ELSEVIER Theoretical Computer Science 180 (1997) 169–180

Theoretical Computer Science

Efficient enumeration of all minimal separators in a graph

Hong Shen[a,*], Weifa Liang[b]

[a] *School of Computing and Information Technology, Griffith University, Nathan, QLD 4111, Australia*
[b] *Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia*

Received March 1995; revised October 1995
Communicated by O. Watanabe

Abstract

This paper presents an efficient algorithm for enumerating all minimal a–b separators separating given non-adjacent vertices a and b in an undirected connected simple graph $G = (V, E)$. Our algorithm requires $O(n^3 R_{ab})$ time, which improves the known result of $O(n^4 R_{ab})$ time for solving this problem, where $|V| = n$ and R_{ab} is the number of minimal a–b separators. The algorithm can be generalized for enumerating all minimal A–B separators that separate non-adjacent vertex sets $A, B \subset V$, and it requires $O(n^2(n - n_A - n_B)R_{AB})$ time in this case, where $n_A = |A|$, $n_B = |B|$ and R_{AB} is the number of all minimal A–B separators. Using the algorithm above as a routine, an efficient algorithm for enumerating all minimal separators of G separating G into at least two connected components is constructed. The algorithm runs in time $O(n^3 R_\Sigma^+ + n^4 R_\Sigma)$, which improves the known result of $O(n^6 R_\Sigma)$ time, where R_Σ is the number of all minimal separators of G and $R_\Sigma \leqslant R_\Sigma^+ = \sum_{1 \leqslant i \neq j \leqslant n, (v_i, v_j) \notin E} R_{v_i v_j} \leqslant (n(n-1)/2 - m) R_\Sigma$. Efficient parallelization of these algorithms is also discussed. It is shown that the first algorithm requires at most $O((n/\log n) R_{ab})$ time and the second one runs in time $O((n/\log n) R_\Sigma^+ + n \log n R_\Sigma)$ on a CREW PRAM with $O(n^3)$ processors.

1. Introduction

In a connected graph G, a *separator* S is a subset of vertices whose removal separates G into at least two connected components. S is called an *a–b separator* [6] if it disconnects vertices a and b. An (a–b) separator is said to be *minimal* if it does not contain any other (a–b) separator [6]. Determining (vertex) connectivity of a graph, which is a fundamental graph problem with important applications in many fields, is closely related to finding separators under various constraints [2,4,8].

The problem of enumerating all minimal a–b separators and all minimal separators of a graph is one of the fundamental enumeration problems in graph theory which has great practical importance in reliability analysis for networks and operation research

* Corresponding author. E-mail: hong@cit.gu.edu.au.

[1] This work was partially supported by Australia Research Council under its Small Grants Scheme.

PII S0304-3975(96)00148-X

for scheduling problems [1,5,8]. This problem has been addressed by many authors in various contexts [2,5,8,9]. In [9] it was shown that all minimal a–b separators and all minimal separators of an n-vertex graph can be enumerated in $O(n^4 R_{ab})$ and $O(n^6 R_\Sigma)$ time, respectively, where R_{ab} and R_Σ are the numbers of minimal a–b separators and minimal separators of the graph, respectively. No better results have been known yet.

A closely related problem to the above problem is to enumerate all a–b (or s–t) cutsets, where a *cutset* is a minimal edge set whose removal disconnects a and b [4]. This problem has been studied extensively in the literature [1,2,11]. It has been shown that all a–b cutsets in an undirected connected graph can be generated in time $O((n+m)\mu) = O(n^2\mu)$ [11], where n and m are the numbers of vertices and edges and μ is the number of a–b cutsets.

In this paper, we show that all minimal a–b separators and all minimal separators of G can be enumerated in time $O(n^3 R_{ab})$ and $O(n^3 R_\Sigma^+ + n^4 R_\Sigma)$, respectively, where $R_\Sigma \leqslant R_\Sigma^+ = \sum_{1\leqslant i \neq j \leqslant n, (v_i, v_j)\notin E} R_{v_i v_j} \leqslant (n(n-1)/2 - m) R_\Sigma$. Our results improve the known results by at least $O(n)$ factor [9]. The main idea resulting in this improvement is to enumerate all minimal a–b separators by generating an expansion tree which expands separators level by level via adjacent-vertex replacements, thus avoiding recursively expanding all previously generated separators which was required previously [9]. To the best of our knowledge, we have not yet seen the same approach which has appeared elsewhere. We also show how to generalize our enumerating algorithm for all minimal a–b separators for the case when a and b are two disjoint vertex sets, and present an efficient parallel implementation for the proposed algorithms.

2. Preliminaries

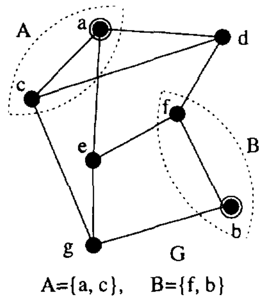
Let $G = (V, E)$ be an undirected connected simple graph. For any $X \subset V$ the subgraph *induced* by the vertices of X is denoted by $G[X] = (X, E(X))$, where $E(X) = \{(u,v) \in E \mid u,\ v \in X\}$.

Two vertices are said *adjacent* if they are connected by an edge. Two disjoint vertex subsets A and B of V are adjacent if there is at least one pair of adjacent vertices $u \in A$ and $v \in B$.

For any vertex $v \in V$, we denote by $N(v)$ the set of all vertices in V that are adjacent to v: $N(v) = \{w \in V \mid (v,w) \in E\}$.

For any subset $X \subset V$, we define $N(X) = \{w \in V - X \mid \exists v \in X,\ (v,w) \in E\}$.

A subset of V is called a *separator* of G if its removal separates G into at least two connected components. Given a pair of non-adjacent vertices a and b in V, a separator is called *an a–b separator* if it separates a and b in distinct connected components. If an a–b separator does not contain any other (a–b) separator, it is referred to as a *minimal a–b separator* [6]. It can be easily seen that the number of (different) minimal a–b separators in the general case can be exponential since any subset of $V - \{a,b\}$ can potentially be a minimal a–b separator, and so is for the total number of minimal separators of G. Clearly, all minimal a–b separators include all *minimal size*



Minimal a-b separators:
{c, e, d}, {g, e, d}, {g, f}, {c, e, f};
Minimal A-B separators:
{g, e, d}
All minimal separators of G:
{c, e, d}, {g, e, d}, {g, f}, {c, e, f};
{e, b, d}, {c, g, b}, {a, d, g}, {a, f, g};
{a, g, b, d}, {a, c, f}, {c, e, b};

Fig. 1. Minimal separators in a graph.

(a–b) separators [8] in which each exactly contains k vertices for a k vertex-connected graph.

Given an a–b separator S, we denote the connected components containing a and b in $G[V-S]$ by C_a and C_b, respectively. For any $X \subset V$, We define the *isolated set* of X, denoted by $I(X)$, to be the set of vertices in X that have no adjacent vertices in C_b of $G[V-X]$ and hence are not connected to C_b.

Let A and B be two disjoint non-adjacent subsets of V. Similarly, we define an A–B separator to be any subset of $V-(A \cup B)$ whose removal separates A and B in distinct connected components. A minimal A–B separator does not contain any other A–B separator.

Fig. 1 depicts examples of minimal a–b separators, minimal A–B separators and all minimal separators of G.

3. Level-by-level adjacent-vertex replacement

Given an undirected connected graph $G(V,E)$ and two non-adjacent vertices a and b in V, the following lemma, originated in [6], provides the necessary and sufficient condition for a minimal a–b separator. Its proof can be found in [9].

Lemma 1. *Let S be an a–b separator of $G(V,E)$. Then S is a minimal a–b separator of G if and only if there are two different connected components C_a and C_b of $G[V-S]$ that contain a and b, respectively, such that every vertex in S has a neighbour in both C_a and C_b.*

Let $S_i^{(j)}$ be the ith a–b minimal separator at level j, $j \geqslant 0$. From the above lemma, it is clear that $N(a) - I(N(a))$ is a minimal a–b separator. So we get the first minimal a–b separator

$$S_1^{(0)} = N(a) - I(N(a)). \tag{1}$$

The next minimal a–b separator can be generated from $S_1^{(0)}$ by replacing a vertex x in $S_1^{(0)}$ with all vertices in $N(x)-\{a\}$ and extracting all vertices in the isolated set $I(S_1^{(0)}\cup(N(x)-\{a\})$. Hence, if $S_1^{(0)}=\{x_1,x_2,\ldots,x_k\}$, we can obtain k other new minimal a–b separators by the following equation (note that $x_i\in I(S_1^{(0)}\cup(N(x_i)-\{a\}))$. Then we have

$$S_i^{(1)}=(S_1^{(0)}\cup(N(x_i)-\{a\}))-I(S_1^{(0)}\cup(N(x_i)-\{a\})),\quad 1\leqslant i\leqslant k. \tag{2}$$

From each $S_i^{(j)}$ we can generate at most $|S_i^{(j)}|$ new minimal a–b separators similarly via the above *vertex replacements* (some of them may be duplicates of the existing ones). This leads to a scheme of *level-by-level adjacent-vertex replacement*. Let $S^{(t)}$ denote any separator at level t, $t\geqslant 0$, and $S^{(-1)}=\{a\}$. We say that separator $S^{(t-1)}$ *precedes* separator $S^{(t)}$, denoted by $S^{(t-1)}\prec S^{(t)}$, if $S^{(t)}$ is generated from $S^{(t-1)}$ by the above vertex replacement scheme. For any $x'\in S^{(t-1)}$ and $x\in S^{(t)}$, we say that vertex x' *precedes* vertex x, denoted by $x'\prec x$, if $(x',x)\in E$ and $S^{(t-1)}\prec S^{(t)}$. For each $x\in S^{(t)}$, we define

$$N^-(x)=\{x'\mid x'\prec x\}, \tag{3}$$

and

$$N^+(x)=N(x)-N^-(x). \tag{4}$$

Lemma 2. *Let $S^{(t)}$ be a minimal a–b separator and $t\geqslant 0$. For any $x\in S^{(t)}$, if $b\notin N^+(x)$ then $S^{(t+1)}$ defined by the following equation is a minimal a–b separator and $S^{(t+1)}\neq S^{(t)}$:*

$$S^{(t+1)}=(S^{(t)}\cup N^+(x))-I(S^{(t)}\cup N^+(x)). \tag{5}$$

Proof. By Lemma 1 for any $x\in S^{(t)}$, clearly if $b\notin N(x)$ then $(S^{(t)}\cup N(x))-I(S^{(t)}\cup N(x))$ is a minimal a–b separator, since all vertices in $I(S^{(t)}\cup N(x))$ are not connected to the vertices in C_b, the connected component containing b, of $G[V-(S^{(t)}\cup N(x))]$. Clearly, $N^-(x)\subseteq I(S^{(t)}\cup N(x))$ since $N^-(x)\subseteq S^{(t-1)}$ and $S^{(t-1)}\prec S^{(t)}$. The lemma follows immediately by Eq. (4). □

Fig. 2(a) shows the relationship between $N^-(x)$ and $N^+(x)$.

When $b\in N^+(x)$, since the replacement of x with any subset of $N^+(x)-\{b\}$ (b cannot be inside an a–b separator) cannot block paths from $N^-(x)$ via x to b, it will not generate any new separators, as depicted in Fig. 2(b). So we have:

Lemma 3. *For $x\in S^{(t)}$ if $b\in N^+(x)$ then no vertex replacements on x will yield a new separator, where $S^{(t)}$ is a minimal a–b separator and $t\geqslant 0$.*

Our level-by-level adjacent-vertex replacement approach generates all minimal a–b separators at level t, $0\leqslant t\leqslant h$, where level 0 contains only one separator $S_1^{(0)}$ generated by Eq. (1) and in the following levels each separator $S^{(t+1)}$ is generated from its

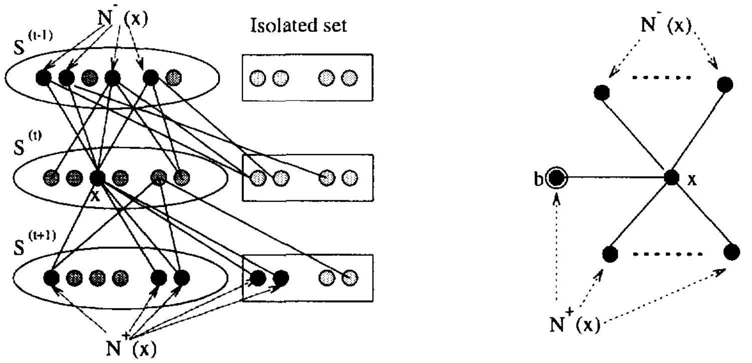


Fig. 2. $N^-(x)$ and $N^+(x)$ of $x \in S^{(t)}$ ($S^{(t-1)} \prec S^{(t)} \prec S^{(t+1)}$): (a) relationship between $N^-(x)$ and $N^+(x)$; (b) $N^+(x)$ containing b.

precedent $S^{(t)}$ via vertex replacement on a vertex $x \in S^{(t)}$ according to Eq. (5). The generation proceeds at each $x \in S^{(t)}$ if $b \notin N^+(x)$, and terminates at those x such that $b \in N^+(x)$ by Lemma 3. Clearly, $h \leqslant n-3$ since the maximal number of levels cannot be greater than the maximal distance from a to any other vertex in G. When G is a linear list with a and b being two end vertices, $h = n-3$.

Let L_t denote the set of minimal a–b separators generated at level t via level-by-level adjacent-vertex replacements, $0 \leqslant t \leqslant h$, where $h \leqslant n-3$ is the maximal distance from a to any other vertex in G. The following theorem shows that $\bigcup_{t=0}^{h} L_t$ contains all minimal a–b separators.

Theorem 1. *Let $L_0 = \{N(a) - I(N(a))\}$. If elements in L_i are generated from the elements in L_{i-1} via level-by-level adjacent-vertex replacements for $1 \leqslant i \leqslant h$, where $h \leqslant n-3$ is the maximal distance from a to any other vertex in G, then $\bigcup_{i=1}^{h} L_i$ contains all minimal a–b separators.*

Let $d(x)$ be the length of the shortest path (distance) from vertex $x \in V$ to a. To prove Theorem 1, we need the following lemma whose correctness is obvious from Eqs. (3) and (4):

Lemma 4. *For any $x \neq b \in V$, if $d(x) < d(b)$ then*

$$N^+(x) = \{v \mid (x,v) \in E,\ v \in V \text{ and } d(v) = d(x)+1\}. \tag{6}$$

This lemma shows that our vertex replacement on x proceeds in an *incremental distance* manner when $d(x) \leqslant d(b)$ in the sense that x is updated by its adjacent vertices which are one step farther from a than x. Now we begin to prove Theorem 1.

Proof. For any minimal a–b separator S in graph G, we can partition it into subsets $X_1, X_2, \ldots, X_p$, where all elements in X_i have the same distance h_i to vertex a and $h_i < h_j$ if $i < j$. We arrange the vertices in V by their *ranks* and redraw G accordingly:

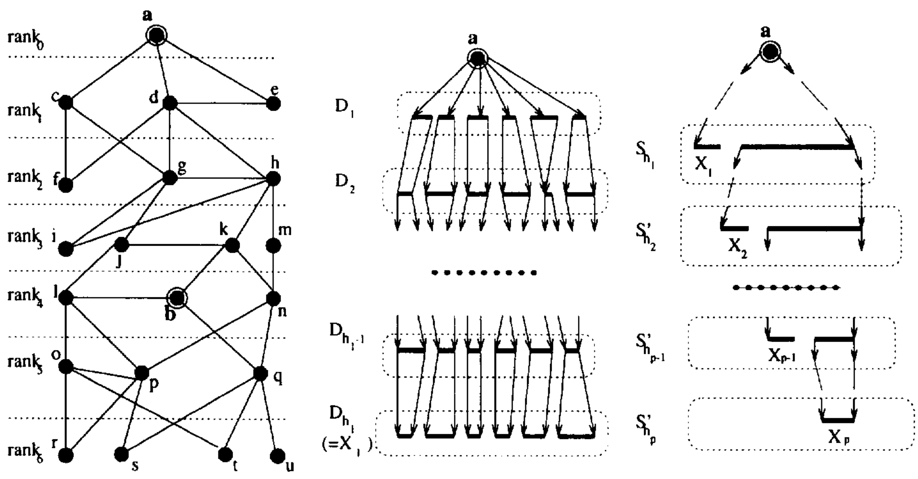


Fig. 3. Patterns of generating a minimal a–b separator: (a) drawing of G in ranks; (b) generating of $X = \{X_1\}$, $h_1 < h(b) - 1$; (c) generating of $S = \{X_1, X_2, \ldots, X_p\}$.

$rank_0 = \{a\}$, $rank_i = \{v \in V \mid d(v) = i\}$ for $1 \leqslant i \leqslant h$. Fig. 3(a) gives an example of this type of drawing. We say vertex u *dominates* vertex v if $d(u) < d(v)$ and $(u, v) \in E$. We call $D_i \subset rank_i$ the *dominator* of $D_{i+1} \subset rank_{i+1}$ if D_i is the minimal set such that all vertices in D_{i+1} are dominated *only* by vertices in D_i, while D_{i+1} is called the *dependent* of D_i.

First we consider the case that $h_p \leqslant d(b) - 1$. When $p = 1$, $X_1 \subset rank_{h_1}$ and can be generated from its dominator D_{h_1-1} in $rank_{h_1-1}$ via a series of vertex replacements by Eqs. (5) and (6), and D_t can be generated by its dominator in $rank_{t-1}$ for $1 \leqslant t \leqslant h_1 - 1$, as shown in Fig. 3(b). For $p > 1$, first we generate a separator $S_{h_1} \subset rank_{h_1}$. Clearly, $X_1 \subset S_{h_1}$ since otherwise $S = \bigcup_{i=1}^{p} X_i$ will not be minimal. Then we repeatedly replace one-by-one all vertices in $S_{h_1} - X_1$ with their dependents defined by Eq. (6) to expand $S_{h_1} - X_1$ into $S'_{h_2} \subset rank_{h_2}$ that is a separator of $G[V - X_1]$. Clearly $X_2 \subset S'_{h_2}$ and $S_{h_2} = X_1 \cup (S'_{h_2})$ is a separator of G. Assume that we have obtained $S_{h_{p-1}} \supset \bigcup_{i=1}^{p-1} X_i$. We now repeatedly one-by-one replace all vertices in $S_{h_{p-1}} - (\bigcup_{i=1}^{p-1} X_i)$ with their dependents defined by Eq. (6) to expand it into $S'_{h_p} \subset rank_{h_p}$ that is a separator of $G[V - (\bigcup_{i=1}^{p-1} X_i)]$. Clearly, $S_{h_p} = (\bigcup_{i=1}^{p-1} X_i) \cup (S'_{h_p})$ is a separator of G. Since $X_p \subset S'_{h_p}$ and $S = \bigcup_{i=1}^{p} X_i$ is a minimal separator, $X_p = S'_{h_p}$. Fig. 3(c) depicts this pattern of vertex replacement.

If $h_p \geqslant d(b) - 1$, obviously $p > 1$. All X_i are generated in a similar way to the above by Eqs. (4) and (5) with the exclusion of any updating at the adjacent vertices of b by Lemma 3. We leave the details to the reader.

Hence, any S can be generated by a sequence of adjacent-vertex replacements starting from $S_0 = N(a) - I(N(a))$. Since $\sum |X_i| \leqslant n - 2$, the length of this sequence is no more than $n - 2$. $\square$

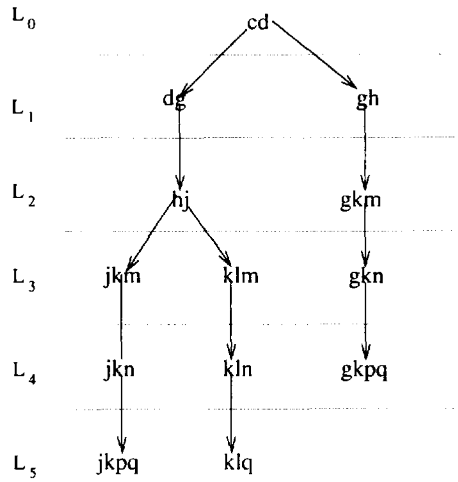


Fig. 4. The minimal-size expansion tree $\mathscr{T}$.

We now build an *expansion tree* which takes S_0 as the root and elements of L_t as the nodes at level t and connects a node $S^{(t-1)}$ at level $t-1$ to any node $S^{(t)}$ in level t if $S^{(t-1)} \prec S^{(t)}$, $1 \leqslant t \leqslant n-3$. It is clear that any minimal a–b separator is a node in the expansion tree.

We have reduced the problem of enumerating all minimal a–b separators which previously requires recursively expanding all the separators produced [9] to the problem of generating an expansion tree which expands separators only level by level. In order to maintain a minimal number of the expansions, we need to guarantee that it contains only distinct minimal a–b separators. Such an expansion tree is called the minimal-size expansion tree and is denoted by $\mathscr{T}$. We realize this by avoiding taking any duplicate that already exists in $\mathscr{T}$ when adding a new separator into it. This can be done by maintaining $\mathscr{T}$ in an AVL tree in lexicographical order of its separators on $(x_1, x_2, \ldots, x_{n-2})$ and using binary search when inserting a new separator (each step during the search requires $n-2$ (the height of $\mathscr{T}$) comparisons). A separator $S = \{x_{\rho_1}, x_{\rho_2}, \ldots, x_{\rho_k}\}$ can be represented by a vector $(b_1, b_2, \ldots, b_{n-2})$, where $b_i = 1$ if $\exists j \in \{1, \ldots, k\}$ such that $i = \rho_j$ and $b_i = 0$ otherwise, $1 \leqslant \rho_1 < \cdots < \rho_k \leqslant n-2$. Whenever S is inserted into $\mathscr{T}$, $\mathscr{T}$ is restructured through a number (at most the height of $\mathscr{T}$) of "rotations" [10] to ensure that the AVL tree properties are maintained. Hence we have the following lemma.

Lemma 5. *Let $\mathscr{T}$ contain a set of separators in $G(V,E)$. For any separator S determining whether $S \in \mathscr{T}$ requires $O(n \log |\mathscr{T}|)$ time.*

Fig. 4 shows the $\mathscr{T}$ generated on the graph in Fig. 3(a).

4. The algorithms

Based on the approach described above, our algorithm for generating all minimal a–b separators is presented below. The algorithm generates the node set of the minimal-size expansion tree $\mathscr{T}$ containing all minimal a–b separators via level-by-level adjacent-vertex replacements, and each node in $\mathscr{T}$ represents a distinct minimal a–b separator.

Procedure (a,b)-separators$(G, a, b, \mathscr{T})$

{*Generate all distinct minimal a–b separators for given non-adjacent vertices a and b in $G=(V,E)$, $|V|=n$. Input G, a and b. Output $\mathscr{T}=\bigcup_{i=0}^{n-3} L_i$, where L_i contains the nodes of the ith level in $\mathscr{T}$.*}

1 Compute the connected component C_b (containing b) of graph $G[V-N(a)]$;

2 Compute the isolated set $I(N(a))$ of set $N(a)$;

3 $L_0 := \{N(a)-I(N(a))\}$; $k := 0$;

4 **while** $(k \leqslant n-3) \wedge (C_b \neq \emptyset)$ **do**

  for each $S \in L_k$ **do**

    for each $x \in S$ that is not adjacent to b **do**

4.1   Compute the connected component C_b of graph $G[V-(S \cup N^+(x))]$;

    if $C_b \neq \emptyset$ **then**

4.2   Compute $I(S \cup N^+(x))$;

4.3 $S' := (S \cup N^+(x)) - I(S \cup N^+(x))$;

      {*Generate a new separator S' for the next level L_{k+1}.*}

4.4   **if** $S' \notin \bigcup_{i=0}^{t} L_i$ **then** $L_{k+1} := L_{k+1} \cup \{S'\}$;

      {*S' is distinct from those already in $\mathscr{T}$ and hence added to L_{k+1}.*}

  $k := k+1$

end.

The algorithm can enumerate all minimal a–b separators by Theorem 1, and these separators are distinct since the duplicates are excluded by Step 4.4. Each minimal a–b separator is generated correctly by Eq. (5).

In Step 1 we need to compute the connected component C_b containing b in graph $G[V-N(a)]$ which can be done by first computing the connected components of $G[V-N(a)]$, which takes time $O(|V|+|E|)=O(n^2)$, and then finding the one containing b in at most $O(n)$ time (there are at most $n-1$ connected components of $G[V-N(a)]$). So Step 1 requires $O(n^2)$ time. Applying the same for the computation of the connected component containing b in $G[V-N^+(x)]$) we know that Steps 4.1 can also be finished in $O(n^2)$ time. Note that $N^+(x)$ can be obtained in $O(n)$ time by Eqs. (3) and (4). Steps 2 and 4.2 require clearly at most $O(n^2)$ time. Since the maximal size of any separator is $n-2$, Steps 3 and 4.3 require time $O(n)$. By Lemma 5, Step 4.4 can be completed in time at most $O(n \log |\mathscr{T}|)=O(n^2)$, since the total number of minimal a–b separators in $\mathscr{T}$ is clearly at most $O(2^n)$. The third loop is executed at most $n-2$ times ($|S| \leqslant n-2$). Since $\mathscr{T}$ does not contain any duplicates, the first two nested loops are executed $\sum_{i=1}^{n-2} |L_i| = |\mathscr{T}|$ times. Hence, we have the following theorem.

Theorem 2. *For non-adjacent vertices a and b in an n-vertex undirected graph, all minimal a–b separators can be generated in $O(n^3 R_{ab})$ time, where R_{ab} is the number of minimal a–b separators.*

For given non-adjacent vertex sets A and B in G, the above algorithm can be adapted for generating all minimal A–B separators with almost no modification by simply replacing the single vertex a with set A and b with B.

Corollary 1. *Given non-adjacent subsets A and B of V in $G(V,E)$, all minimal A–B separators can be generated in $O(n^2(n-n_A-n_B)R_{AB})$ time, where $n_A = |A|$, $n_B = |B|$, $n = |V|$ and R_{AB} is the number of minimal A–B separators.*

Proof. $N(A)$ can be obtained in $O(n_A n)$ time. To compute the connected component C_B (containing all vertices in B) of graph $G[V-N(A)]$ if it exists (otherwise the algorithm terminates), we first compute the connected components in $G[V-N(A)]$ and then examine those whose size is at least n_B (at most $(n-n_A-|N(A)|)/n_B$ such ones) to find out which one contains all vertices in B. Having sorted these identified connected components by their sizes, we can realize the examination by binary search. Let n_i be the size of the ith one of these connected components, where $1 \leqslant i \leqslant (n-n_A-|N(A)|)/n_B$ and $\sum n_i = n-n_A$. Sorting takes $O(\sum(n_i \log n_i))$ time which is less than $O((n-n_A)\log(n-n_A))$, and searching takes $O(n_B \sum \log n_i)$ time which is at most $O(n_B((n-n_A)/n_B)\log(n-n_A)) = O((n-n_A)\log(n-n_A))$. As a result, it needs at most $O((n-n_A)^2)$ time for computing C_B in $G[V-N(A)]$. The computation of C_B of graph $G[V-N^+(x)]$ requires at most $O(n^2)$ time. The third loop in procedure `(a,b)-separators` now needs to be executed $n-n_A-n_B$ times. The total number of iterations of the first two nested loops is equal to the number of all minimal A–B separators, R_{AB}. This yields the corollary. $\square$

As the set of all minimal separators of G is the union of all minimal a–b separators for all different pairs of non-adjacent vertices $a, b \in V$; we therefore can use the procedure `(a,b)-separators` to generate all minimal separators for all $a, b \in V$ s.t. $(a,b) \notin E$, and then merge them to obtain all minimal separators of G. Below is the algorithm.

Procedure `all-separators`(G, $\mathscr{T}$)

{*Generate all minimal separators of G. Input $G = (V,E)$, $|V| = n$. Output $\mathscr{T} = \bigcup \mathscr{T}_c$, where $\mathscr{T}_c$ is the set of all minimal a–b separators for a pair $a, b (\in V)$ such that $(a,b) \notin E$.*}

1 **for** $i := 1$ **to** $n-1$ **do**

for $j := i+1$ **to** n **do**

```
        if $(v_i, v_j) \notin E$ then
            $(a,b)$-separators($G$, $v_i$, $v_j$, $\mathcal{T}_c$); $c := c + 1$;
        {*$c$ is initialized with value 0. Output separators in $\mathcal{T}_c$ are kept in an $AVL$
        tree in lexicographical order of $(x_1, x_2, \ldots, x_{n-2})$.*}
2   for $i := 0$ to $\log c - 1$ do
        for $j := 0$ to $\frac{c}{2^i+1} - 1$ do
            $\mathcal{T}_j := \mathcal{T}_j \cup \mathcal{T}_{j+\frac{c}{2^i+1}}$;
        $\mathcal{T} := \mathcal{T}_0$
        {*$\mathcal{T} = \bigcup_{i=0}^{c} \mathcal{T}_i$ contains all minimal separators of $G$.*}
    end.
```

Let R_Σ and R_Σ^+ be the number of all minimal separators of G and the summed number of minimal a–b separators for all different pairs of non-adjacent vertices a and b in V, respectively. Clearly, $1 \leqslant R_\Sigma^+/R_\Sigma \leqslant \frac{1}{2}(n(n-1)) - m$ since there are at most $\frac{1}{2}(n(n-1)) - m$ pairs of non-adjacent vertices in G and $R_\Sigma \geqslant \max\{|R_{ab}| \mid (a,b) \notin E\}$.

For Step 1, $\sum_{i=0}^{c} |\mathcal{T}_i| = R_\Sigma^+$, so $O(n^3 R_\Sigma^+)$ time is sufficient. In Step 2, we compute $\mathcal{T}_j \cup \mathcal{T}_k$ by mereging them using binary search, i.e. for each element in the smaller set searching its position in the larger set, where each operation involves $n-2$ comparisons (from x_1 to x_{n-2}). Thus, it requires time at most $O(nc|\mathcal{T}|\log|\mathcal{T}|) = O(n^4 R_\Sigma)$, where $c < \frac{1}{2}(n(n-1)) - m$ and $|\mathcal{T}| = R_\Sigma < 2^n$. Hence we have:

Corollary 2. *All minimal separators of $G(V,E)$ can be generated in at most $O(n^3 R_\Sigma^+ + n^4 R_\Sigma)$ time, where $R_\Sigma^+ = \sum_{1 \leqslant i \neq j \leqslant n, (v_i, v_j) \notin E} R_{v_i v_j}$, and R_Σ is the number of all minimal separators of G.*

Clearly, our algorithm has a speedup $O(\min\{n^3 R_\Sigma / R_\Sigma^+, n^2\})$ over the one in [9], and since $1 \leqslant R_\Sigma^+/R_\Sigma \leqslant \frac{1}{2}(n(n-1)) - m$, this speedup is between $O(n)$ and $O(n^2)$.

Finally, we show how our algorithms can be efficiently parallelized on PRAM. For procedure (a,b)-separators, we use $O(n^3)$ processors on a CREW PRAM. The detailed analysis is as follows. Steps 1 and 3 require $O(\log^2 n)$ time for computing connected components in G [7] (we can do it in $O(\log n \log\log n)$ time with the recent result of [3]). Step 2 takes at most $O(\log n)$ time. When generating new separators from S in L_k (the third loop in the procedure), we assign $O(n^2)$ processors to each of the $n-2$ (at most) children of S so that all them can be generated in parallel (the third loop in the procedure). Obviously, $N^+(x)$ for any $x \in S$ can be found in $O(\log n)$ time and the connected component C_b of $G[V - N^+(x)]$ can be computed in $O(\log^2 n)$ time [7]. For Step 4.2 computing $I(S \cup N^+(x))$, assign $O(n)$ processors to each element v in $S \cup N^+(x)$ which computes $N^+(v)$ and determines whether $N^+(v) \cap C_b = \emptyset$ in $O(\log n)$ time. Step 4.3 is completed in $O(\log n)$ time. Here we get at most $n-2$ new separators S_1', $S_2', \ldots, S_{n-2}'$, each represented as $(x_1, x_2, \ldots, x_{n-2})$. We assign $O(n)$ processors to each pair (S_i', S_j') for $i < j$ and check their equality in $O(1)$ time, and then collect the results and identify the duplicates in time $O(\log n)$. Finally, for all distinct ones (each with $O(n^2)$ processors) we do in parallel for each S_i' an n^2-way search on $\mathcal{T}$ (each operation requires O(1) time using O(n) processors) and insert it if not already in $\mathcal{T}$. Maintaining $\mathcal{T}$ in a variant of B-tree of height O($n/\log n$) and order O(n), we can complete this step in at most O($n/\log n$) time, since $|\mathcal{T}|$ is at most O(2^n). Clearly, the first two nested loops in the procedure is executed at most O($|\mathcal{T}|$) times. Hence we have:

Theorem 3. *Given a pair of non-adjacent vertices a and b in a graph, all minimal a–b separators can be generated in* O($(n/\log n)R_{ab}$) *time using* O(n^3) *processors on a* CREW PRAM, *where* R_{ab} *is the number of minimal a–b separators.*

Based on the above theorem, the following corollary for parallelization of procedure `all-separators` is straightforward. Here in Step 2 computing $\mathcal{T} = \bigcup_{i=0}^{c} \mathcal{T}_i$ we assign O(n^2) processors to each $\mathcal{T}_i$ and use O(n) processors for each step of comparison of a pair of separators. We leave the proof to the reader.

Corollary 3. *All minimal separators of* $G = (V,E)$ *can be generated in at most* O($(n/\log n)R_{\Sigma}^{+} + n\log nR_{\Sigma}$) *time using* O($n^3$) *processors on a* CREW PRAM, *where* $R_{\Sigma}^{+} = \sum_{1\leqslant i\neq j\leqslant n,(v_i,v_j)\notin E} R_{v_iv_j}$ *and* R_{Σ} *is the number of all minimal separators of G.*

5. Concluding remarks

We have presented two new algorithms for enumerating all minimal a–b separators and all minimal separators of a graph, respectively. Our algorithms use a greedy approach and enumerate these separators by a level-by-level adjacent-vertex replacement scheme, where the separators at each level are generated via one-by-one replacing every vertex of each separator in the previous level with a set of its adjacent vertices, thus avoiding expanding all previously generated separators and making the search reduced considerably. The proposed algorithms improve the known result of time complexity O(n^4R_{ab}) to O(n^3R_{ab}) for generating all minimal a–b separators, and O(n^6R_{Σ}) to O($n^3R_{\Sigma}^{+} + n^4R_{\Sigma}$) for generating all minimal separators of G [9], where R_{ab} and R_{Σ} are the number of all minimal a–b separators and all minimal separators of G respectively, and $R_{\Sigma}\leqslant R_{\Sigma}^{+} = \sum_{1\leqslant i\neq j\leqslant n,(v_i,v_j)\notin E} R_{v_iv_j} \leqslant (n(n-1)/2 - m)R_{\Sigma}$.

Our first algorithm can be adapted for the more general case to generate all minimal A–B separators for given non-adjacent vertex sets A and B in G. We have shown that in this case the algorithm works in O($n^2(n - n_A - n_B)R_{AB}$) time, where $n_A = |A|$, $n_B = |B|$ and R_{AB} is the number of all minimal A–B separators.

Both of our algorithms can be efficiently parallelized. We have shown that, using O(n^3) processors on a CREW PRAM, the first algorithm requires at most O($(n/\log n)$ R_{ab}) time, and the second one runs in time O($(n/\log n)R_{\Sigma}^{+} + n\log nR_{\Sigma}$).

A challenging open problem is to find an algorithm that generates all minimal a–b separators in the same time as generating all a–b cutsets for which O(n^2) per cutset algorithm was already known [11].

It will be interesting to see whether we can find a parallel algorithm that generates all minimal a–b separators in polylogarithmic time per separator using polynomial number of processors in n.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] H. Ariyoshi, Cut-set graph and systematic generation of separating sets, *IEEE Trans. Circuit Theory* **CT-19** (1972) 233–240.

[2] S. Arnberg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey, *BIT* **25** (1985) 2–23.

[3] K.W. Chong and T.W. Lam, Connected components in $O(\log n \log\log n)$ time on the EREW PRAM, in: *Proc. 4th Ann. ACM-SIAM Symp. Discrete Algorithms* (1993) 11–20.

[4] A. Gibbons, *Algorithmic Graph Theory* (Cambridge Univ. Press, Cambridge, 1985).

[5] L.A. Goldberg, *Efficient Algorithms for Listing Combinatorial Structures* (Cambridge Univ. Press, Cambridge, 1993).

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

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

[8] A. Kanevsky, On the number of minimum size separating vertex sets in a graph and how to find all of them, in: *Proc. 1st Ann. ACM-SIAM Symp. Discrete Algorithms* (1990) 411–421.

[9] T. Kloks and D. Kratsh, Finding all minimal separators of a graph, in: *Proc. Theoretical Aspects of Computer Sci.*, Lecture Notes in Computer Science, Vol. 775 (Springer, Berlin, 1994) 759–767.

[10] D.E. Knuth, *The Art of Computer Programming, Vol 3: Sorting and Searching* (Addison-Wesley, Reading, MA, 1973).

[11] S. Tsukiyama, I. Shirakawa and H. Ozaki, An algorithm to enumerate all cutsets of a graph in linear time per cutset, *J. ACM* **27** (1980) 619–632.