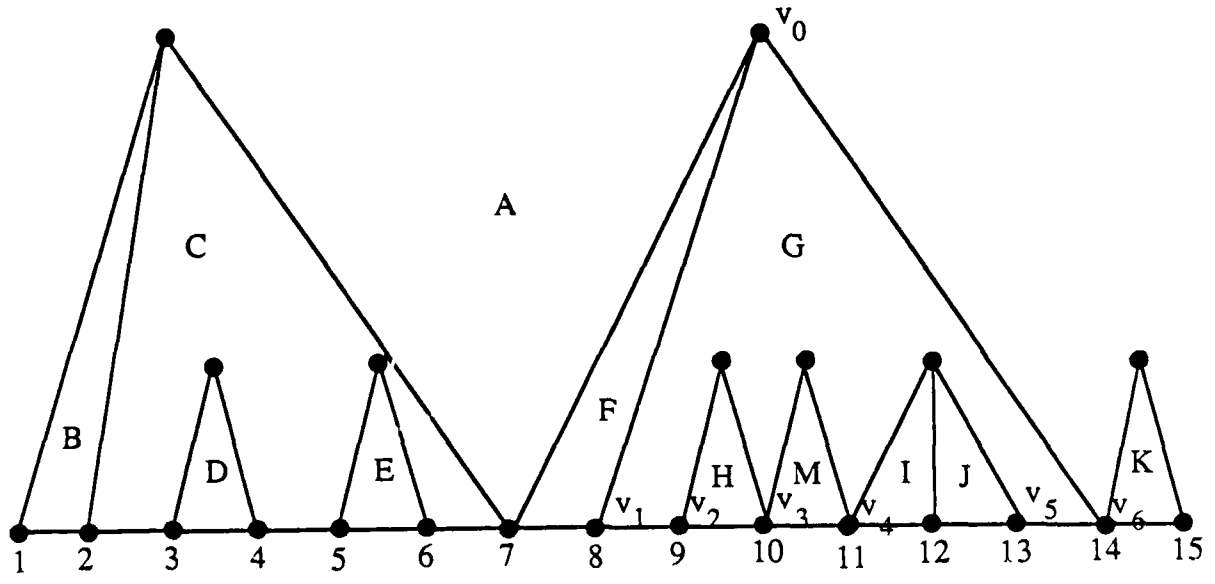


Figure 4.2.
Star embedding

Let p be the number of vertices of a nontrivial ear P. Let L be a planar region of the star embedding of P. L consists of the set of vertices in the stars of P and all other vertices $(v_1, \cdots, v_x)$ on P. It divides P into a collection of paths P' $(1\text{–}v_1, v_1\text{–}v_2, \cdots, v_x\text{–}p)$, including trivial paths of length 0 for the vertices of L on P which are the endpoints of two other regions. We will denote each of the trivial paths by a single vertex. For example planar region G of star embedding of P divides P into paths: (1–8,8–9,9–10,10,10–11,11,11–13,13–14,14–15). Each vertex of P belong to at most 3 of these paths. Let group L_1 be the union of those paths of P which belong to L (including trivial paths). Let L_2 be the union of $1\text{–}v_1$ and $v_x\text{–}p$. Let groups $L_3, \cdots, L_s$ be the other paths. For example if G is the planar region of P then L_1 is 8–9∪10∪11∪13–14, L_2 is 1–8∪14–15, L_3 is 9–10, L_4 is 10–11 and L_5 is 11–13. All vertices of P of each remaining planar region of the star embedding of P belong to L_i for some i. Let p_i be the number of vertices of L_i, then $\sum_{i=1}^{s} p_i \leq p + 2(s-1)$. This is true because every vertex of P is counted once unless it belongs to L and some other group, then it is counted at most 3 times. Let $r(p)$ be the number of vertices of P in all planar regions of the star embedding of P including repetitions. Let $r(p_i)$ be the number of vertices of P in all planar regions of star embedding of P which uses only vertices of L_i. Then,

$$r(p) \leq \max_{s}\left(\sum_{i=1}^{s} r(p_i)\right)$$

We can show that this inequality implies that $r(p) \leq 3p - 6$ (see Appendix 4.1.3). Note that if L has only two adjacent vertices on P (like D) then we get $r(p) \leq r(p) + r(2)$, where $r(2) = 2$. But there are at most p of them. And for all other L's $r(p_i) < r(p)$ for all i's. Hence, all separating pairs which belong to ear P have $O(p)$ size representation. Summing over all nontrivial ears we get an $O(n)$ size representation for the separating pairs of a graph.

3). There is another representation of separating pairs of G. It is based upon the decomposition of a biconnected graph into a collection of cycles. Actually both representations are the

same.

Let $G = (V,E)$ be an undirected biconnected graph with n vertices and m edges. We denote by $g(n)$ the upper bound on the size of a compact representation of the separating pairs of a biconnected graph on n vertices. Let $\{v_1,v_2\}$ be a separating pair that divides G into nonempty G_1 and G_2. Let $\{w_1,w_2\}$ be a separating pair of the third type with $w_1 \in G_1$ and $w_2 \in G_2$.

Consider a maximal set of vertices u in G_2 such that $\{w_1,u\}$ is a cross separating pair and, analogously, consider a maximal set of vertices x in G_1 such that $\{x,w_2\}$ is a cross separating pair. The set of u's is the set of articulation points in G_2. Moreover, the set of u's along with the subgraphs of G_2 between them form a path between v_1 and v_2. Analogously, the set x's is a set of articulation points of G_1. And the set of x's along with the subgraphs of G_1 between them form a path between v_1 and v_2. Number the vertices v_1, u's, v_2, and x's by y_1, y_2 and so on going clockwise along these paths. We denote by G_i the subgraph of G between y_i and y_{i+1} (the last G_i is between y_i and y_1). Note that some G_i can be empty (consists of a single edge). Thus, the graph G becomes a cycle with vertices y's and G_i's alternating on it (see Figure 4.3).

Every pair of vertices y_i and $y_j, j>i$ gives a separating pair of G unless $j=i+1$ and the subgraph G_i between them is empty. Hence, we can represent all separating pairs of this form by the following structure (cycle):

1) the set of vertices y's,

2) a vertex for every G_i with the flag to specify if G_i is empty,

3) edges between G_i and y_i, y_{i+1}.

Note that when there are no cross separating pairs in G then we get a trivial cycle with two vertices v_1 and v_2, two vertices G_1 and G_2, and four edges connecting them. Since the sets x's and u's are the maximal sets, all other separating pairs of G are inside some $G_i \cup y_i \cup y_{i+1}$. Note that G_i can be the union of disconnected components, but each of them is connected to y_i and y_{i+1}. Let the cardinality of the set of y's be l. Let n_i be the cardinality of G_i, and $\sum_{i=1}^{l}(n_i + 1) = n$.

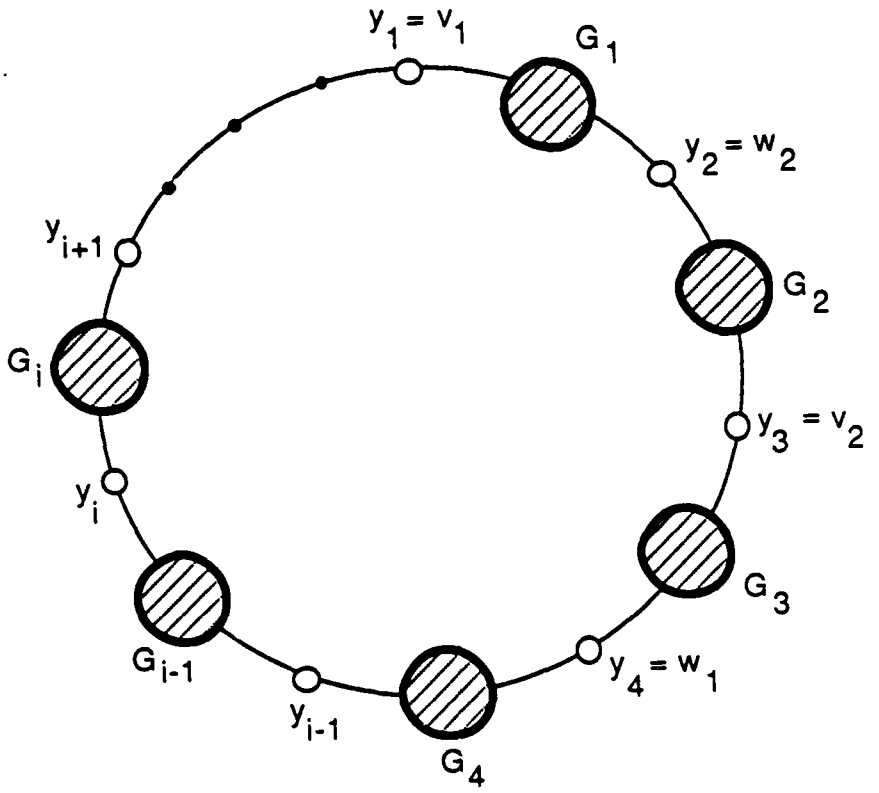


Figure 4.3.
Representation for k=2.

Based upon the above observations we get the following recurrence relation

$$g(n) \leq \max_{l}\left(\sum_{i=1}^{l} g(n_i + 2) + 3l\right),$$

where $g(n_i + 2)$ represents the upper bound for all separating pairs of G inside $G_i \cup y_i \cup y_{i+1}$. We assume that we are using one unit of space for each G_i. The space to represent the cycle is $3l$: one l for the set of y_i's, one l for the set of G_i's, and one l for the set of the flags of G_i's. Any $g(n)$ that satisfy this recurrence will be an upper bound on the size of representation of separating pairs of G. Clearly, linear $g(n)$ is one of them (see Appendix 4.1.3).

Using the above representation we can list all separating pairs in a biconnected graph in $O(M + n)$ time, where M is the number of separating pairs in a graph. For each cycle every pair of vertices of the form (y_i, y_j), $i < j$ give a separating pair of a graph, unless $j = i + 1$ and there is a flag for the vertex for the subgraph G_i specifying that it is empty. Note that some of the separating pairs can be repeated.

4.1.3. Appendix

1). Solution to the recurrence $r(p) \le \max_s(\sum_{i=1}^{s} r(p_i))$ for the second representation of separating pairs with restrictions

$$\sum_{i=1}^{s} p_i \le p + 2s - 2 \qquad 2 \le s \le p.$$

Let $r(p) = 3p - 6$, then

$$3p - 6 = r(p) \le \max_s(\sum_{i=1}^{s} r(p_i)) = \max_s(\sum_{i=1}^{s} (3p_i - 6)) \le \max_s(3p + 6s - 6 - 6s) = 3p - 6$$

Hence, $r(p) \le 3p - 6$.

2). Solution to the recurrence $g(n) \le \max_l(\sum_{i=1}^{l} g(n_i + 2) + 3l)$ for the third representation of separating pairs with restrictions

$$\sum_{i=1}^{l} (n_i + 1) = n \qquad 2 \le l \le n \qquad n_i \ge 0.$$

Let $g(n) = 3n - 12$,

$$g(n) \le \max_l(\sum_{i=1}^{l} g(n_i + 2) + 3l) = \max_l(\sum_{i=1}^{l} (3(n_i + 2) - 12) + 3l) =$$

$$\max_l(3\sum_{i=1}^{l} (n_i + 1) + 3l - 12l + 3l) = \max_l(3n - 6l) \le 3n - 12$$

Hence, $g(n) \le 3n - 12$.

4.2. Separating Triplets

4.2.1. O(n) Representation by Wheels and $O(n^2)$ Upper Bound

Let G be a triconnected graph on n vertices and m edges. Assume there exists a separating triplet $\{v_1,v_2,v_3\}$ in G, which separates G into nonempty G_1 and G_2 (see Figure 4.4).

Then all separating triplets of G can be divided into the following six types:

1). Separating triplets completely inside $G_1 \cup \{v_1,v_2,v_3\}$,

2). Separating triplets completely inside $G_2 \cup \{v_1,v_2,v_3\}$,

3). Separating triplets with one vertex from G_1, one vertex from G_2 and one vertex from $\{v_1,v_2,v_3\}$,

4). Separating triplets with one vertex from G_1 and two vertices from G_2,

5). Separating triplets with two vertices from G_1 and one vertex from G_2,

6). The separating triplet $\{v_1,v_2,v_3\}$.

Lemma 4.1. Only one of three vertices $\{v_1,v_2,v_3\}$ can participate in the third type separating triplets $\{w_1,v_i,w_2\}$ such that $w_1 \in G_1$, $w_2 \in G_2$ and $v_i \in \{v_1,v_2,v_3\}$.

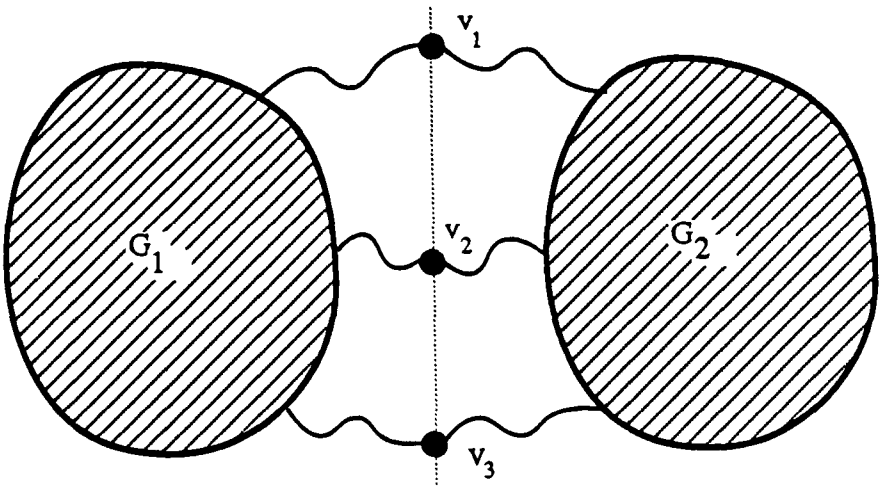


Figure 4.4.
Separating G into G_1 and G_2 by separating triplet $\{v_1,v_2,v_3\}$

Proof: Assume there is a separating triplet $\{w_1,v_2,w_2\}$ of the third type in G (see figure 4.5), where $w_1 \in G_1$ and $w_2 \in G_2$. It separates G_1 into K_1 and K_2, and separates G_2 into K_3 and K_4. Vertices v_1 and v_3 must belong to the different components of G with respect to the separating triplet $\{w_1,v_2,w_2\}$. Otherwise either $\{w_1,v_2\}$ is a separating pair, or $\{w_2,v_2\}$ is a separating pair, or both.

Claim 4.1. Vertex v_2 has an edge to every nonempty subgraph K_1,K_2,K_3,K_4.

Proof: W.L.O.G. assume that K_1 is not empty and $\forall x \in K_1$, $(x,v_2) \notin E$. Then $\{v_1,w_1\}$ is a separating pair of G, which separates K_1 from the rest of the graph.

□ Claim 4.1.

Now, we will prove that there is no separating triplet of the third type which uses v_1 or v_3. We will prove this by contradiction. W.L.O.G. assume there is a separating triplet $\{u_1,v_1,u_2\}$,

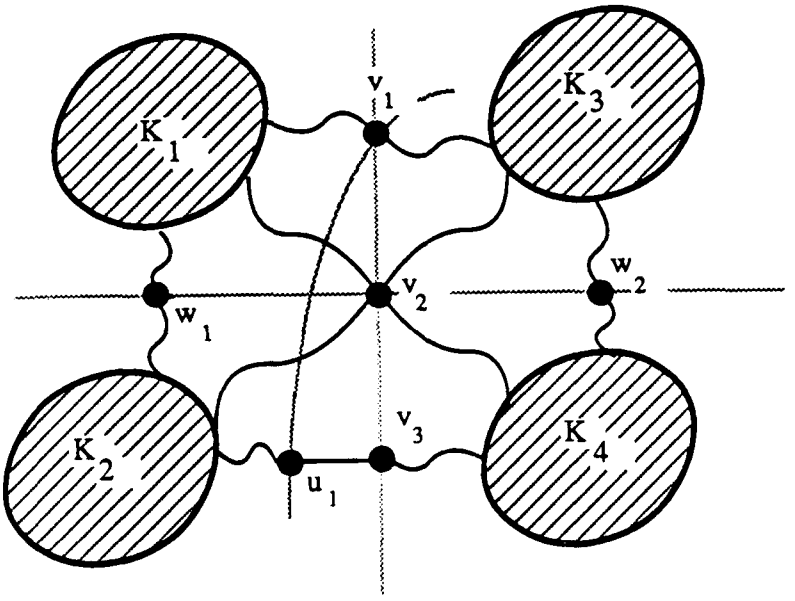


Figure 4.5
Illustrating the proof of Lemma 4.1.

such that $u_1 \in G_1$ and $u_2 \in G_2$ (u_1 may be equal to w_1 and u_2 may be equal to w_2).

Case 1: $u_1 \in K_2$, if K_2 is not empty (see Figure 4.5).

With respect to the separating triplet $\{u_1,v_1,u_2\}$, the sets K_1, $\{w_1\}$, and $K_2 - \{u_1\}$ belong to the same connected component of $G-\{u_1,v_1,u_2\}$. This is true because of the Claim 4.1 for v_1 and the existence of separating triplet $\{u_1,v_1,u_2\}$. If v_2 belongs to the same component then $\{v_1,u_1\}$ is a separating pair which separates $K_3 \cup \{w_2\} \cup K_4 \cup \{v_3\}$ from the rest of the graph. If v_2 does not belong to the same component then $\{v_1,u_1\}$ is a separating pair which separates $K_1 \cup \{w_1\} \cup K_2 - \{u_1\}$ from the rest of the graph.

Analogously, $u_2 \notin K_4$.

Case 2: $u_1 = w_1$.

Since $\{u_1,v_1,u_2\}$ is a separating triplet, K_1 is empty by Claim 4.1, otherwise $\{v_1,u_1\}$ is a separating pair which separates $K_1 \cup \{v_2\}$ from the rest of the graph. But then $\{v_1,u_2\}$ is a separating pair, if $\{u_1,v_1,u_2\}$ is a separating pair.

Analogously, $u_2 \neq w_2$.

Case 3: $u_1 \in K_1$ and $u_2 \in K_3$.

Either $\{u_1,u_2\}$, or $\{u_1,v_1\}$, or $\{v_1,u_2\}$ is a separating pair, if $\{u_1,v_1,u_2\}$ is a separating triplet.

That means that if there is a separating triplet of the third type which uses one of the $v_i, i=1,2,3$ then there is no separating triplet of the third type that uses the other two v_i's.

□ Lemma 4.1.

Let $\{v_1,v_0,v_2\}$ be a separating triplet of a graph G on n vertices, and v_0 be the only one of the three vertices of this separating triplet which might participate in a separating triplets of the third type with respect to $\{v_1,v_0,v_2\}$. Consider the separating triplet $\{v_1,v_0,v_2\}$, together with all separating triplets of the third type $\{w_1,v_0,w_2\}$ such that $w_1 \in G_1$ and $w_2 \in G_2$. All such separating triplets use v_0 as the "central" vertex. Let W_1 be the set of vertices w_1 in G_1 such

that there is a separating triplet of the third type which uses w_1. Analogously, let W_2 be the set of vertices w_2 in G_2 such that there is a separating triplet of the third type which uses w_2. Let $G'_1 = G_1 \cup \{v_1, v_3\}$, and analogously, let $G'_2 = G_2 \cup \{v_1, v_3\}$.

Claim 4.2. Every vertex in W_1 is an articulation point in G'_1, and analogously, every vertex in W_2 is an articulation point in G'_2.

Proof: We will prove Claim for W_1 only, since W_1 and W_2 are symmetric. If $w_1 \in W_1$ is not an articulation point of G'_1 then there is a path between v_1 and v_3 in $G'_1 - \{w_1\}$. Hence, either G is not triconnected or there is no separating triplet of the third type which involves w_1.

□ Claim 4.2.

$G'_1 - W_1$ consists of a collection of subgraphs $\{C_1, \cdots, C_s\}$. Let us replace each C_i by a complete graph on vertices of W_1 which are adjacent to C_i. Let us denote this structure (the set of W_1 with these complete graphs connecting them) by W'_1. Let us denote by W'_2 the structure which we get by applying this procedure to G'_2.

Claim 4.3. W'_1 is a simple path connecting v_1 and v_3. Analogously, W'_2 is a path connecting v_1 and v_3.

Proof: We will prove Claim for W'_1 only, since W'_1 and W'_2 are symmetric. There is always a path between v_1 and v_3 through W'_1 formed by the vertices of W_1. Assume there is vertex $v \in W_1$ which is not on that path. Then there is a path through $W'_1 - \{v\}$ between v_1 and v_3. So there is a path through G'_1 between v_1 and v_3. Hence, either G is not triconnected or there is no separating triplet of the third type which involves v.

□ Claim 4.3.

The combinations of these paths W'_1 and W'_2 create a cycle. Rename the vertices v_1, the vertices in W_2, v_3 and the vertices in W_1 into a sequence $<y_1, y_2, \cdots, y_l>$, such that $y_1 = v_1$, y_2

is $w_2 \in W_2$ closest to v_1 on W'_2 and so on following this cycle clockwise starting from v_1 (see Figure 4.6). Note that y_l is $w_1 \in W_1$ closest to v_1 on W'_1. Let $y_0 = v_0$.

Lemma 4.2. For all i and j with $i < j$, either the vertices y_i, y_j and y_0 form a separating triplet of G, or $j = i+1$ and there is an edge $(y_i, y_j) \in E$.

Proof: If $y_i \in G_1$ and $y_j \in G_2$ then $\{y_i, y_j, y_0\}$ is a separating triplet of G, since y_i is an articulation point in G_1, and y_j is an articulation point in G_2. If y_i and $y_j \in G_1$ then either $\{y_i, y_j, y_0\}$ is a separating triplet of G or $j = i+1$ and there is an edge $(y_i, y_j) \in E$, since y_i and y_j are articulation points in G_1. Analogously, either $\{y_i, y_j, y_0\}$ is a separating triplet of G or $j = i+1$ and there is an edge $(y_i, y_j) \in E$, if y_i and $y_j \in G_2$.

□ Lemma 4.2.

The set of y_i's together with each G_j replaced by an edge connecting the two y_i's adjacent to it forms a cycle. Any two nonadjacent vertices on the cycle of this wheel form a separating triplet together with y_0. The subgraph between y_i and y_{i+1} is denoted with G_i for each i, and some of them may be empty. Now, the graph G looks like a wheel with y_0 in a center, with y_i's and G_i's ($i = 1, \cdots, l$) on a cycle (see Figure 4.6).

Every pair of vertices on the cycle of the wheel forms a separating triplet with y_0 unless they are adjacent (y_i and y_{i+1}) and the subgraph (G_i) between them is empty. Hence, we can represent these separating triplets by the following structure (wheel):

1) $\{y_0, y_1, \cdots, y_k\}$ with edges of G between them,

2) a vertex for every G_i with a flag to specify if G_i is empty,

3) the edges between G_i and y_i, y_{i+1} $i = 1, \cdots l$. The edge between y_0 and $G_i, i = 1, \cdots, l$ with the flag to specify if the edge is in G.

Let us see where the rest of separating triplets of G lie.

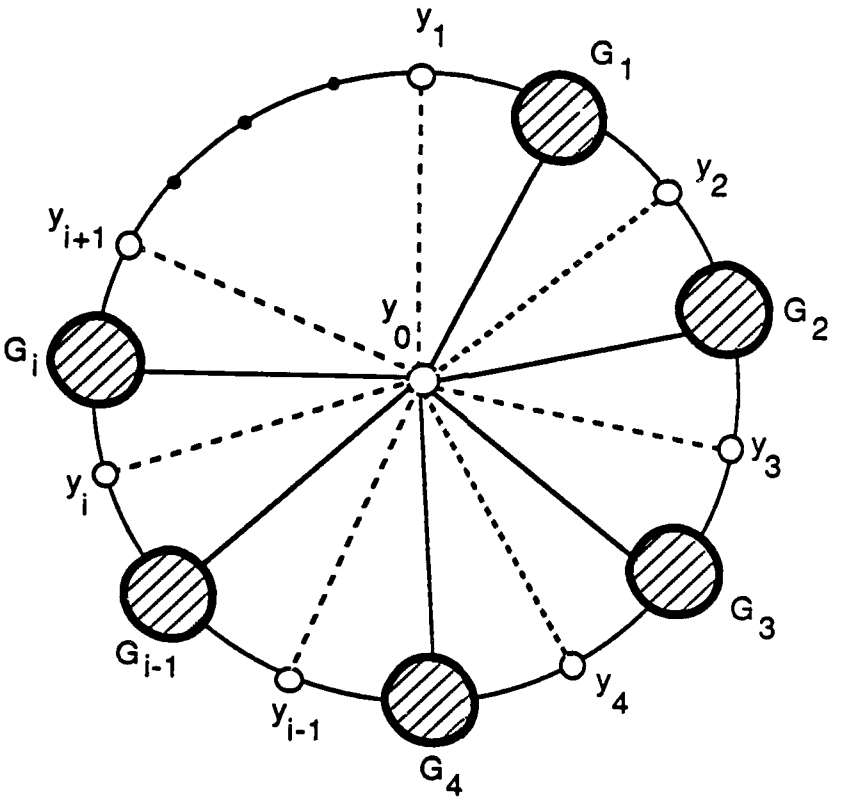


Figure 4.6.
Representation for k=3

Lemma 4.3. Each remaining separating triplet of G belong to $G_i \cup y_0 \cup y_i \cup y_{i+1} \cup$ {the neighbor of y_i in G_{i-1} if such a neighbor is unique} $\cup$ {the neighbor of y_{i+1} in G_{i+1} if such a neighbor is unique} for some i.

Proof: This is true for Type 1 and Type 2 separating triplets. All separating triplet of Type 3 are on the wheel so the Lemma is true for them. Also Type 6 separating triplet is on the wheel. Hence, we need to prove this Lemma for Types 4 and 5 only. Since these two Types are symmetric we will consider only Type 4.

Let $\{w_1,w_2,w_3\}$ be a separating triplet with $w_1 \in G_1$ and $w_2,w_3 \in G_2$. The separating triplet $\{w_1,w_2,w_3\}$ separates G_1 into L_1 and L_2, and separates G_2 into L_3 and L_4 (Figure 4.7). Let us see how the original separating triplet $\{v_1,v_2,v_3\}$ is separated by the separating triplet $\{w_1,w_2,w_3\}$.

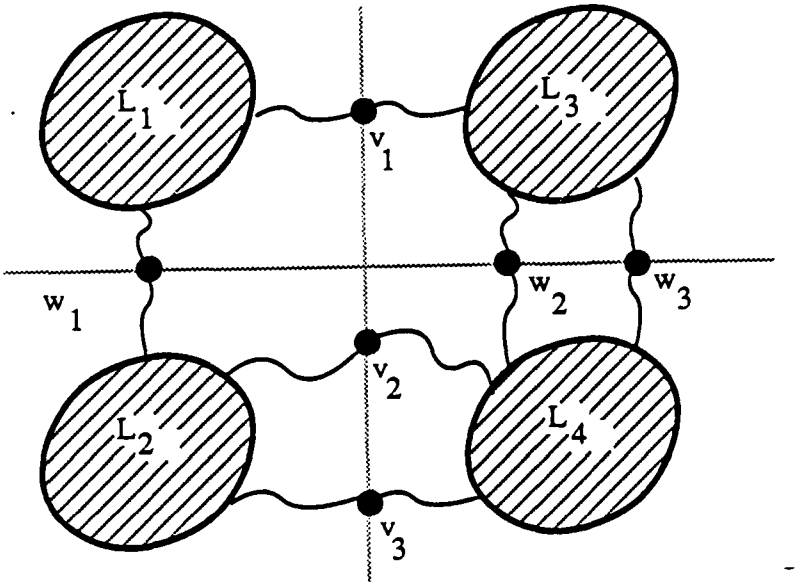


Figure 4.7.
Illustrating the proof of the Lemma 4.3.

The vertices $\{v_1,v_2,v_3\}$ cannot belong to the same connected component of G with respect to the separating triplet $\{w_1,w_2,w_3\}$. Otherwise either w_1 would be an articulation point, or $\{w_2,w_3\}$ would be a separating pair, or both. W.L.O.G. assume that v_1 belongs to one connected component and v_2,v_3 to the other with respect to the separating triplet $\{w_1,w_2,w_3\}$.

Subgraph L_1 must be empty, otherwise $\{w_1,v_1\}$ becomes a separating pair separating L_1 from the rest of the graph. Since the graph is triconnected, we have

1) $(w_1,v_1)\in E$,

2) There exist vertices $x,y\in L_3\cup\{w_2,w_3\}$ such that $(x,v_1)\in E$, and $(y,v_1)\in E$; and

3) For every $z\in L_2\cup L_4\cup\{v_2,v_3\}$ there is no edge in G of the form (z,v_1).

Hence, vertex w_1 is the unique neighbor of vertex v_1 in G_1. Moreover, if there are any separating triplets of G with one vertex in G_1 and two vertices in G_2 which separate v_1 from v_2 and v_3, then w_1 is one of the vertices of this separating triplet.

A separating triplet of G cannot have all of its three vertices in three different G_i's, for otherwise two of these vertices would form a separating pair. From the proof of the Lemma 4.2 and the fact that the set $\{y_1, y_2, \cdots, y_l\}$ is maximal, we know that if there is a separating triplet which involves a vertex from G_i, then the other two vertices belong to $\{y_i\} \cup \{y_{i+1}\} \cup \{y_0\} \cup G_i$ and the neighbor of y_i in G_{i-1}, if such a neighbor is unique, and, symmetrically, the unique neighbor of y_{i+1} in G_{i+2}, if such a neighbor exists. This proves the Lemma 4.3.

□ Lemma 4.3.

Let $g(n)$ be the size of a compact representation of the separating triplets in a graph on n vertices, and let the number of vertices in G_i be n_i. Then $\sum_{i=1}^{l}(n_i + 1) + 1 = n$, and by Lemma 4.3. we can write the following recurrence

$$g(n) \leq \max_{l}\left(\sum_{i=1}^{l} g(n_i + 5) + (5l + 1)\right),$$

where $(5l + 1)$ stands for the size of the representation of the wheel. The solution to this recurrence is clearly linear (see Appendix 4.2.3). This proves that there is a succinct $O(n)$ size representation of the separating triplets.

Analogously, the recurrence for the upper bound on the number of separating triplets becomes,

$$f(n) \leq \max_{1 \leq l \leq n-4}\left(\sum_{i=1}^{l} f(n_i + 5) + \frac{l(l-1)}{2}\right),$$

where $f(n)$ is the upper bound on the number of separating triplets of a triconnected graph on n vertices. The solution to this recurrence is clearly $O(n^2)$.

Using the above representation we can list all separating triplets in $O(M)$ time, where M is the number of separating triplets, using the procedure similar to the one for separating pairs.

4.2.2. Tight $\frac{(n-1)(n-4)}{2}$ Upper Bound

As mentioned in Chapter 3, the wheel W_n is triconnected and has $\frac{(n-1)(n-4)}{2}$ separating triplets. In the following theorem we prove that this is the worst-case for the number of separating triplets in an n-node triconnected graph.

Theorem 4.2. The number of separating triplets in an undirected triconnected graph is $\leq \frac{(n-1)(n-4)}{2}$ for any n.

Proof: Assume there exists a separating triplet $\{v_1,v_2,v_3\}$ in G, which separates G into nonempty G_1 and G_2 (see Figure 4.4).

Then all separating triplets of G can be divided into six types as in Section 4.2.1.

Let the number of vertices in G_1 be l. Then the number of vertices in G_2 is $n-l-3$. Let $g(n)$ be the maximum number of separating triplets in a graph on n vertices, $h(l,n-l)$ be the maximum number of separating triplets of the third type with respect to a separating triplet which divides the graph into subgraphs of cardinality l and $n-l-3$ and $f(l,n-l)$ and $f(n-l,l)$ be the maximum number of separating triplets of the fourth and fifth types with respect to the separating triplet which divides the graph into subgraphs of cardinality l and $n-l-3$, respectively.

Then any $g(n)$ that satisfies the recurrence

$$g(n) \leq \max_{l} (g(l+3) + g(n-l) + h(l,n-l) + f(l,n-l) + f(n-l,l) + 1)$$

is an upper bound for the number of separating triplets in G.

Let us now find the upper bounds for the functions h and f.

Lemma 4.4. $f(l,n-l) + f(n-l,l) \leq \frac{3}{2}(3n-14)$.

Proof: Let $\{w_1,w_2,w_3\}$ be a separating triplet with $w_1 \in G_1$ and $w_2,w_3 \in G_2$. The separating

triplet $\{w_1,w_2,w_3\}$ separates G_1 into L_1 and L_2, and separates G_2 into L_3 and L_4 (see Figure 4.7). As stated in the proof of the Lemma 4.3. we know that W.L.O.G. v_1 belongs to one separated component and $\{v_2,v_3\}$ to the other component with respect to separating triplet $\{w_1,w_2,w_3\}$, that L_1 is empty, and that $(w_1,v_1)\in E$, $\exists\, x,y\in L_3\cup w_2\cup w_3$: $(x,v_1)\in E$, $(y,v_1)\in E$ and $\forall\, z\in L_2\cup L_4\cup v_2\cup v_3$: $(z,v_1)\notin E$. Hence, vertex w_1 is unique up to a division of the original separating triplet $\{v_1,v_2,v_3\}$ into v_1 and $\{v_2,v_3\}$. As a consequence we have the following useful observation.

Observation 4.1. If there is a separating triplet of the fourth type which separates v_1 from v_2 and v_3 then there is no separating triplet of the fifth type which separates v_1 from v_2 and v_3.

□ Observation 4.1.

Let us see how many separating triplets of the fourth type separate the original separating triplet $\{v_1,v_2,v_3\}$ into v_1 and $\{v_2,v_3\}$. The vertex w_1 must belong to all of them. Let us see the choices for $\{w_2,w_3\}\in G_2$, such that $\{w_1,w_2,w_3\}$ is a separating triplet of the fourth type.

Assume there is a separating triplet of the fourth type $\{w_1,u_1,u_2\}$, where $u_1\in L_3$, $u_2\in L_4$. The separating triplet $\{w_1,u_1,u_2\}$ separates L_3 into L'_3 and $\tilde{L}_3$, and separates L_4 into L'_4 and $\tilde{L}_4$ (see Figure 4.8), such that $L'_3\cup L'_4$ is separated from $\tilde{L}_3\cup\tilde{L}_4$ by $\{w_1,u_1,u_2\}$.

The vertex v_1 is connected by an edge to at most one of the $L'_3\cup\{u_1\}$ and $\tilde{L}_3$, otherwise $\{w_1,u_1,u_2\}$ is not a separating triplet. If v_1 is not connected to either $L'_3\cup\{u_1$ or $\tilde{L}_3$ then $\{w_2,w_3\}$ is a separating pair. W.L.O.G. assume for every $x\in\tilde{L}_3$: $(x,v_1)\notin E$. By the symmetry $\{v_2, v_3\}$ is connected to only one of the L'_4 and $\tilde{L}_4$. Let us see how the separating triplet $\{w_1,u_1,u_2\}$ separates $\{w_2,w_3\}$.

If vertices w_2 and w_3 are not separated by $\{w_1,u_1,u_2\}$ then there are four cases to consider.

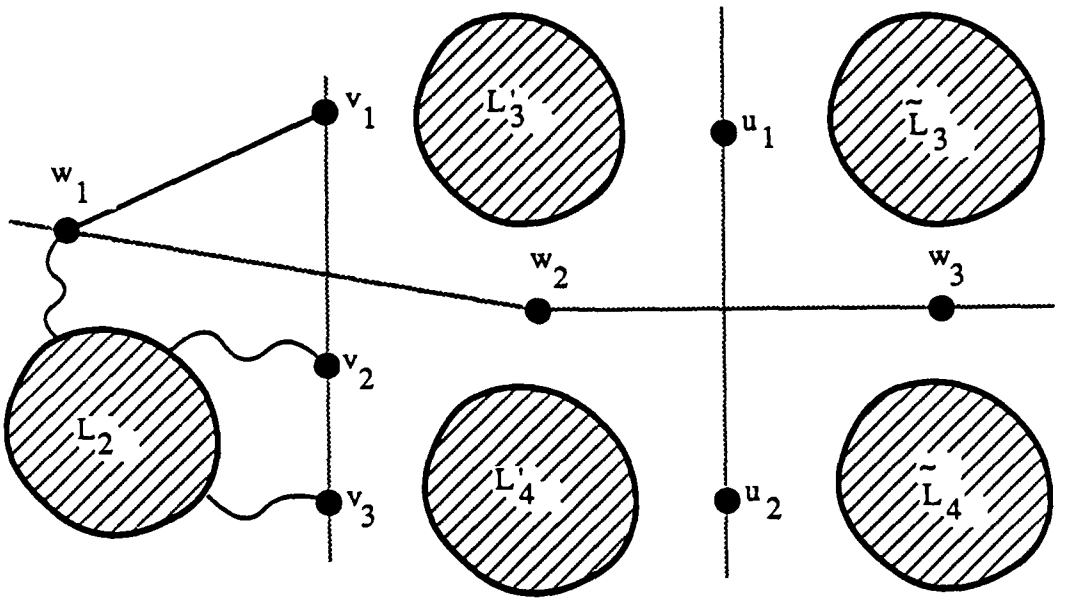


Figure 4.8.
Separating L_3 into L'_3 and $\tilde{L}_3$ and L_4 into L'_4 and $\tilde{L}_4$ by $\{w_1,u_1,u_2\}$

When w_2 and w_3 belong to the same component as L'_3 and L'_4 with respect to the separating triplet $\{w_1,u_1,u_2\}$ and $\{v_2,v_3\}$ is connected by an edge to $\tilde{L}_4$ then $\{w_1,u_2\}$ is a separating pair which separates $L_2 \cup \{v_2,v_3\} \cup \tilde{L}_4$ from $\{v_1\} \cup L_3 \cup \{w_2,w_3\} \cup L'_4$.

When w_2 and w_3 belong to the same component as L'_3 and L'_4 with respect to the separating triplet $\{w_1,u_1,u_2\}$ and $\{v_2,v_3\}$ is connected by an edge to L'_4 then $\{u_1,u_2\}$ is a separating pair which separates $\tilde{L}_3 \cup \tilde{L}_4$ from the rest of the graph.

When w_2 and w_3 belong to the same component as $\tilde{L}_3$ and $\tilde{L}_4$ with respect to the separating triplet $\{w_1,u_1,u_2\}$ and $\{v_2,v_3\}$ is connected by an edge to L'_4 then $\{u_1,u_2\}$ is a separating pair which separates $\tilde{L}_3 \cup \{w_2,w_3\} \cup \tilde{L}_4$ from the rest of the graph.

When w_2 and w_3 belong to the same component as $\tilde{L}_3$ and $\tilde{L}_4$ with respect to the separating triplet $\{w_1,u_1,u_2\}$ and $\{v_2,v_3\}$ is connected by an edge to $\tilde{L}_4$ then $\{w_1,u_1\}$ is a separating pair which separates $L'_3 \cup \{v_1\}$ from the rest of the graph.

Hence, w_2 and w_3 belong to different components with respect to the separating triplet $\{w_1,u_1,u_2\}$. Subgraph $\tilde{L}_3$ must be empty; otherwise $\{u_1,w_3\}$ becomes a separating pair. Hence, $(u_1,w_3)\in E$, otherwise $\{w_1,w_2\}$ is a separating pair. If $\{v_2,v_3\}$ is connected to L'_4 then $\{u_1,u_2\}$ is a separating pair or $\{w_1,u_1,u_2\}$ is not a separating triplet. So, for every $x\in L'_4$: $(x,v_2)\notin E$, $(x,v_3)\notin E$, $\exists\, y,z\in\tilde{L}_4\cup\{w_2,w_3\}$: $(y,v_2)\in E$, $(z,v_3)\in E$. Subgraph L'_4 must be empty, otherwise $\{w_2,u_2\}$ is a separating pair or $\{w_1,u_1,u_2\}$ is not a separating triplet. Hence, $(u_2,w_2)\in E$, otherwise $\{w_1,w_3\}$ is a separating pair (see Figure 4.9).

The above means that for each separating triplet $\{w_1,w_2,w_3\}$ there exists at most one separating triplet $\{w_1,u_1,u_2\}$ such that $u_1\in L_3$ and $u_2\in L_4$. So, for every $x\in L'_3$, and every $y\in\tilde{L}_4$ $\{w_1,x,w_3\}$, $\{w_1,x,u_2\}$, $\{w_1,y,w_2\}$, $\{w_1,y,u_1\}$ and $\{w_1,y,x\}$ are not separating triplets.

Let the number of vertices in L'_3 be l_3. Then the number of vertices in $\tilde{L}_4$ is $(n-l-3-l_3-4)=(n-l-l_3-7)$. Then the maximum number of separating triplets that use w_1 is

$$r(n-l-3)\le \max_{0\le l_3\le n-l-7}\left[r(n-l-l_3-5)-1+r(l_3+2)-1+4\right]=$$

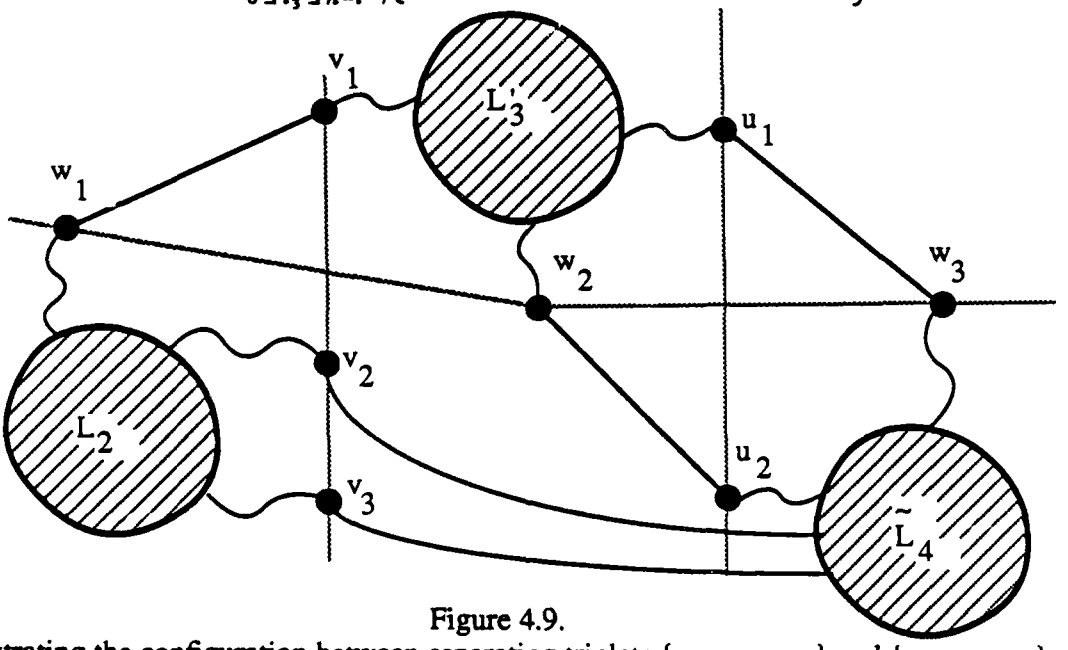


Figure 4.9.
Illustrating the configuration between separating triplets $\{w_1,w_2,w_3\}$ and $\{w_1,u_1,u_2\}$

$$\max_{0 \le l_3 \le n-l-7} \left[r(n-l-l_3-5) + r(l_3+2) \right] + 2, \quad r(2) = 1, r(1) = 0,$$

where $r(n-l-l_3-5) - 1$ counts all separating triplets which use w_1 and two vertices from $\tilde{L}_4 \cup \{u_2, w_3\}$, $r(l_3+2) - 1$ counts all separating triplets which use w_1 and two vertices from $L'_3 \cup \{u_1, w_2\}$ and 4 counts $\{w_1, u_1, u_2\}$, $\{w_1, w_2, w_3\}$, $\{w_1, u_1, w_2\}$ and $\{w_1, u_2, w_3\}$.

The solution for this recurrence is $r(n-l-3) = \frac{3}{2}(n-l-3) - 2$. We note that this analysis leads to the following observation, which we will need later.

Observation 4.2. The maximum number of separating triplets of the fourth type which separate $\{v_i\}$ from $\{v_1, v_2, v_3\}$-$\{v_i\}$ is $\le \frac{3}{2}(n-l-3) - 2$, and the maximum number of separating triplets of the fifth type which separate $\{v_i\}$ from $\{v_1, v_2, v_3\}$-$\{v_i\}$ is at most $\frac{3}{2} \cdot l - 2$.

□ Observation 4.2.

Since there exists a unique w_1 for every separation of v_i i=1,2,3 from the other two v_i's, the upper bound for the separating triplets of the fourth and fifth types together is:

$$f(l, n-l) + f(n-l, l) \le 3 \cdot \frac{3}{2} \cdot \left(\max_{1 \le l \le n-4} (n-l-3, l) - 2 \right) \le \frac{3}{2} \cdot \left[3(n-4) - 2 \right] = \frac{3}{2}(3n-14).$$

□ Lemma 4.4.

Lemma 4.5. $h(l, n-l) \le l(n-l-3)$.

Proof: From Lemma 4.1 we know that only one of $\{v_1, v_2, w_2\}$ can participate in the third type in G. Since $|G_1| = l$ and $|G_2| = n-l-3$, $h(l, n-l) \le l(n-l-3)$.

□ Lemma 4.5.

Let us now tighten the upper bound for the number of separating triplets in the triconnected graph G. Assume that $\{v_1, v_2, v_3\}$ divides G such that the ratio $\frac{|V(G_1)|}{|V(G_2)|}$ is as close to one as

possible over all separating triplets in G. From Lemma 4.5 we know that there is a unique vertex among $\{v_1,v_2,v_3\}$ that participates in the separating triplets of the third type. W.L.O.G., let this vertex be v_2.

Lemma 4.6. If there is a separating triplet of the fourth type or the fifth type that separates v_2 from v_1 and v_3 then there are no separating triplets of the third type.

Proof : W.L.O.G., assume there exists a separating triplet of the fourth type $\{w_1,w_2,w_3\}$, with $w_1 \in G_1$ and $w_2,w_3 \in G_2$, which separates v_2 from v_1 and v_3. It separates G_1 into K_1 and K_2, and separates G_2 into K_3 and K_4. From the proof of Lemma 4.3, K_1 is empty, $(w_1,v_2) \in E$ and for all $x \in G_1 \cup \{v_1,v_3\} - \{w_1\}$, $(x,v_2) \notin E$ (see Figures 4.10a and 4.10b).

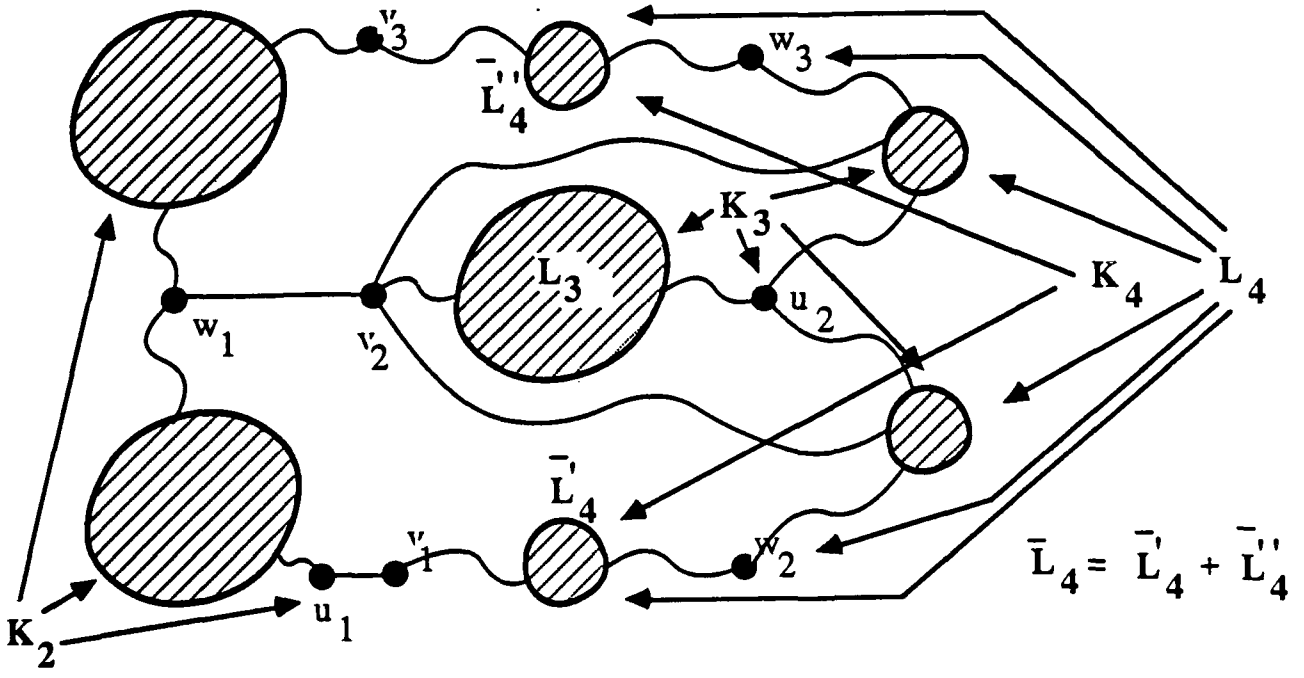


Figure 4.10a.
Illustrating the proof of Lemma 4.6.

Assume there is a separating triplet of the third type $\{u_1,v_2,u_2\}$, where $u_1 \in G_1$ and $u_2 \in G_2$. It separates G_2 into L_3 and L_4. Let $L_4 \cap K_4 = \bar{L}_4$. By Claim 4.1 v_2 must be connected by an edge to every nonempty component of G_1, G_2 which is created by separator $\{u_1,v_2,u_2\}$. But since w_1 is the only vertex in G_1 that is connected to v_2, we must have either $u_1 = w_1$, or u_1 is the unique neighbor of v_1 in G_1, or u_1 is the unique neighbor of v_3 in G_1. However, if $u_1 = w_1$ then K_2 must be empty, otherwise $\{w_1,v_3\}$ is a separating pair or $\{w_1,v_1\}$ is a separating pair or both. Hence in the case when $u_1 = w_1$, G_1 consists of the single vertex $w_1 = u_1$, which is connected to v_1, v_2, and v_3. Since the cases when u_1 is the unique neighbor of v_1 in G_1 and when u_1 is the unique neighbor of v_3 in G_1 are symmetric we can W.L.O.G. analyze only for the case when u_1 is the unique neighbor of v_1 in G_1.

If v_1, v_2 and v_3 are separated by $\{w_1,w_2,w_3\}$ from each other then we have the following analysis. In this case $\bar{L}_4$ consists of two disjoint parts: one $\overline{L'}_4$ which has all of the paths between $\overline{L'}_4$ and u_2 passing through w_2, and the other $\overline{L''}_4$ which has all of the paths between $\overline{L''}_4$ and u_2 passing through w_3 (see Figure 4.10a). One of w_2 and w_3 must be in the same component as v_1 in $G - \{u_1,v_2,u_2\}$, and the other must be in the same component as v_3. This is the case because if both w_2 and w_3 are not in the component of v_1 in $G - \{u_1,v_2,u_2\}$, then $\{w_1,u_2\}$ separates v_1 from w_2 and w_3, since $\{w_1,w_2,w_3\}$ separates v_1, v_2, and v_3 from each other and every path between v_1 and $\{w_2,w_3\}$ must pass through u_2. W.L.O.G. assume that w_2 and v_1 belong to the same connected component in $G - \{u_1,v_2,u_2\}$, and w_3 and v_3 belong to another component (see Figure 4.10a). Hence, edges (v_1,w_3) and (v_3,w_2) cannot be present in G. Moreover by the same argument, there are no edges $(l,w_3), l \in \overline{L'}_4$ and $(l,w_2), l \in \overline{L''}_4$. Also there are no edges (u_2,v_1) and (u_2,v_3) in G, since $\{w_1,w_2,w_3\}$ is a separating triplet which separates v_1, v_2, and v_3 from each other. Moreover by the same argument, there are no edges $(l,u_2), l \in \bar{L}_4$. As noted above, we assume W.L.O.G. that u_1 is the unique neighbor of v_1 in K_2. Hence $\{u_1,w_2\}$ separates $\{v_1\} \cup \overline{L'}_4$ from the rest of G_2, which contradicts the fact that G is

triconnected.

If v_1 and v_3 are not separated from each other by $\{w_1,w_2,w_3\}$ then we have the following analysis. Recall that G_2 is separated into K_3 and K_4 by $\{w_1,w_2,w_3\}$ (see Figure 4.10b). Since u_1 is the unique neighbor of v_1 in K_2 and $\{u_1,v_2,u_2\}$ is a separating triplet, u_2 must break all paths either between v_2 and $\{v_1,v_3\}$ through G_2, or between v_1 and $\{v_2,v_3\}$ through G_2. If $u_2 \in K_3$ then $\{w_1,u_2\}$ separates v_2 from v_1, since $\{u_1,v_2,u_2\}$ is a separating triplet. If $u_2 \in K_4$ then in order for $\{u_1,v_2,u_2\}$ to be a separating triplet, either $\{u_1,u_2\}$ separates v_1 from v_3, or $\{v_2,u_2\}$ separates w_2 and w_3 from v_3. If $u_2 \in \{w_2,w_3\}$ then $\{u_1,v_2,u_2\}$ is not a separating triplet, since there is a path from w_1 to K_3 through $K_2 - \{u_1\}$, v_3, $K_4 \cup \{v_1$, and $\{w_2,w_3\}-\{u_2\}$ in $G - \{u_1,v_2,u_2\}$.

These two contradictions prove the lemma.

□ Lemma 4.6.

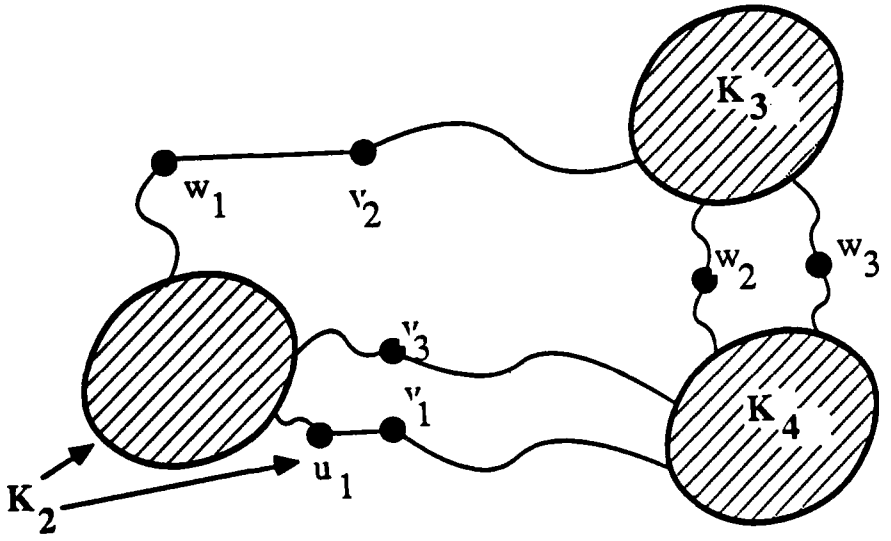


Figure 4.10b.
Illustrating the proof of Lemma 4.6.

Now we will do a case by case analysis of trade-offs between separating triplets of the third type and the separating triplets of the fourth type and the fifth type.

If there exists a separating triplet of the fourth type that separates v_1 from v_2 and v_3, then no separating triplet of the fifth type exists which separates v_1 from v_2 and _ (Observation 4.1). Since the separating triplets of the fourth type and the fifth type are analogous, we need to consider only one of them in the case analysis.

If there is a separating triplet of fourth type $\{w_1,w_2,w_3\}$ with $w_1 \in G_1$ and $w_2,w_3 \in G_2$, that separates v_1 from v_2 and v_3, then we have the following analysis. G_2 is separated by $\{w_1,w_2,w_3\}$ into G'_2 and $\tilde{G}_2$ and $G_1 = \{w_1\} \cup \tilde{G}_1$ (see Figure 4.11). Choose $\{w_1,w_2,w_3\}$ to maximize $|V(G'_2)|$. Let $|V(G'_2) \cup \{w_2,w_3\}| = l'_2$. Now we will consider two cases

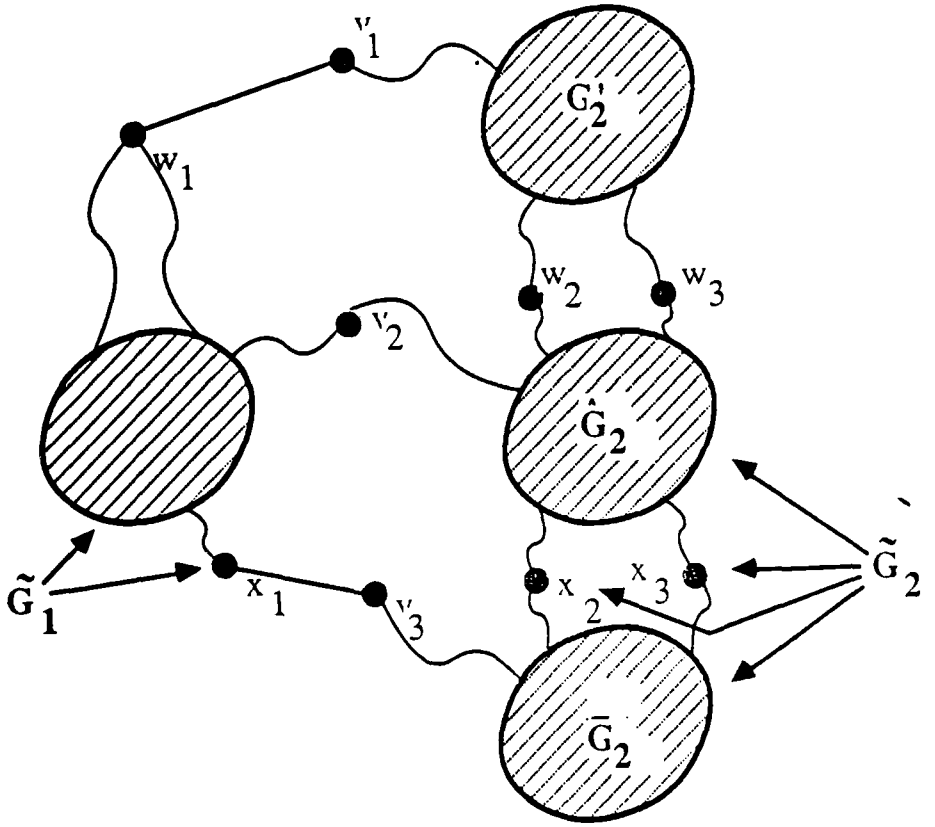


Figure 4.11.
Illustrating Case A in trade-off analysis.

depending on whether or not a separating triplet of the fifth type exists, which separate v_3 from $\{v_1,v_2\}$. We do not restrict separating triplets of the third type (recall that such a triplet always includes v_2), or separating triplets of the fourth or fifth types which separate v_2 from $\{v_1,v_3\}$.

Case A: There is no separating triplet of the fifth type that separates $\{v_3\}$ from $\{v_1,v_2\}$ with $x_1 \in G_1$ and $x_2,x_3 \in G_2$. If there is a separating triplet of the fourth type $\{x_1,x_2,x_3\}$ that separates $\{v_3\}$ from $\{v_1,v_2\}$ with $x_1 \in G_1$ and $x_2,x_3 \in G_2$ (see Figure 4.11), then it separates G_2 into $\bar{G}_2$ and the rest of G_2. Since it separates v_3 from $\{v_1,v_2\}$, we must have for every $x \in \bar{G}_2$:$(x,v_2) \notin E$. Vertices x_2 and $x_3 \in \tilde{G}_2 \cup \{w_2,w_3\}$ (recall that $\tilde{G}_2 = G_2 - G'_2 - \{w_2,w_3\}$), otherwise there is no vertex of G_2 which has an edge to v_2.

Let $\hat{G}_2 = \tilde{G}_2 - \bar{G}_2$. Furthermore, suppose $\{x_1,x_2,x_3\}$ maximizes $|V(\bar{G}_2)|$. Let $|V(\bar{G}_2) \cup \{x_2,x_3\}| = \bar{l}_2$. Note that if there is no separating triplet of the fourth type which separates v_3 from $\{v_1,v_2\}$ then $\bar{l}_2 = 0$.

Assume there is a separating triplet $\{u_1,v_2,u_2\}$, of the third type where $u_1 \in G_1$ and $u_2 \in G_2$. Since $\{w_1,w_2,w_3\}$ separates v_1 from v_2 and v_3, and since $|G'_2|$ is maximum among all separating triplets of the fourth type which separate v_1 from v_2 and v_3, we must have for every $x \in G'_2$:$(x,v_2) \notin E$. From Claim 4.1 we know that there must be edges between v_2 and every nonempty component of G with respect to the two separating triplets $\{v_1,v_2,v_3\}$ and $\{u_1,v_2,u_2\}$. If $u_2 \in G'_2 \cup \{w_2,w_3\}$ then $\{w_1,u_2\}$ would be a separating pair. Hence, $u_2 \in \tilde{G}_2$. Analogously, using the above analysis for the separating triplet $\{x_1,x_2,x_3\}$, we get that $u_2 \in \hat{G}_2$. Hence, using Lemma 4.5 we get that the number of separating triplets of the third type is at most $|V(G_1)| \cdot |V(\tilde{G}_2)| = l(n-l-l'_2-\bar{l}_2-3)$.

By the proof of Lemma 4.4 and the fact that $|G'_2|$ is maximum we know that for all separating triplets $\{y_1,y_2,y_3\}$ of the fourth type which separate v_1 from v_2 and v_3, vertices y_2 and y_3 are inside $G'_2 \cup \{w_2,w_3\}$. By the proof of Lemma 4.4 and the fact that $|\bar{G}_2|$ is maximum we know that for all separating triplets $\{x_1,x_2,x_3\}$ of the fourth type which separate v_3

from v_2 and v_1, x_2 and x_3 are inside $\overline{G}_2 \cup \{w_2, w_3\}$. Hence, the number of separating triplets of fourth type which separate vertex v_3 from v_2 and from v_1 or v_1 from v_2 and v_3 is at most $|V(G'_2)| + |V(\hat{G}_2)| = \frac{3}{2}(l'_2+\overline{l}_2)$ (Observation 4.2).

We can have separating triplets of the fourth or fifth types which separate v_2 from $\{v_1, v_3\}$ if we do not have any separating triplets of the third type (Lemma 4.6). We cannot have separating triplets of both the fourth and fifth types which separate v_2 from $\{v_1, v_3\}$ by Observation 4.1. If we have separating triplets of the fourth type which separate v_2 from $\{v_1, v_3\}$ then by the analysis above the maximum number of separating triplets of the fourth type which separate one of three vertices $\{v_1, v_2, v_3\}$ from the other two is upper bounded by $\frac{3}{2}(n-l-3)$. If we have separating triplets of the fifth type which separate v_2 from $\{v_1, v_3\}$ then by the analysis of Case B below (i.e. there is a separating triplet of the fifth type $\{x_1, x_2, x_3\}$ which separates $\{v_3\}$ from $\{v_2, v_1\}$ with $x_1 \in G_2$ and $x_2, x_3 \in G_1$) for separating triplets of the fifth type the maximum number of separating triplets of the fourth or fifth types which separate one of three vertices $\{v_1, v_2, v_3\}$ from the remaining two is at most $\frac{3}{2}(l'_2+\overline{l}_2 + l-1)$. Note that $\frac{3}{2}(l'_2+\overline{l}_2 + l-1) \leq \frac{3}{2}(n-4)$, because $0 \leq \overline{l}_2 \leq n-l-l'_2-4$, $0 \leq l'_2 \leq n-l-4$ and $1 \leq l \leq n-4$. Hence, we obtain that the upper bound on the number of cross separating triplets of Case A is

$$f_a = \max\Big(\max_{\substack{0 \leq l'_2 \leq n-l-4 \\ 0 \leq \overline{l}_2 \leq n-l-l'_2-4}} \Big(l(n-l-l'_2-\overline{l}_2-3) + \frac{3}{2}(l'_2+\overline{l}_2) \Big) , \frac{3}{2}(n-4) \Big) .$$

Case B: There is a separating triplet of the fifth type $\{x_1, x_2, x_3\}$ which separates $\{v_3\}$ from $\{v_2, v_1\}$ with $x_1 \in G_2$ and $x_2, x_3 \in G_1$ (see Figure 4.12). It separates G_1 into $G'_1 \cup \{w_1\}$ and $\tilde{G}_1$. Choose $\{x_1, x_2, x_3\}$ to maximize $|V(\tilde{G}_1)|$. Let $\tilde{l}_1 = |V(\tilde{G}_1) \cup \{x_2\} \cup \{x_3\}|$.

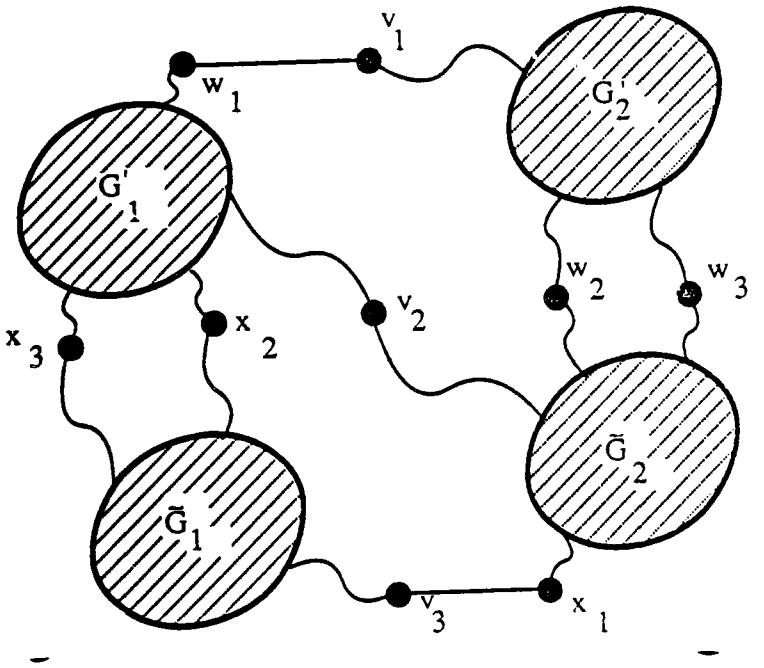


Figure 4.12.
Illustrating Case B in trade-off analysis.

If there is a separating triplet of the third type $\{u_1,v_2,u_2\}$, where $u_1 \in G_1$ and $u_2 \in G_2$, then by analysis of Case A $u_2 \in \tilde{G}_2 \cup \{x_1\}$. Analogously, by the same analysis for the separating triplet $\{x_1,x_2,x_3\}$, $u_1 \in G'_1 \cup \{w_1$. Hence, using Lemma 4.5 the number of separating triplets of the third type is at most $(n-l-l'_2-3)(l-\tilde{l}_1)$.

The number of separating triplets of the fourth and the fifth types except the ones which separate v_2 from $\{v_1,v_3\}$ is at most $\frac{3}{2}(l'_2+\tilde{l}_1)$ by the analysis of Case A and the above analysis. We can have separating triplets of the fourth or fifth types which separate v_2 from $\{v_1,v_3\}$ if we don't have any separating triplets of the third type (Lemma 4.6). We cannot have separating triplets of both the fourth and fifth types which separate v_2 from $\{v_1,v_3\}$ by Observation 4.1. If

we have separating triplets of the fourth type then number of separating triplets of the fourth type is at most $\frac{3}{2}(n-l-3)$ by the analysis of Case A. If we have separating triplets of the fifth type then number of separating triplets of the fifth types is at most $\frac{3}{2}\cdot l$ by the above analysis. Hence in this case the upper bound on the number of cross separating triplets is

$$f_b = \max\Big(\max_{\substack{0\le l'_2\le n-l-4\\ 0\le \tilde{l}_1\le l-1}}((n-l-l'_2-3)(l-\tilde{l}_1)+\frac{3}{2}(l'_2+\tilde{l}_1))\,,\ \frac{3}{2}(n-4)\Big)$$

Combining the results from Cases A and B we obtain the following recurrence for $g(n)$:

$$g(n) \le \max_{1\le l\le n-4}(g(l+3)+g(n-l)+\max(f_a,f_b)+1)$$

Note that the first term in each of f_a and f_b is bilinear in l'_2, $\bar{l}_2$ and $\tilde{l}_1$, hence the maximum is reached at the endpoints of the intervals for l'_2, $\bar{l}_2$, and $\tilde{l}_1$. The recurrences we get at the end points of the intervals for $\bar{l}_2$ and $\tilde{l}_1$ are identical up to the symmetry with respect to l and $n-l-3$. Hence, we will analyze the recurrence for l'_2, $\bar{l}_2$ only.

The maximum is reached at $l'_2 = \bar{l}_2 = 0$ when $n-4>l>1$ and the recurrence become

$$g(n) \le \max_{1\le l\le n-4}(g(l+3)+g(n-l)+l(n-l-3)+1)$$

The largest function satisfying this recurrence is $g(n)=\frac{1}{2}n^2-\frac{5}{2}n+2$. Note that, with this solution, equality holds since this recurrence is the recurrence for the wheel and the wheel W_n has this number of separating triplets.

If $l=1$ or if $l=n-4$ then maximum is reached only when $\tilde{l}_1 = l' = 0$ and $\bar{l}_2 = n-4$ or $\bar{l}_2 = \tilde{l}_1 = 0$ and $l'_2 = n-4$. But then either $\{v_1,w_2,w_3\}$ or $\{v_3,x_2,x_3\}$, or the separating triplet of the fourth or fifth type which separate v_2 from $\{v_1,v_3\}$ would be chosen instead of $\{v_1,v_2,v_3\}$ to get the ratio $\frac{|V(G_1)|}{|V(G_2)|}$ closer to one. Hence, l cannot be equal to 1 or $n-4$.

Hence the maximum number of separating triplets for an undirected triconnected graph on n vertices is exactly $\frac{1}{2}n^2 - \frac{5}{2}n + 2$.

□ Theorem 4.2.

4.2.3. Appendix

Solution to the recurrence $g(n) \le \max_l(\sum_i^l g(n_i + 5) + 5l + 1)$ for the representation of separating triplets with constrains

$$\sum_{i=1}^{l}(n_i + 1) + 1 = n \qquad 2 \le l \le n-1 \qquad n_i \ge 0$$

Let $g(n) = 5n - 46$,

$$g(n) \le \max_l(\sum_{i=1}^{l} g(n_i + 5) + 5l + 1) = \max_l(\sum_{i=1}^{l}(5n_i - 21) + 5l + 1) =$$

$$\max_l(5(\sum_{i=1}^{l}(n_i + 1) + 1) - 26l + 5l - 4) = \max_l(5n - 21l - 4) \le 5n - 46$$

CHAPTER 5

UPPER BOUND AND REPRESENTATION FOR THE SEPARATING K-SETS: GENERAL K

5.1. $O(2^k \frac{n^2}{k})$ Upper Bound and $O(k^2 n)$ Representation for General k

Let $G=(V,E)$ be an undirected k-connected graph with n vertices and m edges. We denote with $g(n)$ and $f(n)$ the upper bounds on the size of representation and the number of a separating k-sets for k-connected graph on n vertices. Let $V' = \{v_1, v_2, \cdots, v_k\}$ be a separating k-set, whose removal separates G into nonempty G_1 and G_2 (see Figure 5.1). A separating k-set $\{w_1, w_2, \cdots, w_k\}$ of G is a *cross* separating k-set with respect to V' if for some i and j, $w_i \in G_1$ and $w_j \in G_2$. Let the cardinalities of G_1 and G_2 be l and $n-l-k$, respectively. Let the upper bound on the size of the representation of the cross separating k-sets be $h(l,n-l)$, and the maximum number of cross separating k-sets be $r(l,n-l)$. Then maximal $g(n)$ and $f(n)$ that satisfy the recurrences

$$g(n) \le \max_l \left\{ g(l+k) + g(n-l) + h(l,n-l) \right\} ,$$

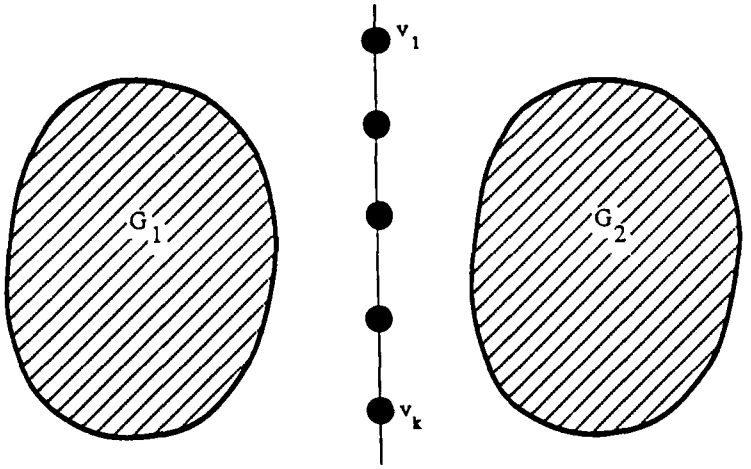


Figure 5.1.
Dividing G into G_1 and G_2 by separating k-set $\{v_1, \cdots, v_k\}$

$$f(n) \le \max_{l} \left[f(l+k) + f(n-l) + r(l,n-l) + 1 \right] ,$$

are upper bounds on the size of representation and the number of separating k-sets in G. Now we will derive upper bounds for the functions h and r and solve the recurrences.

Let $\{w_1,w_2,\cdots,w_k\}$ be a cross separating k-set with $\{w_1,\cdots,w_s\} \subset G_1$, $\{w_{s+t+1},\cdots,w_k\} \subset G_2$ and $\{w_{s+1},\cdots,w_{s+t}\} \subset \{v_1,\cdots,v_k\}$. The separating k-set $\{w_1,w_2,\cdots,w_k\}$ separates G_1 into G_3 and G_4, separates G_2 into G_5 and G_6, and divides $\{v_1,\cdots,v_k\}$ into $\{v_1,\cdots,v_r\}$, $\{v_{r+t+1},\cdots,v_k\}$ and $v_{r+i} = w_{s+i}$, $i = 1,...,t$. (see Figure 5.2)

Case 1 None of G_i, $i = 3,4,5,6$ are empty. (see Figure 5.2)

The sets $\{w_1,w_2,\cdots,w_{s+t},v_1,\cdots,v_r\}$, $\{w_1,w_2,\cdots,w_{s+t},v_{r+t+1},\cdots,v_k\}$, $\{v_1,\cdots,v_{r+t},w_{s+t+1},\cdots,w_k\}$ and $\{v_{r+1},\cdots,v_k,w_{s+t+1},\cdots,w_k\}$ are separating sets of G that

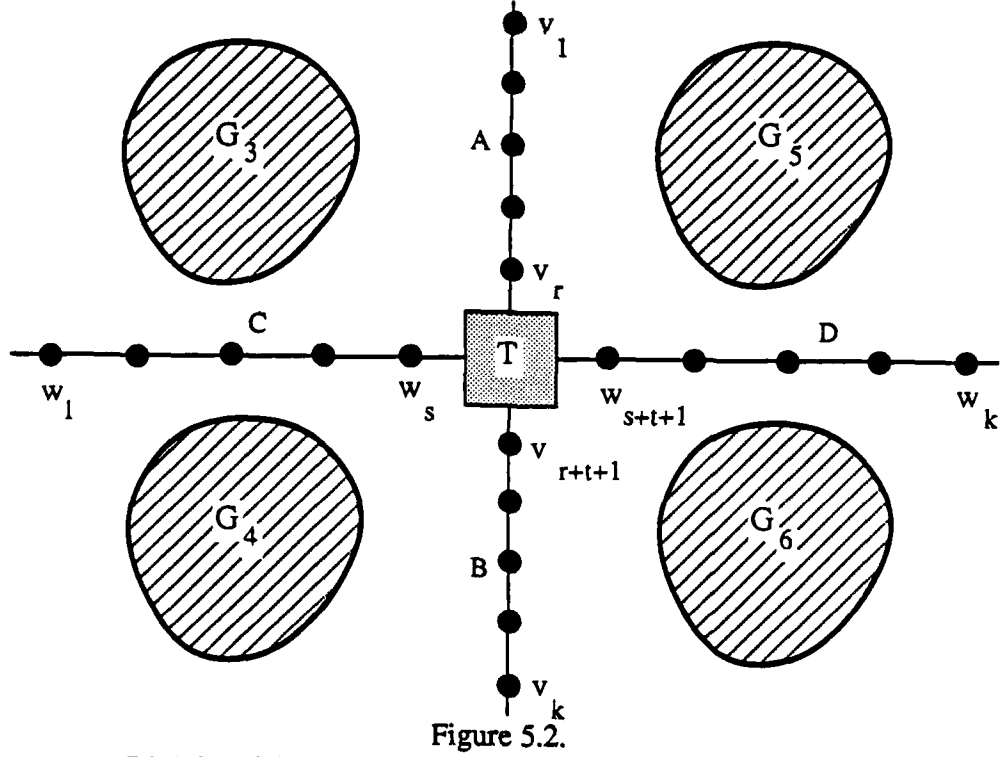


Figure 5.2.
Dividing G into nonempty components by separating k-sets $\{v_1,\cdots,v_k\}$ and $\{w_1,\cdots,w_k\}$.

separate G_3, G_4, G_5 and G_6 respectively, so their cardinalities are greater than or equal to k. Then,

$$\begin{cases} s+t+r \geq k \\ r+t+k-s-t \geq k \\ s+t+k-r-t \geq k \\ k-r+k-s-t \geq k \end{cases} \Rightarrow \begin{cases} r+s+t \geq k \\ r \geq s \\ s \geq r \\ k \geq r+s+t \end{cases} \Rightarrow \begin{cases} r=s \\ r+s+t=k \end{cases}$$

From now on we replace the subscript r by s. Let $A = \{v_1, \cdots, v_s\}$, $B = \{v_{s+t+1}, \cdots, v_k\}$, $C = \{w_1, \cdots, w_s\}$, $D = \{w_{s+t+1}, \cdots, w_k\}$, and $T = \{v_{s+1}, \cdots, v_{s+t}\} = \{w_{s+1}, \cdots w_{s+t}\}$. For Case 1 $|A| = |B| = |C| = |D| = \frac{k-t}{2}$.

Claim 5.1. For all i, $i = s+1,...,t$ and for each $j = 3,4,5,6$, there is $x_j \in G_j$, $j = 3,4,5,6$ such that $(v_i, x_j) \in E$.

Proof: W.L.O.G. assume there is v_i such that for all $x \in G_3$: $(x, v_i) \notin E$. Then $\{v_1, \cdots, v_{s+t}, w_1, \cdots, w_s\} - \{v_i\}$ is a separating (k-1)-set.

□ Claim 5.1.

Claim 5.2. For every $x \in A$ there are $y \in G_3$ and $z \in G_5$, such that $(x,y) \in E$ and $(x,z) \in E$. Analogously, for every vertex x of B, C and D there are vertices v_i in those G_i, i=3,4,5,6, which are adjacent to x, such that $(x, v_i) \in E$.

Proof: W.L.O.G. assume there is $x \in A$ such that for every $y \in G_3$ $(x,y) \notin E$. Then $A \cup C \cup T$-{x} is a separating (k-1)-set.

□ Claim 5.2.

Lemma 5.1. All cross separating k-sets containing $C \cup T$ and at least one fixed vertex of D can be represented in $O((\frac{k-t}{2})^2)$ space, and their number is $O(2^{\frac{k-t}{2}})$.

Proof: Assume we have a separating k-set $R = \{w_1, \cdots, w_{s+t+a}, x_{s+t+a+1}, \cdots, x_{s+t+a+b}, y_{s+t+a+b+1}, \cdots, y_k\}$, where

$\{x_{s+t+a+1}, \cdots, x_{s+t+a+b}\} \in G_5$, $\{y_{s+t+a+b+1}, \cdots, y_k\} \in G_6$, $a \geq 1$, and either b or $k-s-t-a-b$ is greater or equal to 1 (the new cross separating k-set is different from the old one) (see Figure 5.3).

Let $H = \{x_{s+t+a+1}, \cdots, x_{s+t+a+b}\}$ and $I = \{y_{s+t+a+b+1}, \cdots, y_k\}$, and let D be divided into D', E, and F, where: $D' = \{w_{s+t+1}, \cdots, w_{s+t+a}\}$; E is in the same connected component of G with respect to the separating k-set R as G_3, A, and part of G_5; and F is in the same connected component of G with respect to the separating k-set R as G_4, B and part of G_6. Also let H divide G_5 into G'_5 and G''_5, and let I divide G_6 into G'_6 and G''_6 (see Figure 5.3).

Separating sets $T \cup D' \cup E \cup H$ and $T \cup D' \cup F \cup I$ separate G''_5 and G''_6, respectively. The cardinalities of these separating sets are less than k. Hence, G''_5 and G''_6 are empty. Moreover, since $C \cup T \cup D' \cup H \cup F$ and $C \cup T \cup D' \cup E \cup I$ are separating sets and $C \cup T \cup D$ and $C \cup T \cup D' \cup H \cup I$ are separating k-sets, $|E| = |H|$, and $|I| = |F|$. Note that the argument still

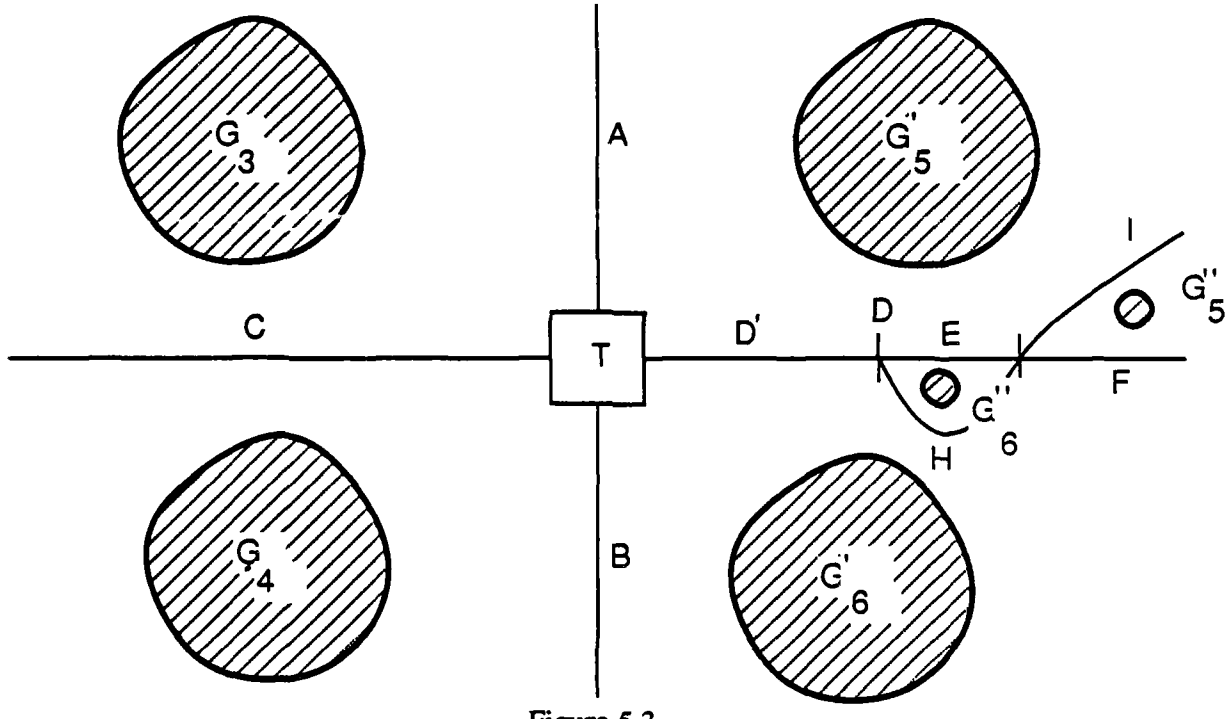


Figure 5.3.
Illustrating the proof of Lemma 5.1.

holds if either H or I is empty.

Next, we will show that if we replace part of E and/or part of F we will necessarily use only vertices of H and/or I for it, regardless of whether we replace part of D' or not. In other words, H and I are unique for E and F. The proof is by contradiction.

Assume that there exists $I_1 \cup H_1 \neq I \cup H$, such that $C \cup T \cup D' \cup H_1 \cup I_1$ is a separating k-set. Let $H_1 \subseteq G_5$ and $I_1 \subseteq G_6$. Also, let $I_1 + H_1$ divide E into E_1 and E_2, and divide F into F_1 and F_2 (see Figure 5.4).

Let H_1 be separated into two parts, H'_1 adjacent to E and E''_1 adjacent to F. By the above arguments H'_1 is adjacent to E_1, H''_1 is adjacent to F_2, and I_1 is adjacent to $E_2 \cup F_1$. Since all neighbors of E in G_6 are also in I, and all neighbors of F in G_5 are also in H, $H''_1 \subset H$ and I_1 is divided into $I'_1 = I \cup I_1$ and $I''_1 = I_1 - I'_1$. Let $H' = H - H''_1$ and let $I' = I - I'_1$.

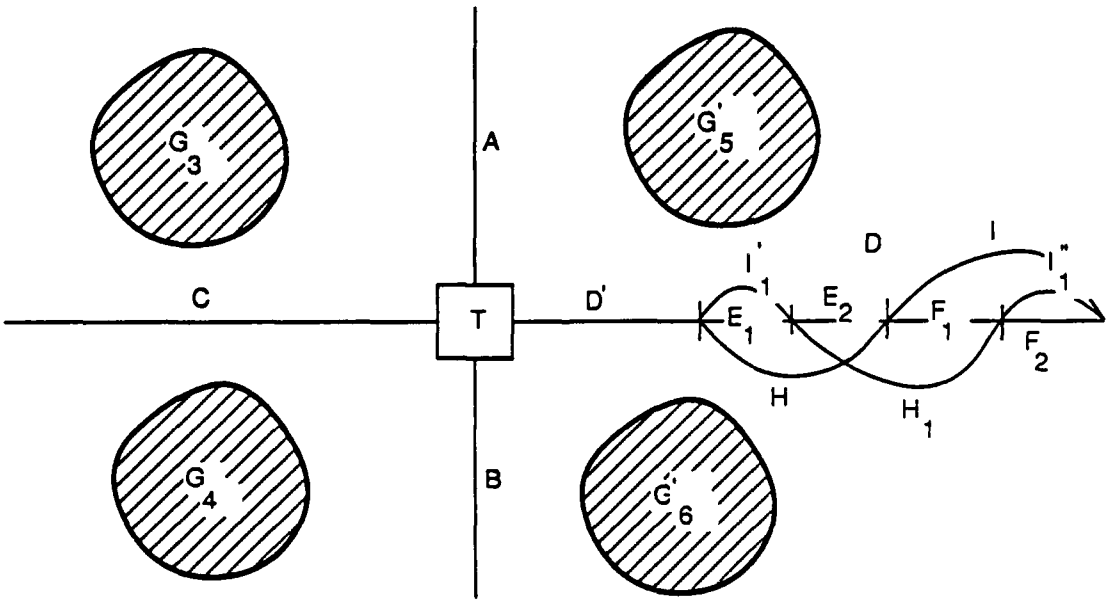


Figure 5.4.
Illustrating the uniqueness of a replacement for a part of cross separating k-set.

The separating set $T \cup D' \cup H'_1 \cup H$ separates E_1 from the rest of the graph and has cardinality less than k. Hence, E_1 is empty and we have $I = I'_1$, $E = E_2$ and $H_1 = H''_1$. Analogously, the separating set $T \cup D' \cup I_1 \cup H$ separates F_1 from the rest of the graph and has cardinality less than k. Hence, F_1 is empty and we have $F = F_2$, $E = E_1$, $H = H_1$ and $I = I_1$. This contradicts the assumptions.

Note that the arguments still hold if either H or I is empty, or if we replace only parts of E and F. If part of D' is replaced as well, then we will not replace it, so that we will look only at the replacements for E and F. Also, if there exists a separating k-set that replaces F by H, then there is no $I_1 \subseteq G_6$ that replaces any part of F for any cross separating k-set described in Lemma 5.1.

Thus, any replacement of any part of F for any cross separating k-set specified by Lemma 5.1 lies in H. The *fringe* of D is the set of vertices which is used for all possible replacement of any part of D for a cross separating k-set specified by Lemma 5.1. H is the fringe of F and I is the fringe of E. Note that there could be parts of D which do not have any replacements. The cardinality of the fringe of D is less than $\frac{k-t}{2} = |D|$. Hence, the representation of all cross separating k-sets with $C \cup T$ fixed along with at least one vertex from D takes $O((\frac{k-t}{2})^2)$ space, where $O((\frac{k-t}{2})^2)$ space is needed to specify all edges between D and its fringe. This proves the space complexity for the representation.

The number of different subsets of D is $2^{|D|}$. Since for every subset $E \cup F$ of D there is a unique replacement (if it exists), the number of separating k-sets of G with $C \cup T$ and at least one vertex from D fixed is upper bounded by $O(2^{\frac{k-t}{2}})$. This proves the second part of the Lemma.

□ Lemma 5.1.

Corollary 5.1. All cross separating k-sets containing $T \cup D$ and at least one vertex from C can

be represented in $O((\frac{k-t}{2})^2)$ space, and their number is $O(2^{\frac{k-t}{2}})$.

Take a maximal set $X=\{C_1, C_2, \cdots, C_a\}$ of disjoint $C_i \in G_1$ such that $C_i \cup T \cup D$ is a separating k-set and that for each C_i there are no vertex of $C_j, j \neq i$ in the fringe of C_i. Analogously, take a maximal set $Y=\{D_1, D_2, \cdots, D_b\}$ of disjoint $D_i \in G_2$ such that $C \cup T \cup D_i$ is a separating k-set and that for each D_i there are no vertex of $D_j, j \neq i$ in the fringe of D_i. For T fixed, all cross separating k-sets are upper bounded by $O(2^{\frac{k-t}{2}} |X| 2^{\frac{k-t}{2}} |Y|) = O(2^{k-t} |X| |Y|)$, and are represented in $O((\frac{k-t}{2})^2 (|X| + |Y|))$ space. Next we will see how many different T's we need to consider.

Take the smallest $T = T_1$ such that a cross separating k-set will have nonempty G_i i=3,4,5,6, if it exists. If there is a separating k-set with different $T = T_2, T_1 \neq T_2$, then it can be of four different types:

Type 1). $T_2 \cap A \neq \emptyset$ and $T_2 \cap B \neq \emptyset$,

Type 2). $\left[T_2 \cap A = \emptyset \text{ or } T_2 \cap B = \emptyset\right]$ and $T_1 \cap T_2 \neq \emptyset$,

Type 3). $\left[T_2 \cap A = \emptyset \text{ or } T_2 \cap B = \emptyset\right]$ and $T_1 \cap T_2 = \emptyset$,

Type 4). $T_2 \cap A = \emptyset$ and $T_2 \cap B = \emptyset$.

Let us first consider type 4 cross separating k-sets. Since Γ_2 must lie completely inside T_1 and T_1 has the smallest cardinality, then $T_2 = T_1$. Let the cardinality of X, the maximal disjoint set of C's, be l_1, and let the cardinality of Y, the maximal disjoint set of D's be l_2, where $l_1 + l_2 = l$. Recall that the set X is the maximal disjoint set of separating $\frac{k-t}{2}$-sets in G_1, and the set Y is the maximal disjoint set of separating $\frac{k-t}{2}$-sets in G_2. Let C_1 be the separating $\frac{k-t}{2}$-sets in G_1 such that all other C's $\in X$ are on one side of it (C_1 separate G_1 into G' and G''. $\bigcup C_i \in G'$ or $\bigcup C_i \in G''$.) and all paths from them to B in $G_1 \cup B$ must pass through it. Let C_i be

the separating $\frac{k-t}{2}$-sets in G_1 such that all C_j $j<i$ are on one side of it (analogous to the one above). Note that by Lemma 5.1 such an ordering of C_i's must exist. By Lemma 5.1 and the fact that the C_i's themselves are disjoint, note that such an ordering of C_i's must exist. Analogously, let D_1 be the separating $\frac{k-t}{2}$-sets in G_2 such that all other D's $\in Y$ are on one side of it and all paths from them to A in $G_2 \cup A$ must pass through it. And let D_i be the separating $\frac{k-t}{2}$-sets in G_2 such that all D_j $j<i$ are on one side of it. Let us relabel A, the set of elements of X, B and the set of elements of Y. So A becomes A_1, the D_1 becomes A_2, ... , D_b becomes A_{b+1}, B becomes A_{b+2}, C_1 becomes A_{b+3}, ... , C_a becomes A_{b+a+2} (see Figure 5.5). The cardinality of this set is $l+2$. From the proof of Lemma 5.1 we know that all cross separating k-sets of type 4 consist of three parts: T_1, C which is inside G_1 and is inside some C's from set X and its fringe, and D which is inside G_2 and is inside some D's from set Y and its fringe. Note that $T \cup$ any two $A_i, i=1, \cdots, l+2$ are also separating k-sets if the parts of the graph between them are nonempty. We can also replace parts of A_i by its fringe as long the above condition will be true. Let the part of the graph G between A_i and $A_{i+1}, i=1, \cdots, l+2$ be $G_i, i=1, \cdots, l+2$ (i in this case taken mod $l+2$). Let $G_i -$ the fringe of A_i in $G_i -$ the fringe of A_{i+1} in G_i be $G'_i, i=1, \cdots, l+2$. The only case when $T \cup A_i \cup A_j$ (or parts of the fringe of A_i and A_{i+1}) $i<j$ is not a separating k-set when $i=j-1$ and $G'_i = \emptyset$.

Based upon above observations the structure (structure 1) which covers all cross separating k-sets of type 4 will be the following (see Figure 5.5):

1) A_i with its fringes for all $i=1, \cdots, l+2$,

2) For every nonempty $G'_i, i=1, \cdots, l+2$ we fill all nonexistent edges of the complete graph on the neighbors of G'_i as real edges. If $G'_i, i=1, \cdots, l+2$ is empty for some i then we fill these edges as virtual edges. All of the edges of G between A_i and $G_{i+1}, i=1, \cdots, l+2$ are in the structure as real edges.

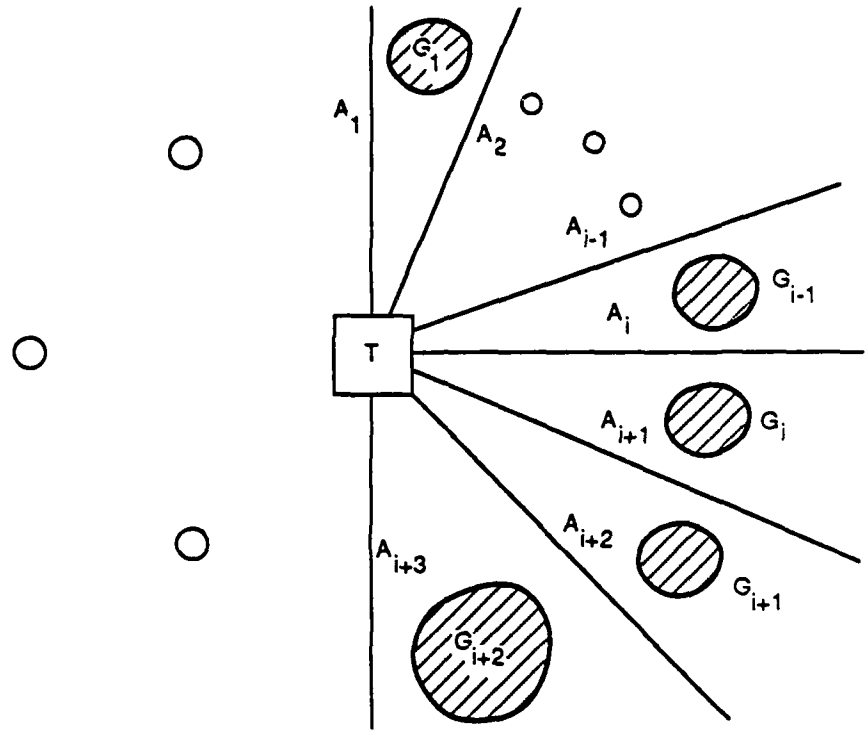


Figure 5.5.
Illustrating the representation of the separating k-sets
of Case 1 Types 3 and 4
(Structure 1).

Let us see where the rest of the separating k-sets lie assuming there are no cross separating k-sets of type 1 and type 2. Note that we allow separating k-sets of type 3. Let us first define exceptional separating k-sets. A separating k-set is *exceptional* if it separates only part of A_i and nothing else for some $i=1, \cdots, l+2$.

Lemma 5.2. All separating k-sets which are not covered by the structure 2 and not of type 1 and 2 and not exceptions are inside $G_i \cup A_i$ and its fringes inside $G_{i-1} \cup A_{i+1}$ and its fringes inside G_{i+1}.

Proof: Since there are no type 1 and type 2 and no exceptions in separating k-sets, no separating k-set uses T. There are also no cross separating k-sets which are not covered by the structure 1. Let us see what happens if a separating k-set crosses some $A_i, i=1, \cdots, l+2$ (see Figure 5.6). (Separating k-set divide G into G' and G'' and it is such that there is $x \in A_i$ and $x \in G'$ and there is $y \in A_i$ and $y \in G''$.)

W.L.O.G. let $E \cup F \cup H$ is this separating k-set, which crosses A_i, where $E \subset G_5$, $F \subset G_6$ and $H \subset A_i$. It divides A_i into A'_i, A''_i and H. It also divides G_5 into G'_i and G''_i, and it divides G_6 into G'_6 and G''_6. Both A''_i and A'_i are nonempty, otherwise the set Y is not maximal, or there is no cross separating k-set. If G''_5 and G''_6 are nonempty then $E \cup H \cup A''_i$ and $F \cup H \cup A''_6$ are separating sets with cardinalities bigger or equal to k. But both of them can not have cardinality bigger or equal to k, hence, one of G''_5 or G''_6 must be empty. W.L.O.G. let G''_6 be empty. Since $A_{l+1} \cup T \cup A_i$ and $A_{l+1} \cup T \cup A'_i \cup H \cup F$ are separating k-set and separating set, respectively $|F| \geq |A''_i|$. Since $E \cup H \cup A''_i$ is a separating set, and since both G''_5 and G''_6 cannot be empty (exception), we must have $|A''_i| \geq |F|$. Hence, $A''_i| = |F|$, and F is part of the fringe of A_i.

Let us see what happens if a cross separating k-set crosses (as defined above) two adjacent A_i's. W.L.O.G. $E \cup H_1 \cup F \cup H_2 \cup I$ is a separating k-set, which divides A_i into A'_i, H_1, and A''_i,

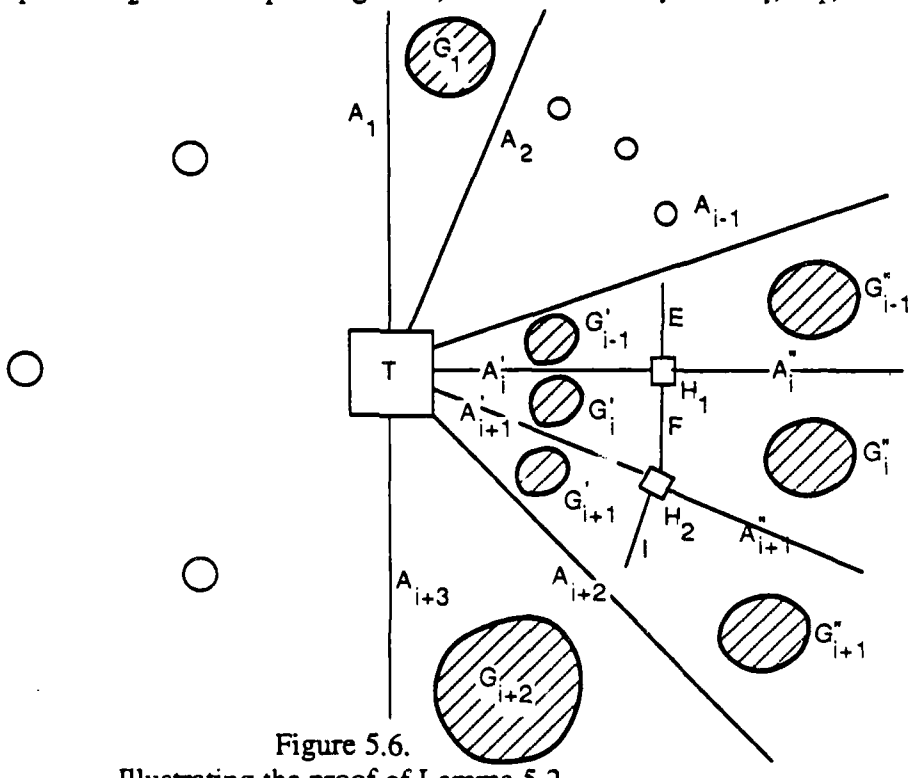


Figure 5.6.
Illustrating the proof of Lemma 5.2.

and divides A_{i+1} into A'_{i+1}, H_2, and A''_{i+1}. It separates G_{i-1} into G'_{i-1} and G''_{i-1}, it separates G_i into G'_i and G''_i, it separates G_{i+1} into G'_{i+1} and G''_{i+1}. By the above argument, G''_{i-1} and G''_{i+1} are empty, and E belongs to the fringe of A_i, and I belongs to the fringe of $A_{i=1}$. Note that we don't need to use the assumption that there are no exceptions. A cross separating k-set cannot cross three adjacent A_i's, since with respect to the middle A_i neither of G''_5 and G''_6 can be empty. Hence, all other separating k-sets, except exceptions, belong to $G_i \cup A_i \cup$ { its fringes in G_{i-1} } $\cup A_{i+1} \cup$ { its fringes in G_{i+1} }.

□ Lemma 5.2.

Let us now consider exceptions. W.L.O.G. let there exist an exceptional separating k-set, which separates part of A_i. In other words, there is a separating k-set which separates part of A_i (A'_i), such that all of the vertices not in $A_i \cup T$ are neighbors of A'_i. The number of the neighbors of A'_i in $G_{i-1} \cup A_{i-1} \cup G_i \cup A_{i+1}$ is less than k. Consider the minimal set of subsets of A_i that covers all vertices of A_i which can be separated by some exceptional separating k-set. The number of subsets in this set is less than or equal to the cardinality of A_i, hence is at most $\frac{k-t}{2}$. The number of neighbors of A_i that are used for separating these subsets is less than or equal to k vertices per subsets, so their total is at most $\frac{k^2}{2}$. Note that $\frac{k^2}{2}-k$ such vertices can be inside either $G_{i-1} \cup A_{i-1}$ or $G_i \cup A_{i+1}$. Moreover, if $v \in A_i$ participates in some subset of A_i, that can be separated by an exceptional separating k-set, then v has less than k vertices in $G_{i-1} \cup A_{i-1} \cup G_i \cup A_{i+1}$. Hence, if we take the union of the following sets

1) $G_i \cup A_i \cup A_{i+1}$

2) the neighbors of A_i in $G_{i-1} \cup A_{i-1}$, that are used for exceptional separating k-sets

3) the fringe of A_i

4) the neighbors of A_{i+1} in $G_{i+1} \cup A_{i+2}$, that are used for exceptional separating k-sets

5) the fringe of A_{i+1}

for all i's we will obtain all separating k-sets which are not covered by the structure.

The number of exceptional separating k-set for A_i is bounded by the number of different subsets of A_i. Hence, it is less than or equal to $2^{\frac{k-t}{2}}$. Thus, the number of exceptional separating k-sets is at most $(l+2)2^{\frac{k-t}{2}}$.

Based upon Lemma 5.1 and the above observation about exceptions, and using structure 1, we can write the following recurrence, which is valid if there are no type 1 or type 2 separating k-sets:

$$g(n) \le \max_{l}\left(\sum_{i=1}^{l+2} g(n_i+k(k-t)+t) + (l+2)\left(\frac{k-t}{2}\right)k + t\right),$$

where every term inside the sum covers one of the G_i's, and $(l+2)(\frac{k-t}{2}) + t$ is the upper bound on the size of the structure 1. Note that $\sum_{i=1}^{l+2} n_i + \frac{(l+2)(k-t)}{2} + t = n$. Note also that for some of the $g(n_i+k(k-t)+t)$ we might not be in Case 1 anymore, then we will use the recurrences for Cases 2 and 3. Once we enter Case 2 or 3 we can never return to Case 1. The solution to this recurrence is $O(kn + k^3)$ (see Appendix 5.2). Note that each $(n_i + k(k-t)+t)$ is less than n itself. We will show later that solutions for Case 2 and 3 are lower than above solution.

Analogously, the recurrence for the upper bound on the number of separating k-sets become

$$f(n) \le \max_{l}\left(\sum_{i=1}^{l+2} f(n_i+k(k-t)+t) + 2^{k-t} l \frac{l+2}{2} + 2^{\frac{k-t}{2}}(l+2)\right).$$

The solution to this recurrence is $O(2^k \frac{n^2}{k})$. Note that all cross separating k-set of type 3 are also covered by these recurrences.

Now we will look at type 1. Let $T_2 \cap A = T'_2$, $T_2 \cap B = T''_2$, and $T_1 \cap T_2 = \bar{T}_2$. With respect to a new cross separating k-set which uses T_2 some G_i i=3,4,5,6 could be empty. Let us first look at a harder case when none of G_i i=3,4,5,6 are empty with respect to a new cross separating k-set.

A new cross separating k-set must cross C and D of the old cross separating k-set which uses T_1, otherwise the Claim 5.1 with respect to the new cross separating k-set will be violated (see Figure 5.7).

Second, $\bar{T}_2 = T_1$, otherwise Claim 5.1 will be contradicted for the old cross separating k-set.

Third, $C'_1 \cup C'_2 \cup H_1 \cup T_1 \cup T''_2$, $C''_1 \cup C''_2 \cup H_1 \cup T_1 \cup T'_2$, $D'_1 \cup D'_2 \cup H_2 \cup T_1 \cup T''_2$, and $D''_1 \cup D''_2 \cup H_2 \cup T_1 \cup T''_2$ are separating sets with cardinalities less than k, which separate G''_4, G''_3, G''_6, and G''_5, respectively. Hence, G''_3, G''_4, G''_5, and G''_6 are empty.

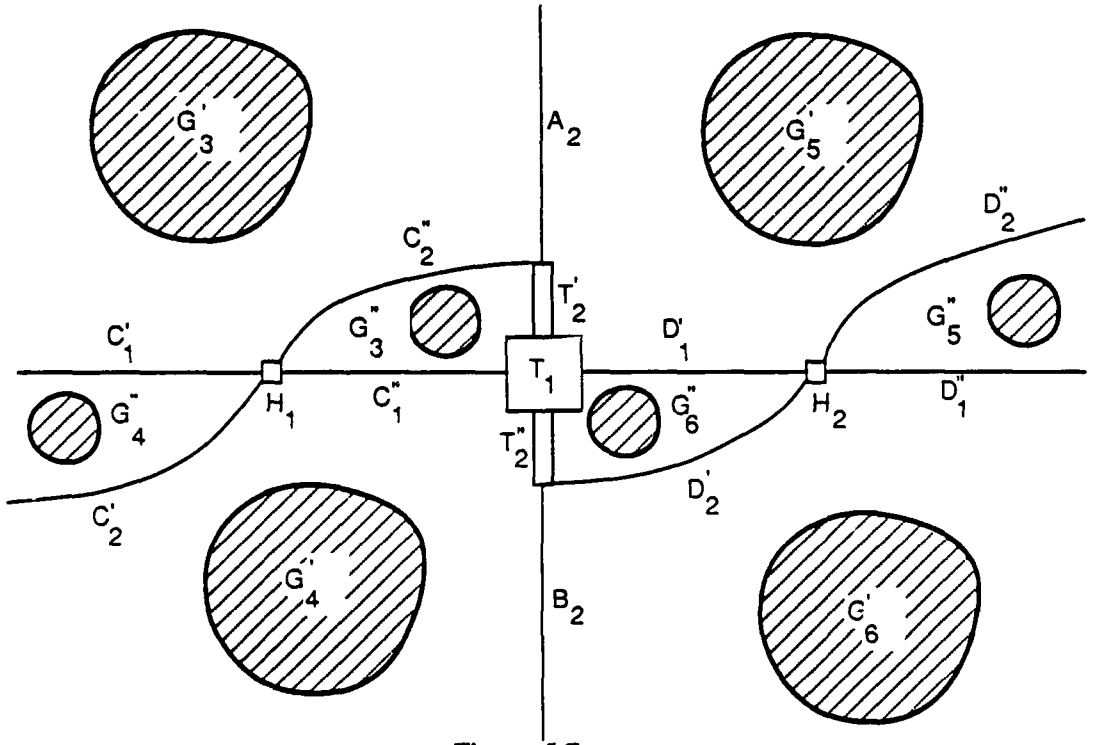


Figure 5.7.
Illustrating the configuration between two cross separating k-sets which use different T's.

Fourth, $C'_1 \cup H_1 \cup C''_2 \cup T_2 \cup D'_2 \cup H_2 \cup D''_2$, $C'_2 \cup H_1 \cup C''_2 \cup T_2 \cup D'_2 \cup H_2 \cup D''_1$, $C'_2 \cup H_1 \cup C''_1 \cup T_2 \cup D'_2 \cup H_2 \cup D''_2$, and $C'_2 \cup H_1 \cup T_2 \cup D'_1 \cup H_2 \cup D''_2$ are separating sets. Hence, $|C'_1| \geq |C'_2|$, $|D'_1| \geq |D'_2|$, $|C''_1| \geq |C''_2|$, and $|D''_1| \geq |D''_2|$. Also, $C'_1 \cup H_1 \cup C''_2 \cup T'_2 \cup T_1 \cup D'_1 \cup H_2 \cup D''_1$, $C'_2 \cup T'_2 \cup H_1 \cup C''_1 \cup T_1 \cup D'_1 \cup H_2 \cup D''_1$, $C'_1 \cup H_1 \cup C''_1 \cup T_1 \cup T'_2 \cup D'_2 \cup H_2 \cup D''_1$, and $C'_1 \cup H_1 \cup C''_1 \cup T_1 \cup T'_2 \cup D'_1 \cup H_2 \cup D''_2$ are separating sets. Hence,

$$\begin{cases} |C'_2| + |T'_2| \geq |C'_1| \geq |C'_2| > 0 \\ |C''_2| + |T'_2| \geq |C''_1| \geq |C''_2| > 0 \\ |D'_2| + |T''_2| \geq |D'_1| \geq |D'_2| > 0 \\ |D''_2| + |T'_2| \geq |D''_1| \geq |D''_2| > 0 \end{cases}$$

Also since we are still in Case 1 with respect to both old and new cross separating k-sets, we have the following equalities

$$\begin{cases} |T'_2| = |T''_2| \\ |A_2| = |B_2| = |D'_2| + |H_2| + |D''_2| = |C'_2| + |H_1| + |C''_2| \end{cases}$$

Note that the set T'_2 has edges to the set D''_1, the set T''_2 has edges to the set D'_1, the set T''_2 has edges to the set C'_1, and the set T'_2 has edges to the set C''_1, because of Claim 5.1 with respect to the new cross separating k-set. Hence, the maximal disjoint sets for C's and D's (X and Y) will have cardinalities equal to 1.

Let us take a maximal T_2, and let us take the fringes of A_2, B_2, C and D (see Figure 5.8).

C'_1 does not have the fringe in G_4, otherwise part of C'_1 which has a fringe becomes a part of I'_1. If C'_1 has the fringe in G_3 then the part of C'_1 which has the fringe can be separated from the rest of the graph by a separating set $C'_2 \cup T''_2 \cup T_1 \cup \{$the fringe of C'_1 in $G_3\}$, whose cardinality is less than k. Hence, C'_1 does not have the fringe. Analogously, C''_1, D'_1, and D''_1 do not have the fringes. Symmetrically, T'_2 and T''_2 do not have the fringes.

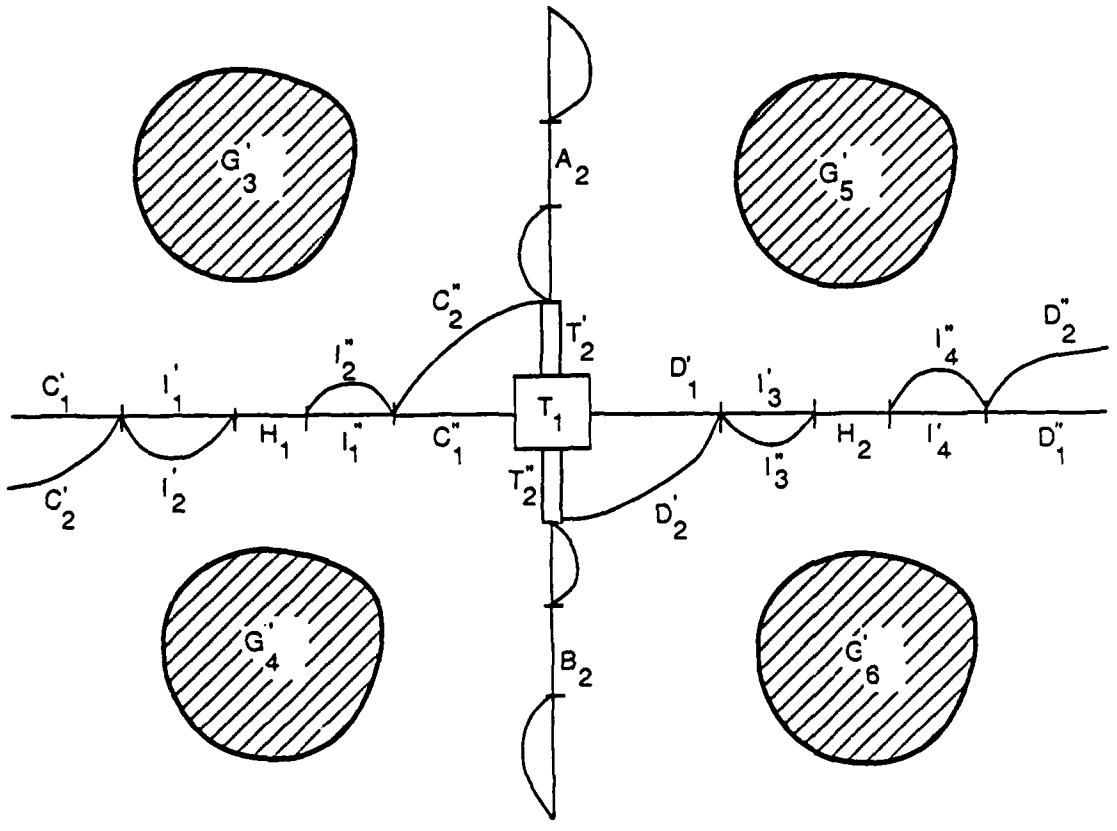


Figure 5.8.
Illustrating the representation of separating k-sets of Case 1
if two or more different intersecting T's exist.
(Structure 2).

Let $\hat{T}_2$ be the union of vertices which are used for all possible T_2 which create a cross separating k-sets with nonempty G_i i=3,4,5,6. Let $\hat{D'}_1$ be the union of all possible D'_1, $\hat{D''}_1$ be the union of all possible D''_1, $\hat{C'}_1$ be the union of all possible C'_1, $\hat{C''}_1$ be the union of all possible C''_1, $\hat{C'}_2$ be the union of all possible C'_2, $\hat{C''}_2$ be the union of all possible C''_2, $\hat{D'}_2$ be the union of all possible D'_2, and $\hat{D''}_2$ be the union of all possible D''_2. Let us show that all of these sets are disjoint.

Since all of them are symmetric we will prove it only for $\hat{C'}_1$ and $\hat{C''}_1$. Assume there are T_3 and T_4 such that C''_1 for T_3 is not disjoint from C'_1 for T_4. Then $C''_1 \cap C'_1$ is nonempty and is separated from the rest of the graph by a separating set C''_2 for $T_3 \cup T'_3 \cup T_1 \cup T''_4 \cup C'_2$ for T_4, whose cardinality is less than k. This contradiction proves the statement.

The cardinality of the union $\hat{D''}_2 \cup \hat{D'}_2 \cup I''_4 \cup I'_4$ is less than $\frac{k-t}{2}$, and analogously, the cardinality of $\hat{C''}_2 \cup \hat{C'}_2 \cup I'_1 \cup I''_2$ is less than $\frac{k-t}{2}$. Let us call $\hat{C'}_2$, $\hat{C''}_2$, $\hat{D'}_2$, and $\hat{D''}_2$ the *pseudofringe*. Note that A and B might have fringes , but by the symmetry $\hat{T}_2 - T_1$ does not have any fringes.

The structure which represents all separating k-sets for all possible T's will be the following (structure 2) (see Figure 5.8):

1) the original separating k-set with its fringes,

2) the cross separating k-set with minimum cardinality T_1 with its fringes and pseudofringes,

3) for every nonempty G'_i i=3,4,5,6 we will fill all nonexistent edges of the complete graph on the neighbors of G'_i. If G'_i is empty for any i=3,4,5,6 we will fill these nonexistent edges of this complete graph by the virtual edges. (For G'_3 we fill the edges between the vertices of the fringe of A in G_3, T_1, $\hat{T'}_2$, part of A_2 which does not have any fringes, $\hat{C'}_1$, I'_1, H_1, I''_2 and $\hat{C''}_2$).

From the construction of the structure it is easy to see that this structure covers all cross separating k-sets for all possible T's, of type 1. Let us see now where the rest of the separating k-sets lie, if we have separating k-sets of type 1.

If there exists T_2 with at least one of the G_i empty i=3,4,5,6, assuming it is not exception, such that there is another T_2 with $T_2 \cap T_1$ nonempty along with nonempty $T_2 \cap B$ and $T_2 \cap A$, then all cross separating k-sets of this T_2 are covered by the above structure. (They belong to the fringes of A and/or B in G_1 or G_2 and the rest belong to the original cross separating k-set with its fringes or pseudofringes). So all cross separating k-sets are covered by this structure, assuming there are no exceptions, hence, all separating k-sets are either inside $G_1 \cup A \cup B \cup T_1 \cup$ the fringes of A and B in G_2, or $G_2 \cup A \cup B \cup T_1 \cup$ the fringes of A and B in G_1, or cross separating k-sets covered by the structure. Since the structure is symmetric, we can look at the cross

separating k-sets where the original separating k-set is $C \cup D \cup T_1$. Then the pseudofringes of C and D become the pseudofringes of A and B. With respect to this separation of G all separating k-sets are either inside $G_3 \cup G_5 \cup C \cup D \cup T_1 \cup$ the fringe of C in G_4 and the fringe of D in G_6, or inside $G_4 \cup G_6 \cup C \cup D \cup T_1 \cup$ the fringe of C in G_3 and the fringe of D in G_5, or separating k-sets covered by the structure. But since in both cases they are the same separating k-sets, all separating k-sets are either inside $G_3 \cup A \cup T_1 \cup C \cup$ the fringe of C in $G_4 \cup$ the fringe of A in G_5, or inside $G_4 \cup B \cup C \cup T_1 \cup$ the fringe of B in G_6, or inside $G_5 \cup A \cup D \cup T_1 \cup$ the fringe of A in $G_3 \cup$ the fringe of D in G_6, or inside $G_6 \cup B \cup D \cup T_1 \cup$ the fringe of B in $G_4 \cup$ the fringe of D in G_5, or the separating k-sets covered by the structure. To cover all exceptions we will do what we did for types 3 and 4 separating k-sets, we will add $k(k-t)$ neighbors of A, B, C and D to each of G_3, G_4, G_5 and of G_6 which can participate in exceptional separating k-sets. Hence, the size of representation is

$$g(n) = \sum_{i=1}^{4} g(n_i + k(k-t)+t) + 8\frac{(k-t)}{2}k + t ,$$

where every term inside the sum covers one of G_i i=3,4,5,6 along with its appropriate neighbors and fringes, and $8\frac{(k-t)}{2}k + t$ is the upper bound on the size of the structure. Note that $\sum_{i=1}^{4} n_i + 2k - t = n$, hence the solution to the above recurrence is $O(nk + k^3)$ (see Appendix 5.2). (Note also that the above recurrence and the recurrence for Types 3 and 4 are basically the same up to a constant factor, hence we can modify the recurrence for Types 3 and 4 to incorporate all three types). The number of exceptional separating k-sets is upper bounded by $4 \cdot 2^{\frac{k-t}{2}}$. The upper bound on the number of separating k-sets becomes

$$f(n) = \sum_{i=1}^{4} f(n_i + k(k-t)+t) + \left[\begin{matrix}4\\2\end{matrix}\right] \cdot 2^{k-t} + 4 \cdot 2^{\frac{k-t}{2}} .$$

The solution to it is $O(2^k n + 2^k k^2)$ (see Appendix 5.2).

Let us now see what happens if we are in type 2 and no separating k-sets of type 1 exist. W.L.O.G. assume there is a separating k-set which uses $T_2 = T'_2 \cup \bar{T}_2$, where $T'_2 \in A$ and $\bar{T}_2 \in T_1$, and no separating k-set of type 1 exist (see Figure 5.9). If G_i's i=3,4,5,6 are nonempty with respect to a new cross separating k-set then we revert to Case 1 with respect to a new cross separating k-set, hence $|A_2| = |B|$ which is impossible. Hence, one of the G_i i=3,4,5,6 with respect to a new cross separating k-set must be empty. W.L.O.G. let the empty G_i be either G_3 or G_4 with respect to the new cross separating k-set. If G_4 is empty then G_5 with respect to the new cross separating k-set must be empty, otherwise $T_1 \cup T'_2 \cup A_2 \cup D_2$ of the new cross separating k-set becomes a separating set with cardinality less than k. Hence, if G_4 is empty then all cross separating k-sets of type 2 belong to the original separating k-set with its fringes. Then all separating k-sets are inside $G_1 \cup A \cup B \cup T_1 \cup$ the fringe of A in $G_5 \cup$ {the fringe of B in G_6}, or inside $G_2 \cup A \cup B \cup T_1 \cup$ {the fringe of A in G_3} $\cup$ the fringe of B in G_4, or they belong to the union of $A \cup B \cup T_1 \cup$ {the fringes of A and B}. Note that in the third case the separating k-sets

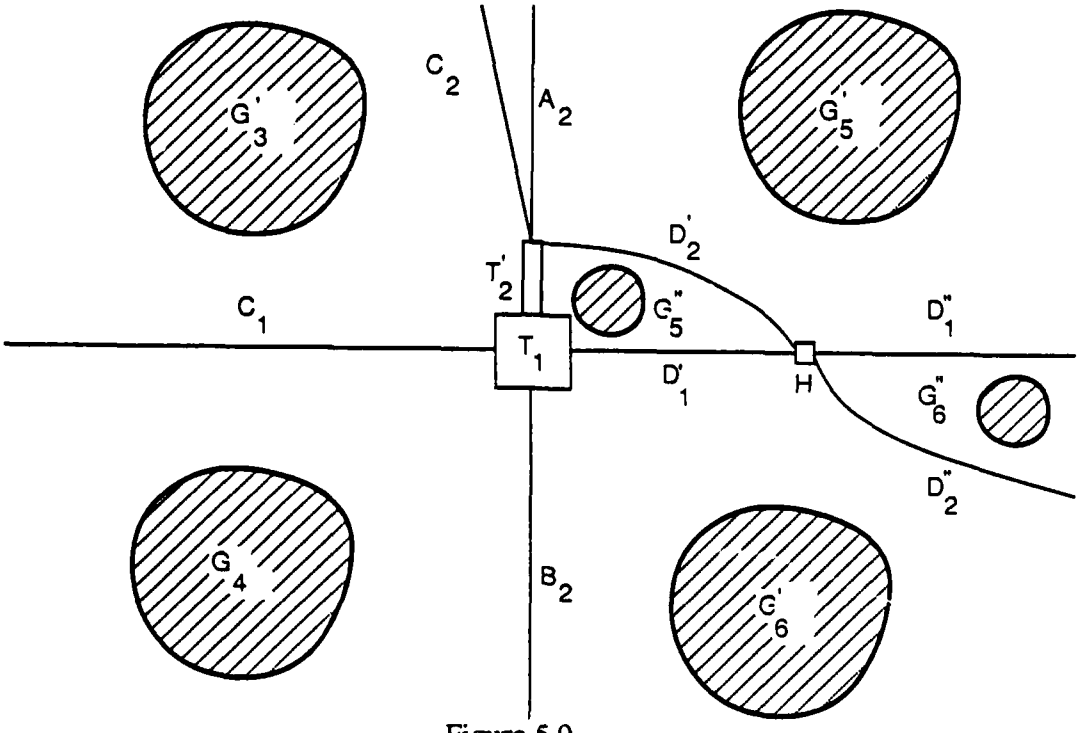


Figure 5.9.
Illustrating type 2 separating k-set when no type 1 separating k-set exist.

are covered by structure 2. We can write the recurrences similar to the above ones except for the sum which will be up to 2 instead of up to 4. The solution will be still of the same order. If G_3 is empty then $|C_2| \geq |A_2|$, otherwise $C_2 \cup T'_2 \cup T_1 \cup B$ is a separating set with cardinality less than k. If D_2 crosses D_1 (see Figure 5.6) then $A_2 \cup T'_2 \cup T_1 \cup D_2$ is a separating set, so $|C_2| = |A_2|$. $C \cup T_1 \cup D'_1 \cup H \cup D''_2$ is a separating set, so $|D''_2| \geq |D''_1|$. Also $C_2 \cup T_2 \cup D'_2 \cup H \cup D''_1$ is a separating set, so $|D''_1| \geq |D''_2|$. Combining these two we get $|D''_1| = |D''_2|$. Since, $C \cup T_1 \cup T'_2 \cup D'_2 \cup H \cup D''_1$ and $C_2 \cup T'_2 \cup T_1 \cup D'_1 \cup H \cup D''_2$ are separating sets, so $|T'_2 \cup D'_2| \geq |D'_1| \geq |D'_2|$. Since $T_1 \cup D'_1 \cup H \cup D'_2$ separates G''_6 from the rest of the graph, and since the cardinality of this separating set is less than k, G''_6 is empty. Hence, D''_2 belongs to the fringe of D in G_6. $\bar{T}_2 = T_1$ in order for the Claim 5.1 with respect to the old cross separating k-set to be true. And since $|C_2| + |T'_2| = |A|$ and since the cardinality of the new cross separating k-set is k, $|D'_2| = |D'_1|$. So, all cross separating k-sets of this type belong to $G_5 \cup A \cup D \cup T_1 \cup$ the fringe of A in $G_3 \cup$ the fringe of D in G_6, if there are no exceptional separating k-sets. Also in the maximal set of disjoint D's (Y) all of D's except D_1 belong to G_6. If G_5 with respect to the new cross separating k-set is nonempty, then by the above argument C_2 will belong to the fringe of A. Hence, all cross separating k-sets belong to the set mentioned above, namely, $G_4 \cup A \cup T \cup D_1 \cup$ the fringe of A in $G_1 \cup$ the fringes of D_1 in G_5.

Let us take the maximal set of C's and D's (X and Y). We know that all cross separating k-sets of type 2 with nonempty G_5 belong to $G_5 \cup A \cup D \cup T_1 \cup$ the fringe of A in $G_3 \cup$ the fringe of D in G_6. Since we need to consider all symmetric cases, and since we don't have any cross separating k-sets of type 1, all cross separating k-sets of the type 2 belong to $G_3 \cup A \cup C \cup T_1 \cup$ the fringe of A in $G_5 \cup$ the fringe of C in G_4, or $G_4 \cup B \cup C \cup T_1 \cup$ the fringe of B in $G_6 \cup$ the fringe of C in G_3, or $G_5 \cup A \cup D \cup T_1 \cup$ the fringe of A in $G_3 \cup$ the fringe of D in G_6, or $G_6 \cup B \cup D \cup T_1 \cup$ the fringe of B in $G_4 \cup$ the fringe of D in G_5. Note that C's and D's are not the same in these sets. In case of G_3 $C = C_a \in X$ in case of G_4 $C = C_1 \in X$, in case of G_5

$D = D_1 \in Y$, and in case of G_6 $D = D_b \in Y$. Let us see where the rest of separating k-sets must lie. First, if there are no cross separating k-sets with G_5 nonempty (or some other appropriate symmetric G_i i=3,4,5,6) then it is still possible to have cross separating k-sets.

All cross separating k-sets consist of three parts: part one is in G_1, part two is in G_2 and part three is T_1. Part one belongs to some C from the set X or its fringe or the fringe of A in G_3 or the fringe of B in G_4. Part two belongs to some D from the set Y or its fringe or the fringe of A in G_5 or the fringe of B in G_6. That covers all cross separating k-sets which use T_1, otherwise either set X or set Y is not maximal. We don't have any cross separating k-sets of type 1. All cross separating k-sets of type 2 with nonempty appropriate G_i with respect to them belong to the part of the graph between A and the nearest D in G_2 along with A and its fringe and D and its fringe. Hence, all other separating k-sets belong to $G_1 \cup A \cup B \cup T_1$ with its fringes, or $G_2 \cup A \cup B \cup T_1$ with its fringes.

Hence, all cross separating k-sets of type 2, except exceptions are covered by the structure 2 or inside the subgraphs associated by G_1, G_{l_1+1}, G_{l_1+2} and G_{l+2}. As for the exceptions the upper bounds we got for types 3 and 4 still hold, since no part of T_1 can be separated by them (otherwise Claim 5.1 is contradicted). So, the recurrence which was written for the type 3 and 4 separating k-sets covers type 2 cross separating k-sets also, including exceptions.

Combining all four types we get the following recurrence for representation of separating k-sets

$$g(n) \le \max_{l} \left(\sum_{i=1}^{l+2} g(n_i + k(k-t)+t) + 2(l+2)\left(\frac{k-t}{2}\right)k + t \right),$$

whose solution is the same as the one for Types 3 and 4 up to a constant. The recurrence for the upper bound on the number of separating k-sets is

$$f(n) \le \max_{l}\left(\sum_{i=1}^{l+2} f(n_i+k(k-t)+t) + 2^{k-t} l \frac{l+2}{2} + 2^{\frac{k-t}{2}}(l+2)\right),$$

whose solution is the same as the one for Types 3 and 4 up to a constant. This concludes Case 1.

□ Case 1.

Case 2 For any separating k-set every cross separating k-set will have one of the G_i i=3,4,5,6 empty. Not every vertex in both G_1 and G_2 can be used for cross separating k-sets.

W.L.O.G. let G_3 be empty (see Figure 5.10).

Since G_4 is nonempty by assumption, and G_5 is nonempty since there are no exceptions, $C \cup T \cup B$ and $A \cup T \cup D$ are separating sets. So their cardinalities are bigger or equal to k, hence, $|C| = |A|$ and $|B| = |D|$. So, C is part of the fringe of A in G_1. Since this is true for every T, all cross separating k-sets belong to $G_1 \cup A \cup T \cup B \cup$ {the fringes of A and B in G_2}, or

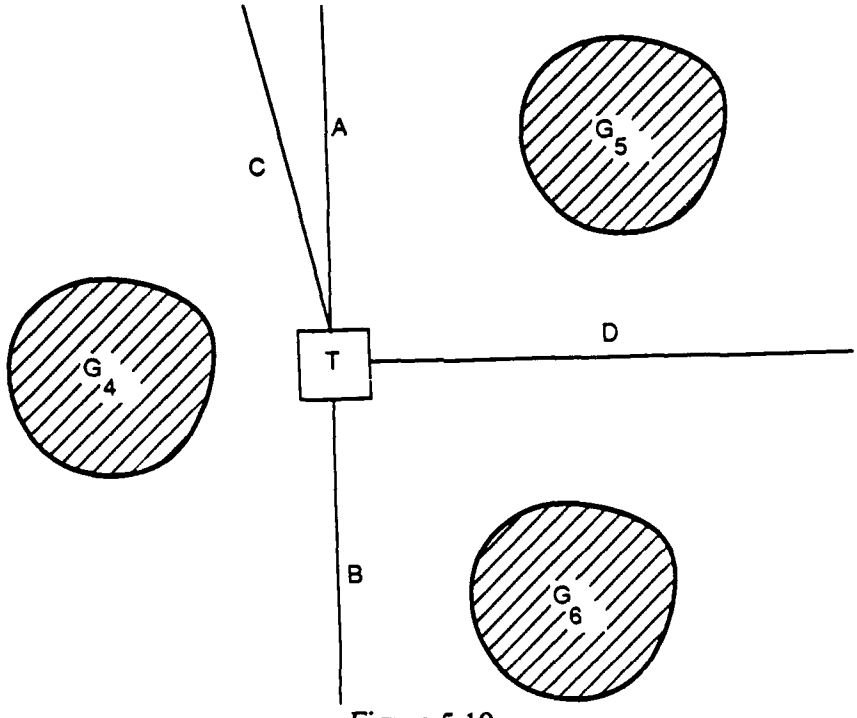


Figure 5.10.
Illustrating Cases 2 and 3.

$G_2 \cup A \cup T \cup B \cup$ {the fringes of A and B in G_1}, except for exceptions. Let E be the set of vertices in G_1 which are used for exceptional separating k-sets which separate parts of A or B. Analogously, let F be the set of vertices in G_2 which are used for exceptional separating k-sets which separate parts of A or B. The cardinalities of E and F are at most k^2. So all separating k-sets including the exceptions are either inside $G_1 \cup A \cup B \cup T \cup E$ or inside $G_2 \cup A \cup B \cup T \cup F$. Hence,

$$g(n) = g(n_1 + k(k-1)) + g(n_2 + k(k-1)) + 4k^2,$$

where n_1 and n_2 are the cardinalities of G_1 and G_2. We still have that $n_1 + n_2 + k = n$, and the solution to this recurrence is $O(k^2 + n)$ (see Appendix 5.2). Note that $n_i + k(k-1) < n$ for i=1,2. Note also that for $g(n_1 + k(k-1))$ or $g(n_2 + k(k-1))$ all separating k-sets may be of Case 3 only, but once we enter Case 3 we cannot return back to Case 2.

For the upper bound on the number of separating k-sets we get the following equality

$$f(n) = f(n_1 + 2k) + f(n_2 + 2k) + 2^k,$$

where 2^k covers all exceptional separating k-sets. And its solution is clearly smaller than $O(2^k \frac{n^2}{k})$ (see Appendix 5.2). This concludes Case 2.

□ Case 2.

Case 3 For every separating k-set all cross separating k-sets are lopsided (one of the G_i i=3,4,5,6 will be empty). And either G_1 or G_2 is such that every vertex of them is used for some cross separating k-set.

W.L.O.G. let G_3 be empty and let $G_4 \cup C$ correspond to the smallest G_1 such that every vertex of G_1 is used for some cross separating k-set (see Figure 5.10). There are two subcases: either G_5 is empty or G_6 is empty. If neither is empty we will be in Case 2. Take C as large as possible.

If G_6 is empty then $A \cup B \cup C \cup D \cup T$ with all edges between them and filling real edges for nonempty G_5 and G_4 and virtual otherwise (analogous to the structure 1) will specify all cross separating k-sets. If G_5 is empty, then $C \cup T \cup D$ separates A from the rest of the graph. Hence, $C \cup T \cup D$ is an exceptional separating k-set. So the third structure will be the following:

1) A, B and T - the original separating k-set,

2) All the neighbors of $A \cup B \cup T$ that are used for a cross separating k-sets with edges between them and the original separating k-set.

Since there are no other separating k-sets, otherwise we would be in Case 2, we derive the following upper bound for the size of the representation:

$$g(n) \leq k^2.$$

Analogously, we have the following upper bound on the number of separating k-sets

$$f(n) \leq 2^k.$$

□ Case 3.

That concludes the proof of all cases. Our final result is that all separating k-sets have $O(k^2 n)$ space representation, and their number is $O(2^k \frac{n^2}{k})$.

From the above representation we can generate all separating k-set in $O(c^k M)$ time, where M is the number of separating k-sets. For that we will take every possible subset of every spoke of the representation and run a matching algorithm to find replacement for this subset. That can be done in $O(polylogk)$ time, since the size of the graph is $O(k^2)$. Each replacement generates a separating k-set of a graph.

5.2. Appendix

The solution to the recurrence which we get for the representation of Type 3 and 4 separating k-sets of Case 1:

$$g(n) \le \max_{l}\left(\sum_{i=1}^{l} g(n_i + (k-t)k + t) + lk\frac{(k-t)}{2} + t\right)$$

with initial conditions

$$\sum_{i=1}^{l}\left(n_i + \frac{k-t}{2}\right) + t = n \qquad 0 \le t \le k-2 \qquad 2 \le l \le 2\frac{n-t}{k-t} \qquad n_i \ge 0$$

Let $g(n) = 2nk - 4k^3 + 2k^2t + \frac{1}{2}k^2 - 3kt - t$,

$$g(n) \le \max_{l}\left(\sum_{i=1}^{l} g(n_i + (k-t)k + t) + lk\frac{k-t}{2} + t\right) \le$$

$$\max_{l}\left(\sum_{i=1}^{l} 2k(n_i + k(k-t) + t) - 4k^3 l + 2k^2 tl + \frac{1}{2}k^2 l - ktl - tl + lk\frac{k-t}{2} + t\right) =$$

$$\max_{l}\left(2k\left(\sum_{i=1}^{l}\left(n_i + \frac{k-t}{2}\right) + t\right) - 2kl\frac{k-t}{2} - 2kt + 2k^2 l(k-t) + 2ktl - 4k^3 l +\right.$$

$$\left. 2k^2 tl + \frac{1}{2}k^2 l - 3ktl - tl + lk\frac{k-t}{2} + t\right) =$$

$$\max_{l}\left(2kn + 2k^3(l - 2l) + 2k^2 t(-l + l) + k^2\left(\frac{1}{2}l + \frac{l}{2} - l\right) + kt\left(l - 2 + 2^l - \frac{l}{2} - 3l\right) + t(-l + 1)\right) \le$$

$$2kn - 4k^3 - 3kt + t \le 2kn - 4k^3 + 2k^2 t + \frac{1}{2}k^2 - 3kt - t$$

Hence, $g(n) = O(nk + k^3)$.

The solution to the recurrence which we get for the upper bound of Type 3 and 4 separating k-sets of Case 1:

$$f(n) \le \max_{l}(\sum_{i=1}^{l} f(n_i + k(k-t)+t) + 2^{k-t}\frac{l(l-2)}{2} + 2^{\frac{k-t}{2}} l)$$

with initial conditions

$$\sum_{i=1}^{l}(n_i + \frac{k-t}{2}) + t = n \qquad 2 \le l \le 2\frac{k-t}{k-t} \qquad 0 \le t \le n-2$$

Let

$$f(n) = 2^{k-t}nl - 2^{k-t}k^2l + 2^{k-t}ktl + \frac{1}{2}2^{k-t}kl - \frac{3}{2}2^{k-t}tl + 2^{k-t}kt +$$

$$\frac{1}{2}2^{k-t}k - 2\,2^{k-t}k^2 - 2^{k-t}t - \frac{1}{2}2^{k-t}l - 2\,2^{\frac{k-t}{2}},$$

$$f(n) \le \max_{l}(\sum_{i=1}^{l}(n_i k(k-t)+t)2^{k-t}l - 2^{k-t}k^2l^2 + 2^{k-t}ktl^2 + \frac{1}{2}2^{k-t}kl^2 - \frac{3}{2}2^{k-t}tl^2 + 2^{k-t}ktl +$$

$$\frac{1}{2}2^{k-t}kl - 2\,2^{k-t}k^2l - 2^{k-t}tl - \frac{1}{2}2^{k-t}l^2 - 2\,2^{\frac{k-t}{2}} + \frac{1}{2}2^{k-t}l^2 - \frac{1}{2}2^{k-t}l + 2^{\frac{k-t}{2}} l) = \max_{l}(2^{k-t}\ln -$$

$$\frac{1}{2}2^{k-t}kl^2 + \frac{1}{2}2^{k-t}tl^2 - 2^{k-t}tl + 2^{k-t}k^2l^2 - 2^{k-t}ktl^2 + 2^{k-t}tl^2 - 2^{k-t}k^2l^2 + 2^{k-t}ktl^2 + \frac{1}{2}2^{k-t}kl^2 -$$

$$\frac{3}{2}2^{k-t}tl^2 + 2^{k-t}ktl + \frac{1}{2}2^{k-t}kl - 2\,2^{k-t}k^2l - 2^{k-t}tl - \frac{1}{2}2^{k-t}l^2 - 2\,2^{\frac{k-t}{2}} l + \frac{1}{2}2^{k-t}l^2 - \frac{1}{2}2^{k-t}l + 2^{\frac{k-t}{2}} l) =$$

$$\max_{l}(2^{k-t}\ln - 2\,2^{k-t}k^2l + 2^{k-t}ktl + \frac{1}{2}2^{k-t}kl - 2\,2^{k-t}kl - 2\,2^{k-t}tl - \frac{1}{2}2^{k-t}l - 2^{\frac{k-t}{2}} l) \le$$

$$\max_{l}(2^{k-t}\ln - 2^{k-t}k^2l + 2^{k-t}ktl + \frac{1}{2}2^{k-t}kl - \frac{3}{2}2^{k-t}tl + 2^{k-t}kt +$$

$$\frac{1}{2}2^{k-t}k - 2\,2^{k-t}k^2 - 2^{k-t}t - \frac{1}{2}2^{k-t}l - 2\,2^{\frac{k-t}{2}})$$

Hence, $f(n) = O(2^k\frac{n^2}{k} + 2^k nk)$.

The solution to the recurrence which we get for the representation of Type 1 and 2 separating k-sets of Case 1:

$$g(n) \le \sum_{i=1}^{4} g(n_i + k(k-t) + t) + 8k\frac{k-t}{2} + t$$

with initial conditions

$$\sum_{i=1}^{4} n_i + 2k - t = n \qquad 0 \le t \le k-2$$

Let $g(n) = 4nk - \frac{16}{3}k^3 + \frac{16}{3}k^2t + \frac{4}{3}k^2 - \frac{16}{3}kt - \frac{1}{3}t$,

$$g(n) \le \sum_{i=1}^{4} g(n_i + k(k-t) + t) + 4(k-t)k + t \le$$

$$\sum_{i=1}^{4}(4(n_i + k(k-t) + t)k - \frac{16}{3}k^3 + \frac{16}{3}k^2t + \frac{4}{3}k^2 - \frac{16}{3}kt - \frac{1}{3}t) + 4(k-t)k + t =$$

$$4k(\sum_{i=1}^{4} n_i + 2k - t) - 8k^2 + 4kt + 16k^3 - 16k^2t + 16kt - \frac{64}{3}k^3 + \frac{64}{3}k^2t + \frac{16}{3}k^2 -$$

$$\frac{64}{3}kt - \frac{4}{3}t + 4k^2 - 4kt + t =$$

$$4kn + k^3(16 - \frac{64}{3}) + k^2t(\frac{64}{3} - 16) + k^2(\frac{16}{3} - 8 + 4) + kt(4 + 16 - \frac{64}{3} - 4) + t(1 - \frac{4}{3}) =$$

$$4kn - \frac{16}{3}k^3 + \frac{16}{3}k^2t + \frac{4}{3}k^2 - \frac{16}{3}kt - \frac{1}{3}t$$

Hence, $g(n) = O(nk + k^3)$.

The solution to the recurrence which we get for the upper bound of Type 1 and 2 separating k-sets of Case 1:

$$f(n) \le \sum_{i=1}^{4} f(n_i + k(k-t) + t) + 6\,2^{k-t} + 4\,2^{\frac{k-t}{2}}$$

with initial conditions

$$\sum_{i=1}^{4}(n_i + \frac{k-t}{2}) + t = n \qquad 0 \le t \le n-2$$

Let $f(n) = 2^{k-t}n - \frac{4}{3}2^{k-t}k^2 + \frac{4}{3}2^{k-t}kt - \frac{5}{3}2^{k-t}t + \frac{2}{3}2^{k-t}k - 2\,2^{k-t} - \frac{4}{3}2^{\frac{k-t}{2}}$,

$$f(n) \le \sum_{i=1}^{4} f(n_i + k(k-t) + t) + 6\,2^{k-t} + 4\,2^{\frac{k-t}{2}} \le \sum_{i=1}^{4}(2^{k-t}(n_i + k(k-t) + t) - \frac{4}{3}2^{k-t}k^2 + \frac{4}{3}2^{k-t}kt -$$

$$\frac{5}{3}2^{k-t}t + \frac{2}{3}2^{k-t}k - 2\,2^{k-t} - \frac{4}{3}2^{\frac{k-t}{2}}) + 6\,2^{k-t} + 4\,2^{\frac{k-t}{2}} = 2^{k-t}n - 2^{k-t}k + 2\,2^{k-t}t - 2^{k-t}t +$$

$$4\,2^{k-t}k^2 - 4\,2^{k-t}kt + 4\,2^{k-t}t - \frac{16}{3}2^{k-t}k^2 + \frac{16}{3}2^{k-t}kt - \frac{20}{3}2^{k-t}t + \frac{8}{3}2^{k-t} - \frac{16}{3}2^{\frac{k-t}{2}} + 6\,2^{k-t} + 4\,2^{\frac{k-t}{2}} =$$

$$2^{k-t}n - \frac{4}{3}2^{k-t}k^2 + \frac{4}{3}2^{k-t}kt - \frac{5}{3}2^{k-t}t + \frac{2}{3}2^{k-t}k - 2\,2^{k-t} - \frac{4}{3}2^{\frac{k-t}{2}}$$

Hence, $f(n) = O(2^k n + 2^k k^2)$.

The solution to the recurrence for the Case 2 for the representation of separating k-sets

$$g(n) \leq g(n_1 + k(k-1)) + g(n_2 + k(k-1)) + 4k^2$$

with initial conditions

$$n_1 + n_2 + k = n \qquad n_1, n_2 \geq 0$$

Let $g(n) = n - 9k^2 + 3k$,

$$g(n) \leq n_1 + k^2 - k - 9k^2 + 3k + n_2 + k^2 - k - 9k^2 + 3k + 4k^2 = n - 9k^2 + 3k$$

Hence, $g(n) = O(n + k^2)$.

The solution to the recurrence for the Case 2 for the upper bound of separating k-sets

$$f(n) \leq f(n_1 + 2k) + f(n_2 + 2k) + 2^k$$

with initial conditions

$$n_1 + n_2 + k = n \qquad n_1, n_2 \geq 0$$

Let $f(n) = 2^k n - 3\,2^k k - 2^k$,

$$f(n) \leq 2^k n_1 + 2k2^k - 3\,2^k k - 2^k + 2^k n_2 + 2k2^k - 3\,2^k k - 2^k + 2^k = 2^k n - 3\,2^k k - 2^k$$

Hence, $f(n) = O(2^k n)$.

CHAPTER 6

ALGORITHMS FOR GRAPH FOUR-CONNECTIVITY

This chapter presents new sequential and parallel algorithms for graph four-connectivity. The new sequential algorithm for testing graph four-connectivity has time complexity $O(n^2)$ and is based upon an ear-decomposition technique. The new efficient parallel algorithm for testing graph four-connectivity runs in $O(log^2 n)$ time using $O(n^2)$ processors on a CRCW PRAM. Both algorithms represent improvements over previous algorithms for this problem. The algorithms actually compute the set of all separating triplets of the input graph G, and if G has no separating triplets, then G is four-connected.

6.1. Open Ear Decomposition and Graph Four-Connectivity

A triconnected graph is four-connected if and only if it does not have any separating triplets.

Lemma 6.1. Let $G=(V,E)$ be a triconnected undirected graph for which $t=(x,y,z)$ forms a separating triplet. Let **D** be an open ear decomposition for G. Then there exists an ear P_i in **D** that contains two of the three vertices in t, say x and y, such that $V_i(x,y)$ contains a vertex other than z, and every path from a vertex in $V_i(x,y)$ to a vertex in $V_i[x,y]$ in G_i passes through x, y, or z. Further ear P_i uniquely determines a connected component C in the subgraph induced by $V-\{x,y,z\}$, in the sense that there is no ear P_j in G such that:

1). P_j contains x and y.

2). P_j contains a vertex in C,

3). $V_j(x,y)-\{x,y,z\}$ is nonempty, and

4). every path from a vertex in $V_j(x,y)$ to a vertex in $V_j[x,y]$ in G_j passes through x, y, or z.

Proof: Since $t=(x,y,z)$ forms a separating triplet, the subgraph of G induced by $V-\{x,y,z\}$ contains at least two connected components. Let C_1 and C_2 be two such connected components.

Case 1 The first ear P_0 contains no vertex in C_2 (see Figure 6.1):

Consider the lowest-numbered ear, P_i, that passes through a vertex v in C_2. Since its endpoints are distinct and must be contained in lower-numbered ears, P_i must enter C_2 through one of the three vertices in t, say x, and must leave C_2 through one of the remaining two vertices in t, say y. Thus P_i must contain two of the three vertices in t, and $V_i(x,y)$ contains at least one vertex other than z. Further, all vertices in $V_i(x,y)$ lie in C_2, and none of the vertices in $V_i[x,y]$ lie in C_2. Thus the vertices in $V_i(x,y)$ are separated from the vertices in $V_i[x,y]$ by t.

To prove the second claim of the lemma for this case, suppose P_j is an ear that contains x and y and also a vertex, say u, in C_2. Then $j>i$, since P_i is the lowest-numbered ear to contain a vertex in C_2. Since P_i contains x and y, x and y must be the endpoints of P_j, and all other vertices on P_j lie in C_2. Further, since $i<j$ and vertex v is contained in P_i, the vertices in the bridge

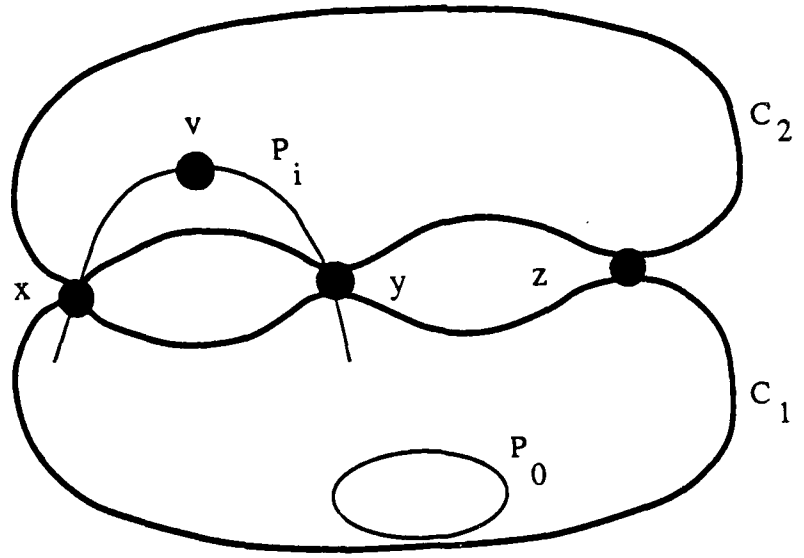


Figure 6.1.
Case 1 in the proof of Lemma 6.1.

of P_j containing v (call it B') are in $V_j[x,y]$, and since C_2 is a connected component in the subgraph induced by $V-\{x,y,z\}$, there is a path from B' to the vertex u in $V_j(x,y)$ that avoids x, y, and z. This establishes the second claim of the lemma for this case.

Case 2 P_0 contains a vertex in C_2:

If P_0 contains no vertex in C_1, then case 1 applies to C_1. Otherwise P_0 contains at least one vertex in C_1 and one vertex in C_2. But then, since P_0 is a simple cycle, it must contain two of the three vertices in t, say x and y, such that (by the argument of case 1), every path from a vertex in $V_0(x,y)$ to a vertex in $V_0[x,y]$ contains x, y or z, and P_0 is the unique ear with this property, which has a vertex in C_2.

□ Lemma 6.1.

We will say that a separating triplet $t=(x,y,z)$ *separates* ear P_i if P_i contains two of the vertices in t, say x and y, with $V_i(x,y)$ not a subset of $\{z\}$, and the vertices in $V_i(x,y)$ are disconnected from the vertices in $V_i[x,y]$ in the subgraph of G induced by $V-\{x,y,z\}$. We will denote this by writing t as $i([x,y],z)$ to indicate that P_i contains x and y, and $V_i(x,y)$, which contains a vertex other than z, is separated from $V_i[x,y]$ by $\{x,y,z\}$. By Lemma 6.1, every separating triplet in G separates some ear, and hence can be written in the above form. We will write $i([x,y],z)$ as simply $([x,y],z)$, if the ear number is obvious from the context.

Analogously, for a star graph $G(P)$, a triplet of vertices $t=([x,y],z)$ in G *separates* P if P contains x and y, $V(x,y)-\{z\}$ and $V[x,y]-\{z\}$ are non-empty, and the vertices in $V(x,y)$ are separated from the vertices in $V[x,y]$ when x, y and z are deleted from $G(P)$.

Lemma 6.2. Let $G=(V,E)$ be a triconnected graph with an open ear decomposition $D=[P_0,\cdots,P_{r-1}]$. Let $i([x,y],z)$ separate P_i. If P_i does not contain z then

i) z is an articulation point in one of the bridges of P_i, and

ii) if P_j is the largest-numbered ear that contains z, then $j>i$.

Proof: Let B be the bridge of P_i containing z. Then B has an attachment in both $V_i(x,y)$ and

$V_i[x,y]$, since otherwise, x,y would be a separating pair. Let a be an attachment of B in $V_i(x,y)$ and let b be an attachment of B in $V_i[x,y]$. Suppose there is a path p between a and b in B that avoids z. Then, if x,y, and z are removed from G, the vertices of $V_i(x,y)$ will remain connected to the vertices of $V_i[x,y]$ by the path p. But this is not possible since $([x,y],z)$ separates P_i. Hence, every path between a and b in B must pass through z, i.e., z is an a.p. of B.

Let C be the connected component containing $V_i(x,y)$ in $G-\{x,y,z\}$. To prove the second claim of the lemma, we note that, by Lemma 6.1, P_i is the lowest numbered ear containing a vertex in C. Hence every edge (w,z) with w in C must belong to an ear numbered greater than i. By the first part of this proof, we know that there is at least one such edge (w,z). This proves the second part of the lemma.

□ Lemma 6.2.

Using Lemma 6.2, we can classify triplets separating ear P_i into two types: Type 1 separating triplets are those for which P_i contains all three vertices; type 2 separating triplets are those for which P_i contains two vertices, and the third is an articulation point in one of the bridges of P_i. Let n_i be the number of vertices contained in a nontrivial P_i. Let us number the vertices on P_i from 1 to n_i. Vertex a is to the *left* of b and vertex b is to the *right* of a if $a<b$. Type 1 separating triplets can be further classified into three types (see Figure 6.2): Type 1a, in which z is to the right of x and y on P_i, type 1b, in which z is to the left of x and y, and type 1c, in which z is between x and y on P_i. The same definitions apply to a star graph $G(P)$; in this case the center of every star is its unique a.p.

Let $([x,y],z)$ be a type 2 triplet separating P_i. By Lemma 6.2, z is an a.p. in a bridge, B, of P_i, and z lies on an ear $P_j, j>i$. We shall refer to such a.p.'s as *high a.p.'s*. Let $B_1, \cdots, B_k$ be the connected components of $B - \{z\}$, and let C be the set of remaining bridges of P_i. Then $C \cup (\bigcup_{i=1}^{k} \{B_i\})$ are the bridges of P_i in $G - \{z\}$. Let $J_i(z)$ be the ear graph of P_i in $G - \{z\}$.

Lemma 6.3. Let G be a triconnected graph, and let $G_i(P_i)$ be the ear graph of P_i. Then,

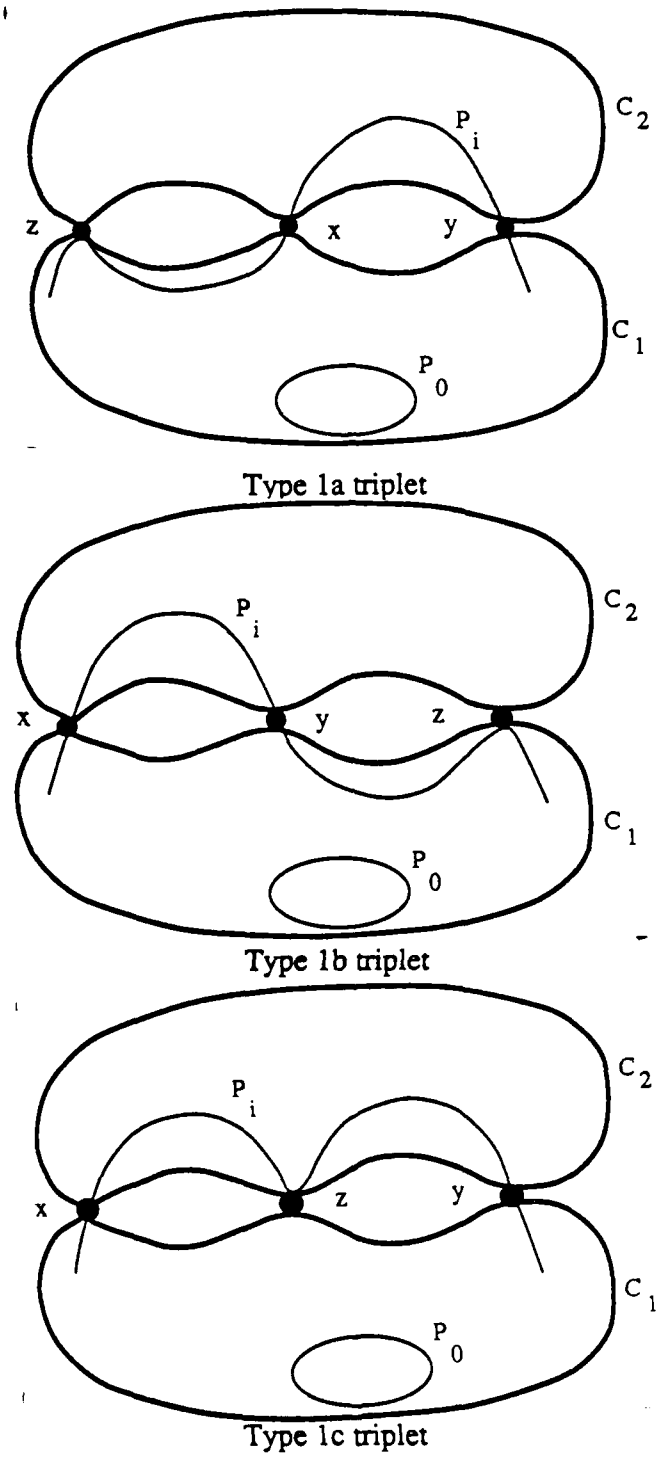


Figure 6.2.
Classification of type 1 separating triplets

a) $([x,y],z)$ is a type 1 triplet separating P_i in G if and only if it is a type 1 triplet separating P_i in

G_i.

b) $([x,y],z)$ is a type 2 triplet separating P_i in G if and only if (x,y) is a pair separating P_i in $J_i(z)$.

Proof: We note that, since G is triconnected, every anchor bridge of P_i in G has attachments to the two endpoints of P_i, and to at least one internal vertex of P_i; we shall call this *Fact 1*. We prove parts a) and b) of the lemma separately.

a) First we note that if $([x,y],z)$ is a type 1 triplet separating P_i in the ear graph G_i then it certainly separates P_i in G.

For the reverse, two cases arise:

i) If x and y are the endpoints of P_i, then by Fact 1, $([x,y],z)$ is a type 1 triplet separating P_i if and only if every anchor bridge of P_i has exactly one internal attachment on P_i, and that attachment is at z. If this holds in G then it continues to hold in the ear graph G_i, since by coalescing such anchor bridges, we do not create any new attachments.

ii) If either x or y is not an endpoint of P_i, then no anchor bridge of P_i can have an attachment in $V_i(x,y)-\{z\}$. Once again, this condition will continue to hold if all anchor bridges are coalesced, and hence will be true in G_i if it was true in G.

b) As in case a), if (x,y) is a pair separating P_i in $J_i(z)$ then clearly $([x,y],z)$ is a type 2 triplet separating P_i in G. For the reverse, once again, two cases arise.

i) z is a high a.p. in an anchor bridge B. Hence z lies on an ear numbered higher than i. Every attachment of an anchor bridge on a vertex in $V_i(x,y)$ must be adjacent to z (since otherwise $([x,y],z)$ would not separate P_i from vertices on lower-numbered ears). Hence in $G-\{z\}$, no anchor bridge has an attachment in $V_i(x,y)$, and so, in the ear graph G_i, the anchoring star has not attachment in $V_i(x,y)$. The non-anchor bridges cannot have an attachment in both $V_i(x,y)$ and $V_i[x,y]$, since each of them is a bridge of G and $([x,y],z)$ is a triplet separating P_i in G. Thus x,y must separate P_i in $J_i(z)$.

If one of x or y is not an endpoint of P_i, then every anchor bridge B' other than B has no attachment in $V_i(x,y)$. This continues to hold in G_i as well.

ii) z is an a.p. in a non-anchor bridge: In this case no anchor bridge of G can have an attachment in $V_i(x,y)$, and the result follows by an argument as in case i.

□ Lemma 6.3.

Based on the characterization in Lemmas 6.1,6.2 and 6.3, we obtain the following high-level algorithm to find all separating triplets in a triconnected graph.

Algorithm 6.1: Finding All Separating Triplets in a Triconnected Graph $G=(V,E)$

1) Find an open ear decomposition $D=[P_0, \cdots, P_{r-1}]$ for G.

2) *For* $i=r-1,r-2, \cdots ,0$ *do*

if P_i is a nontrivial ear *then*

A) Construct the ear graph $G_i(P_i)$.

B) Use $G_i(P_i)$ to find all type 1 triplets separating P_i.

C) In the bridges of P_i, find the a.p.'s that lie on ears numbered higher than i, and use them to find all type 2 triplets separating P_i.

Let $|V|=n$ and $|E|=m$. Step 1 has a linear-time sequential algorithm and an $O(\log n)$ time parallel algorithm with $O(m)$ processors on a CRCW PRAM [33, 35]. Step 2A has a linear-time sequential algorithm and an $O(\log n)$-time parallel algorithm with $O(m \log n)$ processors on an ARBITRARY CRCW PRAM [36, 41].

Let n_i be the number of vertices contained in a nontrivial P_i, and let m_i be the number of edges incident on vertices contained in P_i. Since G is triconnected, it is not difficult to see that $\sum_i m_i = O(m + n^2)$ and $\sum_i n_i = O(n)$ when the summation is over nontrivial ears. In section 6.2.1, we present algorithms to find type 1 triplets separating P_i in $O(n_i^2+m_i)$ sequential time, and in

$O(\log n)$ parallel time with n^2 processors on an EREW PRAM. In section 6.2.2, we show how to find all high a.p.'s in the modified bridges of each ear, organized in a forest of block-trees, in $\sum_i O(n+m_i)=O(n^2)$ time plus some additional time for processing trivial ears, which is $O(m)$ over the execution of the entire algorithm. This parallelizes into an $O(\log^2 n)$ time algorithm to find a.p.'s in bridges of all nontrivial ears on an ARBITRARY CRCW PRAM with n^2 processors. We use this to develop an algorithm to find all type 2 triplets in $\sum_i O(n \cdot n_i+m_i)=O(n^2)$ sequential time, and in $O(\log^2 n)$ parallel time using n^2 processors on an ARBITRARY CRCW PRAM. Thus we obtain an $O(n^2)$ time sequential implementation of Algorithm 6.1, as well as an $O(\log^2 n)$ time parallel implementation on an ARBITRARY CRCW PRAM with n^2 processors.

6.2. Finding All Triplets that Separate an Ear

6.2.1. Finding Type 1 Separating Triplets

In this section we give algorithms to find type 1a, 1b and 1c separating triplets on an ear P_i. Recall that $([x,y],z)$ is a type 1 triplet separating P_i, if x,y and z lie on P_i, and the vertices in $V_i(x,y)$ are separated from the vertices in $V_i[x,y]$ when x,y and z are removed from P_i.

As shown in Lemma 6.3, if x and y are the endpoints of ear P_i then $([x,y],z)$ form a type 1 triplet separating P_i if and only if the anchoring star in $G_i(P_i)$ has exactly one internal attachment on P_i, and that attachment is z. This is a simple condition that can be checked in constant time. For finding any other type 1 triplet separating P_i, it suffices to view the ear graph $G_i(P_i)$ as the path P_i together with a collection of stars, and to identify all type 1 triplets separating P_i in G_i. For this we can work with a star graph $G(P)$ without any reference to the fact that it is the ear graph of an ear.

Let $G(P)$ be a star graph with k vertices on P, l stars, and a total of p edges on the stars. We present an $O(k^2+p)$ time sequential algorithm and an $O(\log k)$ time parallel algorithm with k^2+p processors on an EREW PRAM to find all type 1 triplets separating P in $G(P)$. Assume that the vertices on P are numbered in order as $1, \cdots, k$ from left to right as specified before.

For a closed interval $[x,y]$ on P, let

$L[x,y]$ be the leftmost attachment among all stars that have an attachment in $[x,y]$,

$S[x,y]$ be the second leftmost attachment among all stars that have an attachment in [x,y], and

$R[x,y]$ and $M[x,y]$ be the rightmost and second rightmost attachments, respectively, of stars that have an attachment in $[x,y]$.

The following lemma is straightforward.

Lemma 6.4. Let x,y,z be three vertices on P. Then

a) $([x,y],z)$ is a type 1a triplet separating P if and only if $L[x+1,y-1]=z, S[x+1,y-1]\geq x$ and $R[x+1,y-1]\leq y$; and

b) $([x,y],z)$ is a type 1b triplet separating P if and only if $R[x+1,y-1]=z, M[x+1,y-1]\leq y$ and $L[x+1,y-1]\geq x$.

□ Lemma 6.4.

We compute $L[x,y], S[x,y], R[x,y]$ and $M[x,y]$ for every interval $[x,y]$ with $x\leq y$ by a doubling technique that first computes these values incrementally for intervals whose size is a power of 2, and then computes the values for all remaining intervals as follows. This algorithm runs in $O(k^2+p)$ time sequentially, and in $O(\log k)$ time on an EREW PRAM with n^2 processors.

1) *Initialize:* For $i=1, \cdots, k$ compute $L[i,i]$, $S[i,i]$, $R[i,i]$, and $M[i,i]$. These values can be computed in $O(k+p)$ sequential time and $O(\log n)$ parallel time on an EREW PRAM with $k+p$ processors by using bucket sort to order the star edges in increasing order of attachment, with ties broken in decreasing order of the leftmost (rightmost) attachment of the star the edge belongs to for $L[i,i]$ and $S[i,i]$ (for $R[i,i]$ and $M[i,i]$).

2) For $j=1,\cdots,\lceil \log k \rceil$ compute, for each i, $L[i,i+2^j-1]$ from $L[i,i+2^{j-1}-1]$ and $L[i+2^{j-1},i+2^j-1]$. Similarly compute $S[i,i+2^j],R[i,i+2^j]$ and $M[i,i+2^j]$. Each of these values can be computed in constant time in parallel, and hence sequentially as well. Thus, this total step takes $O(k\log k)$ time sequentially and $O(\log k)$ parallel time on an EREW PRAM with k processors.

3) For each pair $[x,y],x<y$, let i_{xy} be the integer satisfying $x+2^{i_{xy}}\le y<x+2^{i_{xy}+1}$. Compute $L[x,y]$ from the pre-computed values $L[x,x+2^{i_{xy}}-1]$ and $L[y-2^{i_{xy}}+1,y]$ in constant time. Similarly compute $S[x,y],R[x,y]$ and $M[x,y]$. As in step 2, each of these values can be computed in constant time, and hence this step requires $O(n^2)$ sequential time; it is straightforward to implement this in $O(\log n)$ parallel time on an EREW PRAM with n^2 processors.

For complete parallel algorithm for type 1a separating triplets separating ear P_i see appendix 6.3.

An analogous procedure identifies type 1c separating triplets. Let $L[x,y]$ and $R[x,y]$ be as before. Let z_l be a vertex in $[x,y]$ which is an attachment of a star with an attachment at $L[x,y]$; analogously let z_r be a vertex in $[x,y]$ which is an attachment of a star with an attachment at $R[x,y]$. Let $S'[x,y]$ be the leftmost attachment of stars with an attachment in $[x,y]$ - $\{z_l\}$ and let $M'[x,y]$ be the rightmost attachment among stars with an attachment in $[x,y]$ - $\{z_r\}$. Then the following lemma is again straightforward.

Lemma 6.5. the triplet $([x,y],z)$ is a type 1c triplet separating P if and only if $S'[x+1,y-1]\ge x, M'[x+1,y-1]\le y$ and one of the following three conditions hold:

a) $z_l=z_r=z$; or

b) $L[x+1,y-1]\ge x$ and $z=z_r$; or

c) $R[x+1,y-1]\le y$ and $z=z_l$.

□ Lemma 6.5.

Using Lemma 6.5 we can compute the type 1c triplets separating P in a manner analogous to the method used for finding type 1a and 1b triplets separating P. For complete parallel algorithm for type 1c separating triplets separating ear P_i see appendix 6.3.

6.2.2. Finding Type 2 Separating Triplets

There are many implementation details in this algorithm. We give a high-level description first, and then elaborate on each of the steps. We use the result in Lemma 6.2 that if $([x,y],z)$ is a type 2 triplet separating P_i, then z is a high a.p., i.e., z is an a.p. in one of the bridges of P_i, and z belongs to a higher-numbered ear than P_i. Observe that the number of blocks (biconnected components) and the number of articulation points in the bridges of an ear P_i is no more than n. As a matter of notation, we will denote the star(s) in G_i corresponding to a bridge or a collection of bridges B of P_i by $s(B)$, and similarly, the bridge(s) of P_i corresponding to a star or a collection of stars S of G_i by $b(S)$. We now present the high-level algorithm for finding type 2 triplets separating P_i. For convenience we assume that the vertices of G are numbered so that any vertex contained in P_i has a smaller number than a vertex in the interior of any $P_j, j>i$.

1) For each star s of G_i, we construct a list $L(s)$ of those pairs of vertices x,y on P_i for which s is the only star that has attachments in both $V_i(x,y)$ and $V_i[x,y]$. Note that there can be no more than n_i^2 entries in the lists for all of the stars of G_i, since each pair can appear on at most one list. The list for each star is in lexicographically increasing order on (x,y).

2) For each ear P_i, we determine the high a.p.'s in each of its bridges.

3) For each bridge B of P_i, for each high a.p. a in B, we find all pairs of vertices separating P_i in $P_i \cup (B - \{a\})$ (note that we do *not* include the remaining bridges of P_i in this graph), using the triconnectivity algorithm in [36, 41]. These separating pairs can be specified as regions in the *star embedding of the coalesced graph of* $P_i \cup s(B-\{a\})$ *[36]*. We maintain these regions over all articulation points for a given bridge B in a properly sorted manner;

we call this the *region representation for B*.

4) We compare the entries in $L(s)$ for each s with pairs of vertices in a common region in the region representation for $b(s)$, and each match gives a type 2 separating triplet for P_i.

We need the following observation.

Observation 6.1. Let z be a high a.p. of a bridge B of P_i, and x,y, a pair of vertices on P_i. Then $([x,y],z)$ is a type 2 triplet separating P_i if and only if

a) (x,y) is a pair separating P_i in the graph $P_i \cup (B-\{z\})$, and

b) $s(B)$ is the only star of G_i that has an attachment in both $V_i(x,y)$ and in $V_i[x,y]$.

Proof: If $([x,y],z)$ is a type 2 triplet separating P_i, then by Lemma 6.2 we know that z must be an a.p. of the bridge B of P_i to which it belongs. Deleting z renders x,y a pair separating P_i in G - $\{z\}$, and thus certainly in the graph $P_i \cup (B-\{z\})$. Further if any other bridge B' of P_i has an attachment in both $V_i(x,y)$ and $V_i[x,y]$, then removal of x,y, and z leaves $V_i(x,y)$ connected with $V_i[x,y]$, which is not possible since $([x,y],z)$ separates P_i by assumption. Hence part b) of the observation must hold as well.

For the converse, assume that parts a) and b) hold. Then it follows that x,y is a pair separating P_i in G - $\{z\}$, since by b), no bridge other than B can connect $V_i(x,y)$ with $V_i[x,y]$ in G - $\{x,y,z\}$. Hence $([x,y],z)$ must be a type 2 triplet separating P_i.

□ Observation 6.1.

All pairs of vertices on P_i satisfying property b) appear on the list $L(s(B))$, which we construct in step 1. The pairs satisfying property a) are those that lie in a common region in the region representation for B, which we construct in step 2. In step 3 we scan these two sets of pairs of vertices, and identify matches between the two sets; each such match gives a type 2 triplet separating P_i, and every type 2 triplet separating P_i appears as such a match. This establishes the correctness of the above algorithm.

We now explain how to implement steps 1 through 4 to obtain the stated time and processor bounds.

STEP 1

The algorithm for step 1 is similar to the algorithm for finding type 1 separating triplets. By Lemma 6.3, if x and y are the endpoints of ear P_i, then the anchoring star of G_i is the unique star with attachments in both $V_i(x,y)$ and $V_i[x,y]$. For any other pair x,y we can work with a star graph $G(P)$ without any reference to the fact that it is the ear graph of an ear.

As in section 6.2.1, given a star graph $G(P)$ we compute certain values for each interval of vertices on P. The values computed are $L[x,y]$,$S''[x,y]$,$R[x,y]$ and $M''[x,y]$, where $L[x,y]$ and $R[x,y]$ are, as before, the leftmost and rightmost attachments, respectively, among all stars that have an attachment in the closed interval $[x,y]$. Let s_l be a star with attachments at $L[x,y]$ and in $[x,y]$, and similarly, let s_r be a star with attachments at $R[x,y]$ and in $[x,y]$. $S''[x,y]$ is the leftmost attachment among all stars with an attachment in $[x,y]$ except star s_l; similarly, $M''[x,y]$ is the rightmost attachment among all stars with an attachment in $[x,y]$ except s_r. From these definitions, the following lemma is straightforward.

Lemma 6.6. Star s is the only star that has an attachment in $V(x,y)$ and $V[x,y]$ if and only if one of the following three hold:

a) $S''[x,y] \geq x, R[x,y] \leq y$ and $s = s_l$; or

b) $L[x,y] \geq x, M''[x,y] \leq y$ and $s = s_r$; or

c) $S''[x,y] \geq x, M''[x,y] \leq y$ and $s = s_l = s_r$.

□ Lemma 6.6.

Using Lemma 6.6 and the method of section 6.1, we can form the lists $L(s)$ for all stars of all nontrivial ears in $O(n^2)$ sequential time and in $O(\log n)$ parallel time on an EREW PRAM with n^2 processors.

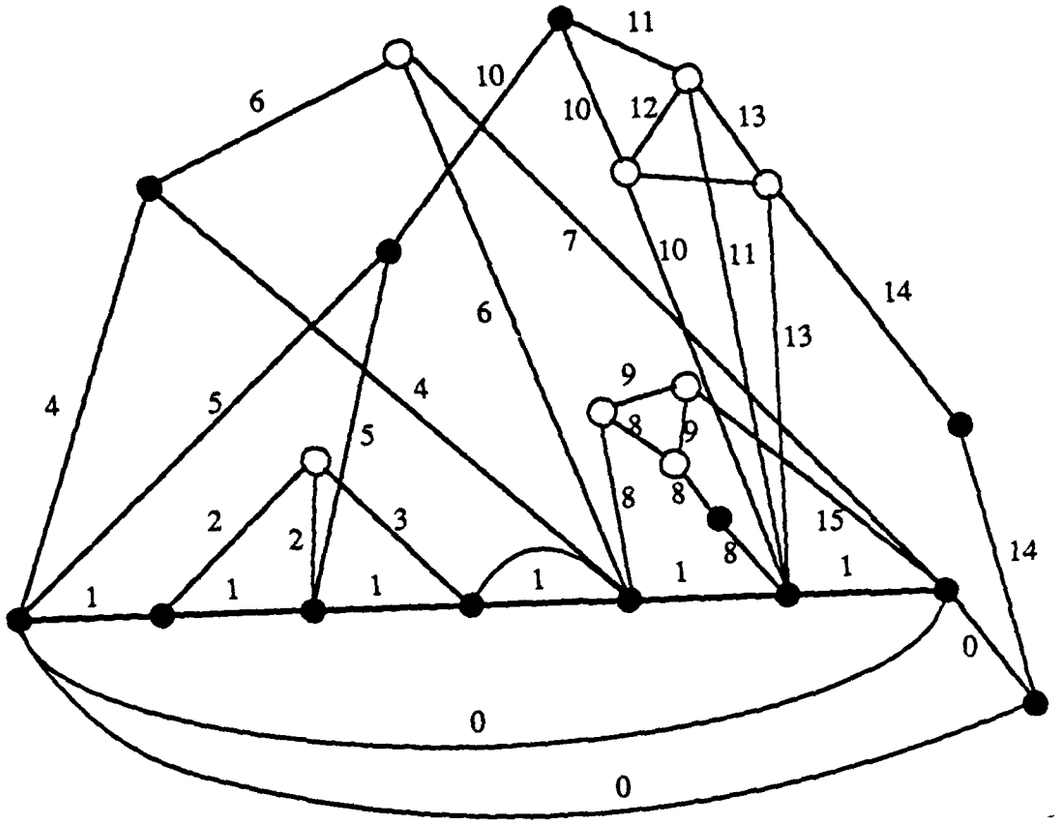
STEP 2

Sequential Algorithm

Let $H_i=\bigcup_{j=0}^{i} P_j$. Let $A_1, \cdots, A_k$ be the bridges of H_i. Let B_j be A_j with its attachment edges and vertices deleted. A *split* of P_i is an articulation point in one of the B_j. An *ex-node* of P_i is a vertex in one of the B_j adjacent to an attachment on H_i. An *adj-node* of P_i is an ex-node, which is adjacent to an vertex on P_i. For example in Figure 6.3, H_1 has four nontrivial bridges and one trivial bridge; vertices a, b, and c are some of the split nodes of P_1; vertices a, b, d, and e are some of the ex-nodes of P_1; of which a, b, and d are adj-nodes as well. We observe that by Lemma 6.2, if $([x,y],z)$ is a type 2 triplet separating P_i then z is a split or ex-node of P_i or z is an attachment of one of the A_j on H_{i-1}.

We organize the splits and ex-nodes of P_i in a forest of *split-trees* analogous to the tree of biconnected components. There is one split-tree for each B_j, whose vertices are the splits, ex-nodes and blocks of B_j. There is an edge between a split and each block it lies in, as well as an edge between each ex-node (that is not also a split) and the unique block it lies in (see Figure 6.3). For u an ex-node, let $A(u,j)$ be the jth smallest vertex adjacent to u and belonging to H_{i-1}, if it exists, *null* otherwise, for j=1,2,3,4. By our numbering scheme for vertices, $A(u,j), j=1, \cdots, 4$ (when defined) represent four distinct vertices on lowest numbered ears adjacent to u. For example, in Figure 6.3 vertex a has $A(a, 1) = 1$, $A(a, 2) = 5$, $A(a, 3) = \varnothing$, $A(a, 4) = \varnothing$. The number of entries in $A(u,j)$, over all ex-nodes u, is $O(n)$.

Let F_{i-1} be H_{i-1} with the two endpoints of P_i deleted. Let $A(u)$ be the set of two smallest non-null vertices in $F_{i-1} \cap \{A(u, 1), A(u, 2), A(u, 3), A(u, 4)\}$. By construction, $A(u)$ contains the two smallest numbered vertices in F_{i-1} adjacent to u (when they exist), and can be obtained in constant time per ex-node, since we have the $A(u,j)$. Note that if we did *not* have the $A(u,j)$, then finding the $A(u)$ would take time proportional to the number of edges incident on the ex-nodes and that could be as large as $\theta(m)$.



Graph $G = (V,E)$ with open ear decomposition indicated by ear number along edges.

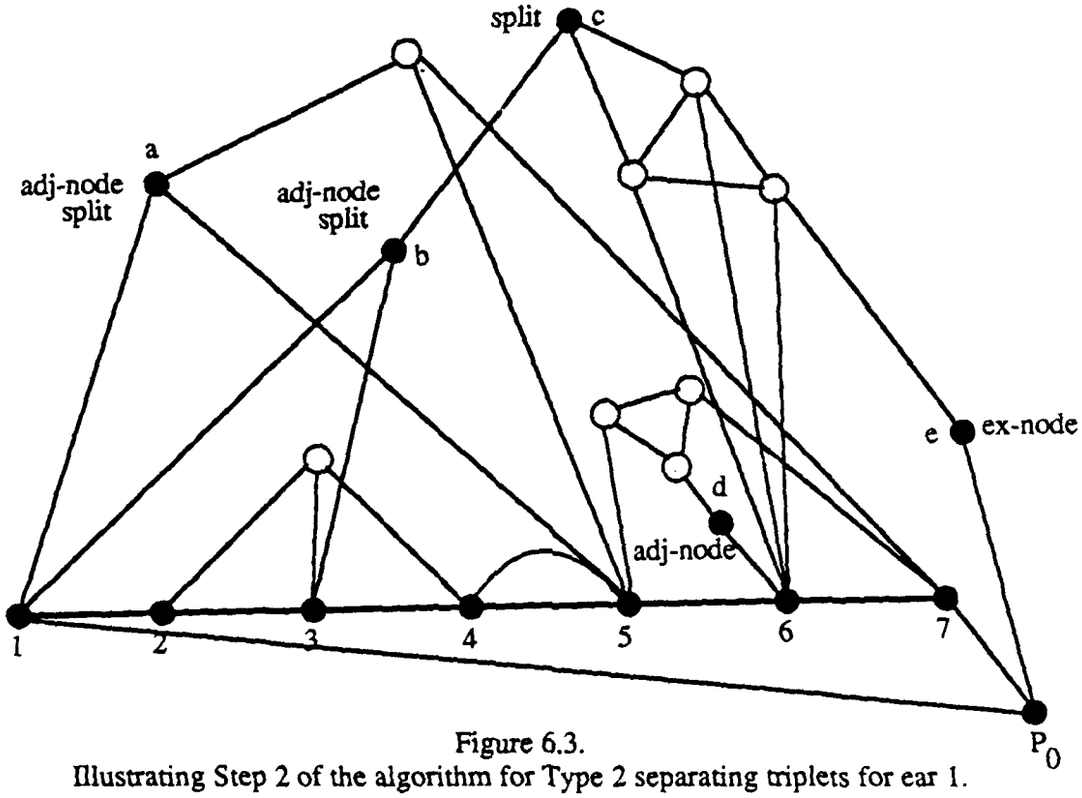


Figure 6.3.
Illustrating Step 2 of the algorithm for Type 2 separating triplets for ear 1.
Vertices on ear 1 are labeled from 1 to 7.

From the forest of split-trees we derive the forest of trees of biconnected components (or *block-trees*) of the bridges of P_i by first constructing the *augmented graph* as follows: We augment the vertex set of the forest of split-trees by adding in vertex v to represent H_{i-1}, — a potential 'high-block' (i.e., a connected component that contains no high a.p.'s), and we add in the set of vertices $U = \bigcup_{\text{ex-nodes } u} A(u)$, — potential high a.p.'s. We put in an edge between v and each vertex in U as well as edges between u and vertices in $A(u)$, for each ex-node u.

Observe that a vertex w in H_{i-1} is a high a.p. in a bridge of P_i if and only if, for some split-tree T of P_i, w is the only vertex in F_{i-1} that is adjacent to a vertex in T. Since by construction $A(u)$ includes the two smallest vertices adjacent to u, if they lie in F_{i-1}, it follows that w is a high a.p. in a bridge of P_i if and only if it is an a.p. in the augmented graph. Similarly, an ex-node u in a split-tree T is an a.p. separating vertices in T from the rest of the bridge of P_i if and only if u has an attachment in F_{i-1} and no other ex-node in T has an attachment in F_{i-1}. This again holds if and only if u is an a.p. in the augmented graph. Hence the blocks and articulation points in this augmented graph are precisely the blocks and articulation points in the bridges of P_i. We find these in $O(n)$ sequential time, using a linear-time algorithm for biconnectivity [48]. At this point we have the forest of block-trees for the bridges of P_i. In additional $O(m_i)$ time, we can obtain all of the adj-nodes by scanning all edges incident on the internal vertices of P_i.

All that remains is to incrementally obtain the split-trees for P_l and the $A(u,j)$ for the new ex-nodes of P_l in an efficient way, where P_l is the next nontrivial ear. To update information for P_l, we first process the forest of split-trees for P_i to eliminate those splits and blocks that disappear and the new ones that appear when $P_i, P_{i-1}, \cdots, P_{l+1}$ are added. This is done in $O(n+m_i+l-i)$ time by finding blocks, a.p.'s and ex-nodes in the graph $\bigcup_j B_j \bigcup_{k=l+1}^{i} P_k \bigcup$ {attachment edges of each B_j in the interior of P_i}. This gives us the split-trees for P_l. The new ex-nodes for P_l are the nodes in the interior of P_i adjacent to a vertex in H_l; in particular, this

includes the nodes in the interior of P_i adjacent to its endpoints. We compute the $A(u,j)$ values for these new ex-nodes. This computation takes $O(m)$ time over the entire execution of the algorithm. Now we are ready to find type 2 triplets separating P_l.

Parallel Implementation of Step 2:

This step is similar to the algorithms in [36, 41] that find the ear graphs of all nontrivial ears. The only difference is that we now find the forest of block-trees instead of connected components. For this we can use any efficient parallel block finding algorithm [33, 35, 50]. By noting that the total size of the graphs present at each stage of the algorithm is $O(n^2)$, we obtain an $O(\log^2 n)$ time parallel algorithm on an ARBITRARY CRCW PRAM with n^2 processors.

STEP 3

Sequential Algorithm

We number the vertices in the forest of block-trees in post-order with respect to a dfs. We label each attachment edge to P_i in the bridges of P_i by the number of the block it belongs to (since each such edge is incident on an adj-node, this is done in constant time per edge). We remove any multiple occurrences of edges with the same block number and attachment. Since the number of blocks and the number of articulation points is $O(n)$ (over all bridges of P_i) this step can be done in $O(n+m_i)$ time for all of the bridges.

We now sort (using bucket sort) the labeled attachment edges in increasing order of the attachments, with edges having the same attachment sorted in increasing order of their label, and we leave the sorted edges in stacks corresponding to their attachment number. Now, with another post-order traversal of the block-trees, we can determine, for each a.p. s of each bridge B of ear P_i, the stars formed from B when s is deleted from B, in $O(n+m_i)$ time.

At this point, for each high a.p. x of bridge B, we have $s(B-\{x\})$, the collection of stars formed from B when x is removed from B. Each of these stars has no more than n_i attachments.

Using the algorithm in [36] we can find the separating pairs on P_i corresponding to these stars in $O(k \cdot n_i)$ time, where k is the number of stars. These are organized as the vertices on the faces of the planar embedding of the *coalesced graph [36];* this has an $O(k \cdot n_i)$ size representation. We find such a representation for each a.p.

Parallel Implementation of Step 3:

This step can be implemented on ear P_i in $O(\log n)$ time with $O(n \cdot n_i)$ processors using efficient parallel algorithms for computing post-order numbering on trees [50], for sorting [9] and for finding separating pairs in a star graph [41].

STEP 4

Sequential Algorithm:

In order to execute step 4 efficiently, we store the representation obtained in step 3 in a special way. Let us confine our attention to a specific bridge B (note that we execute step 3 bridge by bridge). Let B have l a.p.'s $s_1, \cdots, s_l$, and let the ith a.p. have t_i children in the block-tree (each of which is a block of B). Thus $t=1+\sum_{j=1}^{l} t_j$ is the number of blocks in B. The total number of regions for B is no more than t, and further, none of these regions interlace. We represent these regions as follows: We have n_i stacks, one for each vertex on P_i, and in the stack for vertex v, we place pointers to all regions that contain v. These pointers are arranged in increasing order of the first vertex in the region, with ties broken in decreasing order of the last vertex in the region (the topmost pointer points to the region with the lowest numbered vertex). For each region we maintain a pointer to the current leftmost position in the region; initially the pointer for each region points to its leftmost vertex.

We now scan the entries in $L(s(B))$ in order. If the current entry is (x,y), we look at the topmost region R in the stack for vertex y and check its current leftmost vertex z. If $z>x$ then we proceed to the next entry in $L(s(B))$. If $z=x$ then we have found a match, and hence a type 2

triplet separating P_i. If $z<x$, we move the leftmost pointer for R right until it points to a vertex $u \geq x$. If $u=x$, then we have located a type 2 triplet and we leave the leftmost pointer at u. If $u=y$ then we pop the pointer to R off the stack; otherwise we leave the leftmost pointer at u. It is easy to see that this scan locates all type 2 triplets $([x,y],z)$ with z in B, and the time it takes is proportional to the sizes of $L(s(B))$ and the regions representation for B. Hence, over all bridges of P_i this procedure takes time $O(n_i^2+n \cdot n_i)=O(n \cdot n_i)$.

Parallel Implementation of Step 4:

To implement step 4 in parallel we allow ourselves $O(\log n)$ time per entry (x,y) on $L(s(B))$ to determine if x lies in the same region as y for some entry in stack y; this is accomplished by binary search on the entries in stack y followed by a binary search on the vertices in the relevant region R. Hence this step takes $O(\log n)$ time with n^2 processors on a CREW PRAM.

6.3. Appendix

Let $L(x,y)=L[x,y]$, which is defined in section 6.2.1. Analogously, $R(x,y)$ stands for $R[x,y]$, $S(x,y)$ stands for $S[x,y]$, and $M(x,y)$ stands for $M[x,y]$. Let $L(x)$ stands for the left most attachments among all stars that have an attachment at x. Analogously, $S(x)$, $R(x)$, and $M(x)$ denote the second leftmost, rightmost, and the second rightmost attachments among all stars that have an attachment at x, respectively.

6.3.1. Algorithm for Type 1a Separating Triplets for an Ear

STEP 1: **For** every vertex v on ear P_i **do**

$L(v,v+2) = min(v, L(v+1))$

$R(v,v+2) = max(v+2, R(v+1))$

$S(v,v+2) = min(S(v+1),v)$

For k=1 to $\log n_i$ **do**

find $L(v,v+2^k)$, $R(v,v+2^k)$, and $S(v,v+2^k)$

$(L(v,v+2^k)$= min $(L(v,v+2^{k-1}),L(v+2^{k-1},v+2^k),L(v+2^{k-1})))$

$(R(v,v+2^k)$= max $(R(v,v+2^{k-1}),R(v+2^{k-1},v+2^k),R(v+2^{k-1})))$

$(S(v,v+2^k)$= *second smallest*

$(L(v,v+2^{k-1}),L(v+2^{k-1},v+2^k),S(v,v+2^{k-1}),S(v+2^{k-1},v+2^k),$

$L(v+2^{k-1}),S(v+2^{k-1})))$

If $R(v,v+2^k)=v+2^k$ and $S(v,v+2^k)=v$ **then**

(x,y,z) is a separating triplet for ear P_i, where $x=v$, $y=v+2^k$ and $z=L(v,v+2^k)$.

End If

End For

STEP 2: **For** every other interval (v,w) of ear P_i **do**

find k_1 such that $v+2^{k_1+1}>w$ and $v+2^{k_1}<w$,

find k_2 such that $w-2^{k_2+1}<v$ and $w-2^{k_2}>v$,

$(k_1=k_2=\lfloor \log(w-v) \rfloor)$

If $R(w-2^{k_2},w)=v+2^k$ and $R(v,v+2^{k_1})\le v+2^k$ and $S(v,v+2^{k_1})=v$ and $S(w-2^{k_2},w)\ge v$

and $(S(v,v+2^{k_1})\ge v$ or $S(v,v+2^{k_1})=S(w-2^{k_2},w)$

then $(v,w,S(w-2^{k_2},w))$ is a separating triplet for ear P_i,

otherwise no.

End If

End For

6.3.2. Algorithm for Type 1c Separating Triplets for an Ear

STEP 1: **For** every vertex v on ear P_i **do**

If $R(v+1) \geq R(v+2)$ **then** $R(v,v+3) = \max(R(v+1),v+3)$

$z_r(v,v+3) = v+1$

else $R(v,v+3) = \max(R(v+2),v+3)$

$z_r(v,v+3) = v+2$

End If

If $L(v+1) \leq L(v+2)$ **then** $L(v,v+3) = \min(L(v+1),v)$

$z_l(v,v+3) = v+1$

else $L(v,v+3) = \min(R(v+2),v)$

$z_l(v,v+3) = v+2$

End If

$M(v,v+3) = \max(v+3,\min(R(v+1),R(v+2))$

$S(v,v+3) = \min(v,max(L(v+1),L(v+2))$

If

$(S(v,v+3) = v \;\&\; M(v,v+3) = v+3 \;\&\; (R(v,v+3) = v+3 \mid L(v,v+3) = v \mid z_l(v,v+3) = z_r(v,v+3)))$

then

If $L(v,v+3) = v$ **then** $z = z_r(v,v+3)$

else $z = z_l(v,v+3)$

[(v,v+3),z] is a separating triplet

End If

End If

For $(v,v+4)$ use STEP 2.

For k=3 to $\log n_i$ **do**

find $L(v,v+2^k)$, $S(v,v+2^k)$, $M(v,v+2^k)$, $M(v,v+2^k)$, $z_l(v,v+2^k)$, and $z_r(v,v+2^k)$

If $L(v,v+2^{k-1}) \le L(v+2^{k-1},v+2^k)$ **then** $L(v,v+2^k) = L(v,v+2^{k-1})$

$$z_l(v,v+2^k) = z_l(v,v+2^{k-1})$$

else $L(v,v+2^k) = L(v+2^{k-1},v+2^k)$

$$z_l(v,v+2^k) = z_l(v+2^{k-1},v+2^k)$$

End If

If $R(v,v+2^{k-1}) \ge R(v+2^{k-1},v+2^k)$ **then** $R(v,v+2^k) = R(v,v+2^{k-1})$

$$z_r(v,v+2^k) = z_r(v,v+2^{k-1})$$

else $R(v,v+2^k) = R(v+2^{k-1},v+2^k)$

$$z_r(v,v+2^k) = z_r(v+2^{k-1},v+2^k)$$

End If

$S(v,v+2^k)$ = second

smallest ($S(v,v+2^{k-1})$, $S(v+2^{k-1},v\ 2^k)$, $L(v,v+2^{k-1})$, $L(v+2^{k-1},v+2^k)$

$M(v,v+2^k)$ = second

largest ($M(v,v+2^{k-1})$, $M(v+2^{k-1},v\ 2^k)$, $R(v,v+2^{k-1})$, $R(v+2^{k-1},v+2^k)$

If $R(v+2^{k-1}) > R(v,v+2^k)$ **then** $R(v,v+2^k) = R(v+2^{k-1})$

$$z_r(v,v+2^k) = v+2^{k-1}$$

$$M(v,v+2^k) = \max(R(v,v+2^k), M(v+2^{k-1}))$$

else $M(v,v+2^k) = \max(M(v,v+2^k), R(v+2^{k-1})$

End If

If $L(v+2^{k-1}) < L(v,v+2^k)$ **then** $L(v,v+2^k) = L(v+2^{k-1})$

$$z_l(v,v+2^k) = v+2^{k-1}$$

$$S(v,v+2^k) = \min(S(v,v+2^k), S(v+2^{k-1}))$$

else $S(v,v+2^k) = \min(S(v,v+2^k), S(v+2^{k-1})$

End If

If $(S(v,v+2^k) = v\ \&\ M(v,v+2^k) = v+2^k\ \&$

$\left(R(v,v+2^k) = v+2^k \mid L(v,v+2^k) = v \mid z_l(v,v+2^k) = z_r(v,v+2^k)\right)$) **then**

If $L(v,v+2^k) = v$) **then** $z = z_r(v,v+2^k)$

else $z = z_l(v,v+2^k)$

End If

$[(v,v+2^k),z]$ is a separating triplet

End If

End For

End For

STEP 2: **for** every other interval (v,w) of ear P_i **do**

$k = \lfloor \log_2(w-v) \rfloor$

If $L(v,v+2^k) \leq L(w-2^k,w)$ **then** $L(v,w) = L(v,v+2^k)$

$z_l(v,w) = z_l(v,v+2^k)$

If $z_l(v,v+2^k) \neq z_l(w-2^k,w)$

then $S(v,w) = \min(L(w-2^k,w), S(v,v+2^k))$

else $S(v,w) = \min(S(v,v+2^k), S(w-2^k,w))$

End If

else $L(v,w) = L(w-2^k,w)$

$z_l(v,w) = z_l(w-2^k,w)$

If $z_l(v,v+2^k) \neq z_l(w-2^k,w)$

then $S(v,w) = \min(S(w-2^k,w), L(v,v+2^k))$

else $S(v,w) = \min(S(v,v+2^k), S(w-2^k,w))$

End If

End If

If $R(v,v+2^k) \geq R(w-2^k,w)$ **then** $R(v,w) = R(v,v+2^k)$

$z_r(v,w) = z_r(v,v+2^k)$

If $z_r(v,v+2^k) \neq z_r(w-2^k,w)$

then $M(v,w) = \max(R(w-2^k,w), M(v,v+2^k))$

else $M(v,w) = \max(M(v,v+2^k), M(w-2^k,w))$

End If

else $R(v,w) = R(w-2^k,w)$

$z_r(v,w) = z_r(w-2^k,w)$

If $z_r(v,v+2^k) \neq z_r(w-2^k,w)$

then $M(v,w) = \max(M(w-2^k,w), R(v,v+2^k))$

else $M(v,w) = \max(M(v,v+2^k), M(w-2^k,w))$

End If

End If

If $(S(v,w) = v \ \& \ M(v,w) = w \ \& \ (R(v,w) = w \mid L(v,w) = v \mid z_l(v,w) = z_r(v,w)))$ **then**

If $L(v,w) = v$ **then** $z = z_r(v,w)$

else $z = z_l(v,w)$

End If

[(v,w),z] is a separating triplet

CHAPTER 7

ALGORITHMS FOR FINDING ALL SEPARATING *K*-SETS OF A GRAPH

7.1. Sequential Algorithm

In this section we describe a sequential algorithm for finding all minimum size separating vertex sets in an undirected graph $G = (V,E)$. Note that the number of separating k-sets in an undirected k-connected graph is $O(2^k \frac{n^2}{k})$ [26].

First, we find the connectivity k of G using network flows [13, 18, 19]. The time complexity of this algorithm is $O(\max(k,n^{1/2})kmn^{1/2})$. Next, we take a subset of vertices X of G of size k and find all minimum size separating vertex sets (of size k) between pairs of the form (x,v), where $x \in X$ and $v \in V$. Note the following simple observation.

Observation 7.1: Let $x \in X$ and $v \in V$. Assume that we have found all k-sets separating x and v in G. Then we can add edge (x,v) to E without changing (adding or deleting) any other separating k-sets of G.

Proof: This is because for any other separating k-set Y, which does not separate x and v, x and v can not belong to two different components of the graph induced by $V-Y$.

□ Observation 7.1.

We repeat this process for every $x \in X$ and every $v \in V$. During this process we add at most kn edges to E. At the end of this process every vertex $x \in X$ is connected by an edge to every vertex $v \in V$. Now, if there is separating k-set Y in this graph, then $Y=X$. So, we check if X is a separating k-set of G. Every minimum size separating vertex set of G is obtained by this computation.

We note that for a given $x \in X$ we only need to conduct this procedure for these vertices $v \in V$ which are not adjacent to x. Hence, for our algorithm we choose X to be a set of k vertices of G of maximum degrees.

Algorithm 7.1.

1. Find the connectivity k of G. (vertices of G are numbered from $v_1, \cdots, v_n$).
2. Find k vertices with the largest degrees $(x_1, \cdots, x_k)$.
   Check if these k vertices form a separating k-set of G
   (Let $\overline{G}$ be the directed graph which we get from G by applying the Even-Tarjan reduction (see Section 2.3))
   Do $i = 1 \cdots k$
   Do $j = 1 \cdots n$
3. If $v_j \neq x_i$ and v_j is not adjacent to x_i then
4. Compute a maximum flow f in $\overline{G}$ from x_i to v_j
   If $|f| = k$ then
   (Find all k-sets separating x_i and v_j as follows:)
5. Construct the residual graph $\overline{G}_f$ of $\overline{G}$ with respect to the maximum flow f
6. Shrink the strongly connected components of $\overline{G}_f$
7. Find all closed sets of the resulting acyclic network
   (The subset C of nodes of network is a *closed set* iff for every vertex of C all of its predecessors are also members of C).
   For each closed sets find the corresponding separating k-set of G

End If

8. Add edge (x_i, v_j) to G.

End If

enddo

enddo

The results in Picard and Queyranne [38] establish the correctness of steps 5-7 for finding all separating k-sets. Let f be any maximum flow in a network N. The subset C of vertices of N is a *closed set* if and only if for every vertex $v \in C$ all of its predecessors are also members of C in N.

Lemma 7.1: [38] A cut $(S,\overline{S})$ separating s from t in N is a minimum cut if and only if S is a closed set of N containing s but not t.

Let R be the residual graph of N with respect to the maximum flow f. Let C be a strongly connected component of R and $v \in C$. Then based upon this Lemma if $v \in S$ then C is also subset of S.

Observation 7.2: There is one-to-one correspondence between the mincuts of $\overline{G}$ and the closed sets of N.

Definitions 7.1: Let N be the residual network of $\overline{G}$ with respect to a maximum flow. Shrink its strongly connected components into single vertices. Let L be the resulting acyclic network. (We will use L_{st} to emphasize the fact the maximum flow is taken between s'' and t').

Theorem 7.1: [4, 38] The resulting acyclic network L is the same for any maximum flow.

Based upon the above Theorem 7.1 and Lemma 7.1 the problem of finding all $s-t$ mincuts in L is reduced to the problem of finding all closed sets in L which we get after shrinking all strongly connected components of N. This justifies Step 6 of the algorithm. Hence, this establishes the correctness of the algorithm by our discussion preceding the algorithm.

Let us state time complexities of all steps of the above algorithm. We establish the bounds for Steps 1 and 7 below. Step 1 takes $O(max(k,n^{1/2})kmn^{1/2})$ time [18, 19]. Step 2 takes $O(m+n)$ time. Steps 3-8 are repeated kn times. Step 4 takes $O(\min(k,\sqrt{n})(m+n))$ time and step 5 takes $O(m+n)$ time. Step 6 also takes $O(m+n)$ time. Step 7 takes $O(\min(M_{ij}n + kn^2, M_{ij}kn + n))$ time, where M_{ij} is the number of separating k-sets between v_j and x_i. Step 7 takes $O(\min(Mn + k^2n^3, Mkn + kn^2))$ time over the execution of both loops, where M is the number of separating k-sets in G. Step 8 takes constant time. The total time for the algorithm is $\Theta(\min(Mn+k^2n^3, Mkn + knm\min(k,\sqrt{n}))) = O(2^kn^3)$.

Steps 4-7 show how to find all separating k-sets between a pair of vertices s and t of G. Let us see in detail how we actually do this. First, we construct a digraph $\overline{G} = (\overline{V},\overline{E})$ as stated in Chapter 2.3.

Lemma 7.2: [14, 34] If $(s,t) \notin E$ then the least cardinality (s,t) vertex separator is equal to the maximum number of vertex disjoint paths between s and t.

There are two ways to find a maximum flow from s'' to t' in $\overline{G}$. The first method is faster for small values of k, and works as follows. We find k directed paths in $\overline{G}$ from s'' to t', one path at a time. Call the resulting flow F. There are no k-sets separating s and t in G if and only if we can find a path from s'' to t' in the residual graph $\overline{G}$ with respect to F. This entire algorithm runs in $O(k(m+n))$ time.

The second algorithm is faster then the first for large values of k. In this algorithm we simply find a maximum flow in $O(m\sqrt{n})$ time using Dinic's algorithm in a unit network [49, 13].

Hence we implement step 4 using one of these algorithms depending on the value of k. Hence the time complexity of this step is $O(\min(k, \sqrt{n})(m+n))$.

There are several different algorithms which find all closed sets of an acyclic directed network [4, 47]. We will present one of them after the following lemma. Let us see know how many edges the directed acyclic network L can have.

Observation 7.3: Take any two adjacent vertices of G. There are four vertices in $\overline{G}$ corresponding to them. If the vertices of $\overline{G}$ corresponding to them were not used in a maximum flow from s'' to t' then in the residual network they will form a directed cycle of length 4 (see figure 2.2).

Let f be a flow from s'' to t' in $\overline{G}$, which we create by using shortest augmenting paths only. That is, we always choose a shortest augmenting path of the current residual graph to increase the flow. Let us call this flow a *shortest path flow* [11].

Lemma 7.3: Let f be a shortest path flow from s'' to t' in $\overline{G}$. Let N' be the residual graph of $\overline{G}$ with respect to f. Let L' be the acyclic graph which we get after shrinking the strongly connected components of N'. Then the number of edges in L' is $O(|f| \cdot n)$, where $n = |V|$. (Note that there is one path from s'' to t' in $\overline{G}$ for every unit of flow f. These paths are vertex disjoint).

Proof : We will prove that the number of edges of L' is at most $7ln$ by induction on l, where l is the number of paths in $\overline{G}$ for flow f ($|f| = l$).

Let $l = 1$. Take a shortest path P from s to t in G. There are no edges between vertices of the path P except the edges $E(P)$ (edges of the path itself), because the path is the shortest. A shortest path P in G corresponds to a shortest path $\overline{P}$ in $\overline{G}$. Every edge in P corresponds to two edges in $\overline{G}$, one forward edge which is part of $\overline{P}$, and one backward edge which connects two vertices of $\overline{P}$. Also every vertex of P corresponds to an edge in $\overline{P}$. Hence, the number of edges in $\overline{G}$

corresponding to P is $3p$, where p is the length of P in G.

All vertices of $V-P$ which are not on P have at most 3 edges incident on the path. Hence, the number of edges E' between vertices of P and the vertices in $V-P$ is at most $3(n-p)$, where p is the length of P. All edges of $\overline{G}$ which correspond to $E-E'-E(P)$ in G will be shrunk in N' by Observation 7.3. There are 2 edges in $\overline{G}$ corresponding to each edge of E', and there is one edge in $\overline{G}$ corresponding to each vertex of G. So, the number of edges in L' is at most $7n$.

Assume the claim is true for $l \leq r$, and let $l = r+1$. That means that there are at most $7rn$ edges in L' for flow f when $|f| = r$. Let $P^r=(P_1, \cdots, P_r)$ be the r vertex disjoint paths in $\overline{G}$ which form the flow f. Consider the edges $\tilde{E}$ in $\overline{G}$, which neither belong to paths P^r from s'' to t' nor are adjacent to (one of the endpoints belongs to the paths and the other one does not) them. By the assumption there are at most $7rn$ edges in $E - \tilde{E}$ adjacent to paths P^r from s'' to t' or on them. Let N_1 be the residual graph of $\overline{G}$ with respect to the flow f. Find the shortest augmenting path P in the residual graph N_1 of $\overline{G}$ with respect to f from s'' to t'. Let N_2 be the residual graph of $\overline{G}$ with respect to the new flow $f \cup P$. An edge $e \in \tilde{E}$ will be shrunk in N_2 unless e belongs to P or is adjacent to P. Note that all of the neighboring edges of all previous r paths P^r were already counted by the assumption. We claim that there are at most $7n$ edges adjacent to P or on P which are in $\tilde{E}$.

Case 1. Let $e = (v_1, v_2) \in P$ such that v_1 and v_2 are not vertices of P^r (v_1 and v_2 do not belong to any of the paths P^r). Let E_1 be the set of all edges of this type on P. Then there are at most 3 edges between each vertex of $\overline{V}$ and the endpoints of E_1 which are in $\tilde{E}$, because P is the shortest augmenting path.

Case 2. Let $e = (v_1, v_2) \in P$ such that $v_1 \in P^r$ and $v_2 \in P^r$. Then all of the edges adjacent to e were already counted by the assumption.

Case 3. Let $e=(v_1,v_2)\in P$ such that either $v_1\in P^r$ or $v_2\in P^r$ but not both. Note that the only edges of $\overline{G}$ which were reversed in N_1 are the edges of P^r. There are two subcases

Case A. $v_1\in P^r$. Then there is only one edge outgoing from v_2, which is the edge of Case 1. Hence, all of the edges adjacent to v_2 were already counted in Case 1, and all of the edges adjacent to v_1 were counted in Case 2.

Case B. $v_2\in P^r$. Then there is only one edge incoming to v_1 which is the edge of Case 1. Hence, all of the edges adjacent to v_1 were already counted in Case 1, and all of the edges adjacent in v_2 were counted by Case 2.

That conclude the proof of the induction step. Hence, the number of edges in the network L is at most $7(r+1)n$.

This concludes the proof of the lemma.

□ Lemma 7.3.

Corollary 7.1: The number of edges in network L is $O(kn)$. (Follows from Theorem 7.1 and Lemma 7.3).

An *antichain* in an acyclic network is a subset of nodes R such that for all pairs of nodes i and j in R, i is neither a predecessor nor a successor of j. The algorithm Antichain below finds all closed subsets of a directed acyclic network L, and runs in a linear time per subset [4].

Observation 7.4: [4, 38] There is one-to-one correspondence between antichains of L (together with all of their predecessors) and closed sets of L.

Let $V(L)$ be the set of vertices of L, and $E(L)$ be the set of edges of L. We now give an algorithm by Ball and Provan that finds all antichains in a single-source acyclic graph. The algorithm constructs the antichains in a set C, which is initially empty. The algorithm constructs successive antichains by adding a vertex to C at each step.

Antichain $(V(L),E(L),C;M)$

Step 1: Choose an $i \in V(L)$ of in-degree 0 and output $C \cup \{i\}$; set $M = 1$; if $V(L)-\{i\} = \varnothing$, then stop.

Step 2: Delete i from $V(L)$ to obtain $\hat{L} = (V(\hat{L}),E(\hat{L}))$ and Call Antichain $(V(\hat{L}),E(\hat{L}),C;M')$; set $M = M + M'$.

Step 3: Find all successors of i, denoted by SC(i). If $V(\hat{L})-SC(i)-\{i\} = \varnothing$, then stop; otherwise delete $SC(i) \cup \{i\}$ from $V(\hat{L})$ to obtain $\hat{L} = (V(\hat{L}),E(\hat{L}))$ and Call Antichain $(V(\hat{L}),E(\hat{L}),C \cup \{i\};M')$; set $M = M + M'$.

The correctness of this algorithm can be found in [4]. The time complexity of this algorithm is $O(M_{st}m)$, where m is the number of edges in L and M_{st} is the number of k-sets separating s and t in G. Note that all antichains are unique.

But it can be improved. Let us find all successors of each vertex of L before calling algorithm Antichain. That can be done in $O(mn)$ time, where m is the number of edges in L and n is the number of vertices in L. We build a depth first search tree T_x of successors of x for each vertex x of L. Each DFS takes $O(m)$ time per vertex, hence total time is $O(mn)$. Since L is an acyclic graph we can find all successors of x in linear time from T_x. Then all substeps of Step 3 of the algorithm Antichain take only $O(n)$ time, since we only need to read and merge lists of maximum size n. Step 2 clearly takes $O(n)$ time. Hence, the entire algorithm takes $O(M_{st}n + mn)$ time instead of $O(M_{st}m)$.

Recall that $m = O(kn)$ for L. The time spent by the algorithm to find all k-sets separating s and t in G is $O(\min(M_{st}kn + \min(k,\sqrt{n})kmn),(M_{st}n + k^2n^3))$.

Each antichain together with all of the predecessors for all of the vertices of the antichain creates a closed set S. The edges between set S and $\overline{S} = \overline{V} - S$ give a mincut in $\overline{G}$. All of these edges are *internal* edges of $\overline{G}$, and hence each edge corresponds to a vertex in G. The cardinality of a mincut in $\overline{G}$ is k, hence the edges of each mincut of $\overline{G}$ correspond to a separating k-set of G. Since no antichain is repeated, all separating k-sets are distinct.

Since we add edge (x_i, v_j) to G after processing the network with source x_i and sink v_j, the separating k-sets which we find from network $L_{x_i v_j}$ cannot be found again for any other pair of vertices in the updated G. Hence,

$$\sum_{i=1}^{i=k} \sum_{j=1}^{j=n} M_{ij} = M ,$$

where M is the total number of separating k-sets in G. Since $M = O(2^k \frac{n^2}{k})$ [26], we conclude that the total time complexity of the algorithm is $\Theta(\min(Mnk + kmn\min(k, \sqrt{n}), Mn + k^2 n^3)) = O(2^k n^3)$. Note that for finding all minimum size *edge separators* we need to find all minimum separating edge sets between at most $n-1$ pairs of vertices [4]. In contrast our algorithm needs to consider kn pairs in order to find all minimum size separating vertex sets.

7.2. Parallel Algorithm

In this section we present a parallel algorithm for finding all minimum size separating vertex sets of G. Note that if k is bigger than $polylog(n)$, then the time complexity of the sequential algorithm from the previous section might be greater than polynomial in n. The parallel algorithm is very similar to the sequential one, but every step of it will use a parallel version.

Algorithm 7.2.

1. Find connectivity k of a graph G
2. a). Take a set K of k vertices of G. Check if the set K is a separating k-set of G.

   b). Form all pairs of vertices (x,v), where $x \in K$ and $v \in G$. There are kn pairs. Number these pairs (arbitrarily).
3. For every pair (x,v) make a copy of the graph G and add an edge (y,z) for every pair (y,z) whose number is smaller than the number of the pair (x,v). Call this graph G_{xv}.
4. For every pair (x,v) create a directed graph $\overline{G}_{xv}$ by using the Even-Tarjan reduction. Find the maximum flow f_{xv} from x'' to v' in $\overline{G}_{xv}$.
5. If $f_{xv} = k$ then
6. Shrink all strongly connected components of the residual graph of $\overline{G}_{xv}$ with respect to f_{xv}. Let L_{xv} be the resulting acyclic directed graph.
7. Find all closed sets of L_{xv}.

   End If

Now, we will show how to implement each step efficiently in parallel. For step 1 we will use ideas from the sequential algorithm. We will take a subset K of k vertices of G and find the maximum flow between every vertex in K and every vertex of G. Note that since we can run all of these kn maximum flows in parallel, we can stop as soon as we find the maximum flow for one of the pairs.

For maximum flow we can use two different algorithms. The first algorithm is deterministic and is better for small values of k. It uses the straightforward implementation of the first sequential algorithm for this problem. It takes $O(k\log n)$ parallel time using $O(kn\frac{N^{\alpha}}{\log n})$

processors on a EREW PRAM, where N^{α} is the number of processors needed for matrix multiplication [28]. We use matrix multiplication for finding the shortest path in $\overline{G}_{xv}$ for each pair (x,v) in parallel. We need to repeat this at most k times.

The second algorithm is a randomized algorithm, but runs faster for large k. We find a maximum flow for every pair in a unit network using randomized parallel algorithm for matching [37, 28]. That takes $O(\log^2 n)$ parallel time using $O(kn^2N^{\alpha})$ processors on a CRCW PRAM and gives an RNC algorithm.

The first part of Step 2 takes $O(\log n)$ parallel time using $O(n+m)$ processors on a CRCW PRAM [45]. The second part of Step 2 takes $O(\log nk/\log\log nk)$ parallel time using $O(nk/\log kn)$ processors on a CREW PRAM using a parallel prefix computation [28]. Step 3 takes $O(1)$ parallel time using $O(nk(m+n))$ processors on a EREW PRAM. Step 4 is essentially the same as step 1. Step 6 takes $O(\log n)$ parallel time with $O(knN^{\alpha})$ processors on a CRCW PRAM [21]. Step 7 takes $O(\log n)$ parallel time using $O(M_{xv}^2 n^2)$ processors on a CRCW PRAM. We will show the implementation of this step below.

Let L_{xv} be the residual graph of $\overline{G}$ with respect to a maximum flow from x'' to v' with shrunk strongly connected components. Recall that there is one-to-one correspondence between the k-sets separating x and v in G and the *antichains* in L_{xv}. If we add to L_{xv} all edges between every vertex y and all of its successors, then we get a transitive closure L_{xv}^{+}. Then every antichain in L_{xv}^{+} still gives an $(S,\overline{S})$ cut in N_{xv}. The network L_{xv}^{+} is still acyclic and directed. We will use the adjacency matrix of L_{xv}^{+} to determine whether two vertices are incomparable.

For the problem of finding all antichains in a transitive closure of an acyclic network we will use well-known doubling technique. We will first find all antichains of sizes of powers of 2, and then use them to find all other antichains of all other sizes. Take every single vertex as an antichain. Take all antichains of the current size and take all possible unions between them. Now, check all created sets and remove all sets which are not antichains of the double size or

repetitions. Repeat that procedure log$n$ times. This creates all antichains of the sizes of powers of 2. Now we can use these antichains and get antichains of all others sizes using at most log$n$ antichains of the sizes powers of 2.

Algorithm 7.3.

1. Form the transitive closure L^+ of an input network L.
2. Take every vertex as antichain
3. Repeat log$n$ times

   Find all antichains of the double size using antichains of the current size
4. Find all other antichains of all other sizes using at most log$n$ independent sets of sizes of powers of 2.
5. Find all separating k-sets in the network using antichains.

Let us state the time complexities and processor bounds for each step of the above algorithm. We establish the bounds for Steps 3 and 4 below. Step 1 of the above algorithm runs in $O(\log n)$ time using $O(N^{\alpha})$ processors on a EREW PRAM, where N^{α} is the number of processors used for matrix multiplication [28]. Step 2 runs in $O(1)$ parallel time using $O(n)$ processors on a EREW PRAM. Step 3 runs in $O(\log n)$ parallel time using $O(M_{st}^2 n^2)$ processors on a CRCW PRAM as shown below. Step 4 runs in $O(\log n)$ parallel time using $O(M_{st}^2 n^2)$ processors on a CRCW PRAM [50]. Step 5 runs in $O(1)$ parallel time using $O(M_{st} m)$ processors on a CREW PRAM. Hence, total parallel time spent is $O(\log n)$ using $\Theta(M_{st}^2 n^2) = O(4^k \frac{n^6}{k^2})$ processors on a CRCW PRAM.

Let us see in detail the implementation of Step 3. Note that the number of antichains in L_{st}^{+} is equal to the number of k-sets which separate s and t in $G(M_{st})$. First of all, we take all S_1 and S_2 which are two different antichains of current size i, to create at most M_{st}^2 sets of size $2i$. In order to check if a created set is an antichain of size $2i$ we need to check two properties: first, that the created set is an antichain, and second, that its cardinality of a set is $2i$. For the first property we will check the $n \times n$ adjacency matrix of L_{xv}^{+}. For the second property we check if $a \neq b$ for every pair of elements (a,b), where $a \in S_1$ and $b \in S_2$. So we can check each set in $O(1)$ parallel time using $O(n^2)$ processors on a CRCW PRAM. Hence, on a CRCW PRAM Step 3 runs in $O(\log n)$ parallel time using $O(M_{st}^2 n^2)$ processors and Step 4 runs in $O(\log n)$ parallel time using $O(M_{st}^2 n^2)$ processors.

We have to run the above algorithm for kn L_{xv} graphs, one for each pair (x,v). But

$$\sum_{i=1}^{i=k}\sum_{j=1}^{j=n} M_{ij}^2 n^2 \leq (\sum_{i=1}^{i=k}\sum_{j=1}^{j=n} M_{ij})^2 n^2 = M^2 n^2 \leq 4^k \frac{n^6}{k^2},$$

since no separating k-set is created twice.

Hence, step 7 of the parallel Algorithm 7.2 for finding all minimum size separating vertex sets runs in $O(\log n)$ times using $O(4^k \frac{n^6}{k^2})$ CRCW PRAM processers for all kn pairs of vertices (x,v) $x \in K$ and $v \in V$.

The entire parallel algorithm runs in $O(k \log n)$ deterministic time using $\Theta(M^2 n^2 + knN^{\alpha}) = O(4^k \frac{n^6}{k^2} + knN^{\alpha})$ CRCW PRAM processors, or runs in $O(\log^2 n)$ randomized parallel time using $\Theta(M^2 n^2 + kn^2 N^{\alpha}) = O(4^k \frac{n^6}{k^2} + kn^2 N^{\alpha})$ CRCW PRAM processors, where N^{α} is the number of processors needed for matrix multiplication.

CHAPTER 8

CONCLUSION AND OPEN PROBLEMS

8.1. Summary of Results

In this dissertation we have studied various algorithms and bounds which arise in graph connectivity. We have presented bounds on the number of separating k-sets and algorithms for finding all separating k-sets for $k \geq 3$.

In Chapter 3 we presented lower bounds on the worst-case number of separating k-sets for an undirected k-connected graph. For $k = 2$ the graph that achieves the lower bound is the cycle. The lower bound is $\frac{n(n-3)}{2}$. For $k = 3$ the graph that achieves the lower bound is the wheel. The lower bound is $\frac{(n-1)(n-4)}{2}$. For general k the generalized cycle and generalized wheel give an $\Omega(2^k \frac{n^2}{k^2})$ lower bounds on the number of separating k-sets for an undirected k-connected graph on n vertices for even and odd k, respectively.

In Chapter 4 we presented the upper bounds on the number of separating k-sets as well as representations for an undirected k-connected graph on n vertices for small k's. For $k = 2$ the upper bound is the same as the lower bound, namely $\frac{n(n-3)}{2}$. For $k = 3$ the upper bound once again matches the lower bound of $\frac{(n-1)(n-4)}{2}$. The representation for $k = 2$ is based upon the decomposition of an biconnected graph into a collection of cycles, where every pair of vertices on a cycle is either a separating pair of the graph or a pair that separates an edge from the graph. The size of the representation is $O(n)$. The representation for $k = 3$ is based upon the decomposition of an triconnected graph into a collection of wheels, where every pair of vertices on the cycle of the wheel together with the center of the wheel is either a separating triplet of the graph or a triplet that separates an edge from the graph. The size of the representation is also $O(n)$.

In Chapter 5 we advanced the ideas of the previous chapter for general k. We achieved an $O(2^k \frac{n^2}{k})$ upper bound on the number of separating k-sets in an undirected k-connected graph on n vertices. The representation for general k is based upon the decomposition of an undirected k-connected graph into a collection of generalized wheels (or cycles). The size of this representation is $O(k^2 n)$.

In Chapter 6 we presented improved algorithms for testing graph four-connectivity. Consequently, these algorithms find all separating triplets of an triconnected graph if it is not four-connected. The algorithms are based upon the ear decomposition technique. The sequential algorithm runs in $O(n^2)$ time which is an improvement over $O(mn)$ time sequential algorithm which was the previous best. The parallel algorithm runs in $O(\log^2 n)$ time using $O(n^2)$ CRCW PRAM processors. That is an improvement over a previous $O(\log n)$ time algorithm which uses $O(mn \log n)$ CRCW PRAM processors.

In Chapter 7 we presented algorithms for finding all separating k-sets in an undirected k-connected graph on n vertices and m edges for general k. The sequential implementation runs in $\Theta(\min(\max(Mnk, knm\min(k,\sqrt{n})), \max(Mn, k^2 n^3))) = O(2^k n^3)$ time, where M is the number of minimum size separating vertex sets in the graph. The parallel implementation runs either in $O(k \log n)$ deterministic time using $\Theta(M^2 n^2 + knN^\alpha) = O(4^k \frac{n^6}{k^2})$ processors on a CRCW PRAM or in $O(\log^2 n)$ randomized time using $\Theta(M^2 n^2 + kn^2 N^\alpha) = O(4^k \frac{n^6}{k^2})$ processors, where n is the number of vertices in the graph and k is the connectivity of the graph, and N^α is the number of processors needed for parallel matrix multiplication of $n \times n$ matrices in $O(\log n)$ time.

8.2. Open Problems

As we have seen there are linear time algorithms for testing whether a graph is connected, biconnected and triconnected. For testing four-connectivity the best algorithm runs in $O(n^2)$ time. There are linear space representations for all separating k-sets of a k-connected undirected graph for any fixed k.

1). Is there a linear time sequential algorithm for testing graph k-connectivity for all fixed k?

Closely related is the question of finding the representation for all separating k-sets of a graph efficiently. Another question is finding the connectivity of a graph. The best deterministic algorithm uses network flows.

2). Is there another algorithm for finding the connectivity of a graph without use of network flows?

There are $O(\log n)$ time parallel algorithms for testing graph k-connectivity for any fixed k. But the number of processors is increasing by the factor of n for each k.

3). Is there an $O(\log n)$ time parallel algorithm for testing graph k-connectivity for any fixed k which uses $O(n^2)$ processors?

As we mention in Chapter 4 there are fast procedures to list all separating pairs and triplets in a biconnected and triconnected graphs, respectively. The procedure for listing all separating k-sets runs in $O(c^k M)$, where M is the number of separating k-sets in a graph. We are currently investigating better algorithms for this procedure. Also we plan to design efficient procedures for answering the following queries:

A) Decide if a set of k vertices is a separating k-set,

B) Decide if two vertices are in the same connected component with respect to all separating k-sets, and

C). Decide if two vertices are in the same component with respect to a separating k-set.

As we can easily see from Table 1 there is a gap between lower and upper bounds on the number of separating k-sets of an undirected k-connected graph. We conjecture that the real upper bound is the same as the current lower bound, and moreover, the graph that achieves it is the generalized cycle for even k and the generalized wheel for odd k.

The number of minimum size separating edge sets of an undirected graph is $O(n^2)$ as was stated before. But it was proved for general multigraphs.

4). Is there a better upper bound on the number of minimum size separating edge sets of an undirected simple graphs?

We have some preliminary results which indicate that the upper bound for k edge connected graphs for odd k is actually linear.

REFERENCES

[1] A. Agrawal and R. Barlow, "A Survey of Network Reliability and Domination Theory," *Operation Research*, vol. 32, pp. 478-492, 1984.

[2] M. Ball, R. M. Van Slyke, I. Gitman, and H. Frank, "Reliability of Packet Switching Broadcast Radio Networks," *IEEE Transactions on Circuits and Systems*, vol. CAS-23, pp. 806-813, 1976.

[3] M. O. Ball, "Computing Network Reliability," *Operation Research*, vol. 27, pp. 823-838, 1979.

[4] M. O. Ball and J. S. Provan, "Calculating Bounds on Reachability and Connectedness in Stochastic Networks," *Networks*, vol. 13, pp. 253-278, 1983.

[5] M. Becker, W. Degenhardt, J. Doenhardt, S. Hertel, G. Kaninke, W. Keber, K. Mehlhorn, S. Naher, H. Rohnert, and T. Winter, "A Probabilistic Algorithm for Vertex Connectivity of Graphs," *Inform. Proc. Letters*, vol. 15, pp. 135-136, 1982.

[6] R. E. Bixby, "The Minimum Number of Edges and Vertices in a Graph with Edge Connectivity n and m n-Bonds," *Networks*, vol. 5, pp. 253-298, 1975.

[7] J. A. Buzacott, "A Recursive Algorithm for Finding Reliability Measures Related to the Connection of Nodes in a Graph," *Networks* , vol. 10, pp. 311-327, 1980.

[8] C. J. Colbourn, "The Reliability Polynomial," *ARS Combinatorica*, vol. 21-A, pp. 31-58, 1986.

[9] R. Cole, "Parallel Merge Sort," in *Proc. 27th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 511-516, 1986.

[10] E. A. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation," *Soviet Math. Dokl.*, vol. 11, pp. 1277-1280, 1970.

[11] J. Edmonds and R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow problems," *J. ACM*, vol. 19, pp. 248-264, 1972.

[12] S. Even, "An Algorithm for Determining Whether the Connectivity of a Graph is at Least k," *SIAM J. Computing*, vol. 4, pp. 393-396, 1975.

[13] S. Even and R. E. Tarjan, "Network Flow and Testing Graph Connectivity," *SIAM J. Computing*, vol. 4, pp. 507-518, 1975.

[14] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, MD, 1979.

[15] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, Princeton, N.J., 1962.

[16] H. Frank and W. Chou, "Topological Optimization of Computer Networks," *Proceedings of the IEEE*, vol. 60, pp. 1385-1397, 1972.

[17] H. Frank, R. E. Kahn, and L. Kleinrock, "Computer Communication Network Design - Experience with Theory and Practice," in *Proc. Spring Joint Computer Conference*, pp. 255-270, 1972.

[18] Z. Galil, "Finding the Vertex Connectivity of Graphs," *SIAM J. Computing*, vol. 9, pp. 197-199, 1980.

[19] M. Girkar and M. Sohoni, "On Finding the Vertex Connectivity of Graphs," Tech. Report ACT-77, Coordinated Science Laboratory, University of Illinois, Urbana, Ill, May 1987.

[20] A. V. Goldberg and R. E. Tarjan, "Solving Minimum-Cost Flow Problems by Successive Approximation," in *Proc. 19th ACM Symp. on Theory of Computing*, pp. 7-18, 1987.

[21] D. S. Hirschberg, A. K. Chandra, and D. V. Sarwate, "Computing Connected Components on Parallel Computers," in *Communications of ACM*, pp. 461-464, 1979.

[22] J. E. Hopcroft and R. E. Tarjan, "Dividing Graph into Triconnected Components," *SIAM J. Computing*, pp. 135-158, 1973.

[23] A. Kanevsky, "On the Number of Minimum Size Separating Vertex Sets in a Graph," Tech Report ACT-80, Coordinated Science Laboratory, University of Illinois, Urbana, Ill, July 1987. Preliminary version was presented at Chicago Conference on Combinatorics and Complexity, June 1987.

[24] A. Kanevsky and V. Ramachandran, "Improved Algorithms for Graph Four-Connectivity," in *Proc. 28th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 252-259, October 1987.

[25] A. Kanevsky and V. Ramachandran, "A Characterization of Separating Pairs and Triplets in a Graph," Tech Report ACT-79, Coordinated Science Laboratory, University of Illinois, Urbana, Ill, July 1987. Preliminary version was presented at Chicago Conference on Combinatorics and Complexity, June 1987

[26] A. Kanevsky, "Compact Representation of the Separating k-sets of a Graph," Tech Report ACT-88 , Coordinated Science Laboratory, University of Illinois, Urbana, Ill , January 1988.

[27] A. Kanevsky, "Finding All Minimum Size Separating Vertex Sets in a Graph," Tech Report ACT-93, Coordinated Science Laboratory, University of Illinois, Urbana, Ill, June 1988.

[28] R. M. Karp and V. Ramachandran, "A Survey of Parallel Algorithms for Shared Memory Machines," in *Handbook of Theoretical Computer Science*, North Holland, 1989. to appear

[29] N. Linial, L. Lovasz, and A. Wigderson, "A Physical Interpretation of Graph Connectivity, and Its Algorithmic Applications," in *Proc. 27th IEEE Ann. Symp. on Foundations of Computer Science*, 1986.

[30] M. V. Lomonosov and V. P. Polesskii, "Lower Bounds of Network Reliability," *Problems of Information Transmission*, vol. 8, pp. 118-123, 1972.

[31] L. Lovasz, "Computing Ears and Branchings in Parallel," in *Proc. 26th IEEE Ann. Symp. on Foundations of Computer Science*, pp. 464-467, 1985.

[32] V. M. Malhotra, M. P. Kumar, and S. N. Maheshwari, "An $O(|V|^3)$ Algorithm for Finding Maximum Flows in Networks," *Information Processing Letters*, vol. 7, pp. 277-278, 1978.

[33] Y. Maon, B. Schieber, and U. Vishkin, "Parallel Ear Decomposition Search (EDS) and *st*-numbering in Graphs," in *VLSI Algorithms and Architectures*, pp. 34-45, 1986.

[34] K. Menger, "Zur Allgemeinen Kurventheorie," *Fund. Math*, vol. 10, pp. 96-115, 1927.

[35] G. L. Miller and V. Ramachandran, *Efficient Parallel Ear Decomposition with Applications*, unpublished manuscript January 1986.

[36] G. L. Miller and V. Ramachandran, "A New Graph Triconnectivity Algorithm and its Parallelization," in *Proc. 19th ACM Ann. Symp. on Theory of Computing*, pp. 335-344, 1987.

[37] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani , "Matching is as Easy as Matrix Inversion," *Combinatorica*, vol. 7, pp. 105-114, 1988.

[38] J. C. Picard and M. Queyranne, "On the Structure of All Minimum Cuts in a Network and Applications," *Mathematical Programming Study*, vol. 13, pp. 8-16, 1980.

[39] J. S. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability That a Graph is Connected," *SIAM J. Computing*, vol. 12, pp. 777-788, 1983.

[40] J. S. Provan and M. O. Ball, "Computing Network Reliability in Time Polynomial in the Number of Cuts," *Operation Research*, vol. 32, pp. 516-526, 1984.

[41] V. Ramachandran and U. Vishkin, "Efficient Parallel Triconnectivity in Logarithmic Time," in *Aegean Workshop on Computing*, Springer-Verlag LNCS 319, pp. 33-42, 1988.

[42] A. Ramanathan and C. J. Colbourn, "Counting Almost Minimum Cutsets with Reliability Applications," *Mathematical Programming*, vol. 39, pp. 253-261, 1987.

[43] A. Rosenthal, "Computing the Reliability of Complex Networks," *SIAM J. of Appl. Math.*, vol. 32, pp. 384-393, 1977.

[44] J. G. Shanthikumar, "Bounding Network-Reliability Using Consecutive Minimal Cutsets," *IEEE Transactions on Reliability*, vol. 37, pp. 45-49, 1988.

[45] Y. Shiloach and U. Vishkin, "An O(log n) Parallel Connectivity Algorithm," *J. of Algorithms*, vol. 3, pp. 57-63, 1982.

[46] A. W. Shogan, "Sequential Bounds of the Reliability of a Stochastic Network," *Operations Research*, vol. 34, pp. 1027-1044, 1976.

[47] L. Shrage and K. R. Baker, "Dynamic Programming solution of Sequencing problems with Precedence Constrains," *Operations Research*, vol. 26, pp. 444-449, 1978.

[48] R. E. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Computing*, vol. 1, pp. 146-160, 1972.

[49] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, 1983.

[50] R. E. Tarjan and U. Vishkin, "An Efficient Parallel Biconnectivity Algorithm," *SIAM J. Computing*, vol. 14, pp. 862-874, 1985.

[51] Y. H. Tsin and F. Y. Chin, "Efficient Parallel Algorithms for a Class of Graph Theoretic Problems," *SIAM J. Computing*, vol. 13, pp. 580-599, 1984.

[52] W. T. Tutte, *Connectivity in Graphs*, University of Toronto Press, 1966.

[53] R. Van Slyke and H. Frank, "Network Reliability Analysis: Part I," *Networks*, vol. 1, pp. 279-290, 1972.

[54] R. Van Slyke, H. Frank, and A. Kershenbaum, "Network Reliability Analysis: Part II," in *Reliability and Fault Tree Analysis*, R. E. Barlow, J. B. Fussell, N. D. Sigpurwalla, pp. 619-650, 1975.

[55] H. Whitney, "Non-Separable and Planar Graphs," *Trans. Amer. Math. Soc.*, vol. 34, pp. 339-362, 1932.

VITA

Arkady Kanevsky was born on [redacted] He studied at the Mos-[redacted] Civil Engineering from 1978 to 1981; and in the University of Illinois, Chicago, from 1981 to 1983, where he received the Bachelor of Science degree in Mathematics and Computer Science in 1983. He then attended the University of Illinois at Urbana-Champaign, where he received the Master of Science degree in Mathematics, in August 1985. He continued his education at the University of Illinois at Urbana-Champaign from September 1985 to August 1988, where he was a Research Assistant in the Coordinated Science Laboratory. He received the Doctor of Philosophy degree in Computer Science in 1988.