Bipartition of Biconnected Graphs

Hitoshi Suzuki, Naomi Takahashi and Takao Nishizeki

Department of Electrical Communications
Faculty of Engineering, Tohoku University
Sendai 980, Japan

Abstract. This paper presents a linear algorithm for finding two disjoint connected subgraphs in a biconnected graph each of which contains a specified vertex and has a specified number of vertices.

1. INTRODUCTION

We present a linear algorithm for solving bipartition problem for a biconnected graph. The *biparitition problem* is the following:

Input: (1) an undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges;
(2) $s_1, s_2 \in V, s_1 \neq s_2$; and
(3) two natural numbers $n_1, n_2 \in N$ such that $n_1 + n_2 = n$.

Output: a partition (V_1, V_2) of vertex set V such that
(a) $s_1 \in V_1$ and $s_2 \in V_2$;
(b) $|V_1| = n_1$ and $|V_2| = n_2$; and
(c) V_1 and V_2 induce connected subgraphs of G.

Fig. 1 depicts an instance of the problem above and a solution of it.

Clearly the problem has no solution for some graphs. Furthermore the problem determining whether the above problem has a solution is NP-complete if G may be not biconnected[DF]. However, Györi and Lovász independently proved the following theorem.

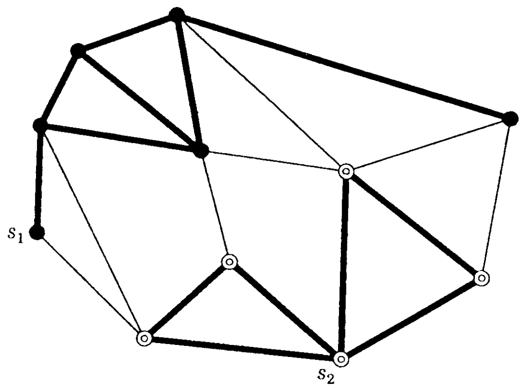

Fig. 1 An instance of the bipartition problem and a solution(thick lines depict the subgraphs induced from V_1 and V_2).

THEOREM 1 [Gy,Lo]. If G is k-connected, then *k-partition problem* has a solution. ■

The *k-partition problem* is one to find k disjoint connected subgraphs in a graph each of which contains a specified vertex and has a specified number of vertices. Since the bipartition problem is a subproblem of k-partition problem, it necessarily has a solution if the given graph

G is biconnected. Although the proof by Györi provides a polynomial algorithm if $k = 2$, naive implementation of the algorithm does not run in linear time.

Our algorithm is not based on the proofs but based on characteristics of a depth first search tree in a biconnected graph.

2. PRELIMINARIES

Let $G = (V, E)$ be an undirected connected graph with vertex set V and edge set E. The vertex set and edge set of a graph H are denoted by $V(H)$ and $E(H)$, respectively. For an edge (v, w) in a graph G, $G/(v, w)$ is the graph obtained from G by contracting edge (v, w), that is, identifying two vertices v and u and removing the resulting self loop and multiple edges, if any. For two vertices v and w in G, $G + (v, w)$ is the graph obtained by adding new edge (v, w) to G if G does not include edge (v, w), or G otherwise. For a set X of vertices in $V(G)$, $G - X$ is the graph obtained by removing all the vertices in X and all the edges incident with vertices in X from G.

Let T be a depth first search tree of G. For each vertex $v \in V$, the set of descendants of v including v itself is denoted by DES(v). Clearly the following lemma holds.

LEMMA 1. Let G be an undirected graph and T be a depth first search tree of G. Then G is biconnected if and only if the root of T has exactly one child and, for each vertex v other than the root and the child of it, an edge of G joins an ancestor of the grandparent of v and a descendant of v. ■

In this paper, ancestors and descendants of $v \in V$ include v itself.

3. ALGORITHM

In this section, we present a linear algorithm PART2 for solving bipartition problem for a biconnected graph G. Since the subgraphs of G induced from V_1 and V_2 cannot include edge (s_1, s_2) even if there is, a solution of the bipartition problem for $G + (s_1, s_2)$ is always one for G. Therefore, in the algorithm below, we may assume that G has edge (s_1, s_2). Let T be a depth first search tree with s_1 as the root and s_2 as the child of the root. Since an edge joins s_1 and s_2, we can find a depth first search tree like above by first searching s_2. The algorithm is the following.

function PART2(G, T, s_1, s_2, n_1, n_2);
begin
(1) **if** $n_1 = 1$ **then**
  return($\{s_1\}, V(G) - \{s_1\}$)
 elseif $n_2 = 1$ **then**
  return($V(G) - \{s_2\}, \{s_2\}$);
(2) let a be an arbitrary child of s_2;
 if s_2 has more than one child **then** {see Fig. 2. Note that Lemma 1 implies that, for every son v of s_2, s_1 is adjacent to a vertex in DES(v)}
(2.1) **if** $|\mathrm{DES}(a) \cup \{s_2\}| \le n_2$ **then**
  begin {include DES(a) into V_2}
   $V_2 :=$ DES(a);
   $G_{21} := G - V_2$;
   $T_{21} := T - V_2$;
   $(V_1, V_2') :=$ PART2($G_{21}, T_{21}, s_1, s_2,$
      $n_1, |V(G_{21})| - n_1$);
   return $(V_1, V_2 \cup V_2')$
  end
(2.2) **else** {$|\mathrm{DES}(a) \cup \{s_2\}| > n_2$, that is,
  $|(\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}) \cup \{s_1\}| \le n_1$}
  begin {include DES(s_2) $-$ DES(a) $- \{s_2\}$ into V_1}

$V_1 := \mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\};$

$G_{22} := G - V_1;$

$T_{22} := T - V_1;$

$(V_1', V_2) := \mathrm{PART2}(G_{22}, T_{22}, s_1, s_2, |V(G_{22})| - n_2, n_2);$

return $(V_1 \cup V_1', V_2)$

end

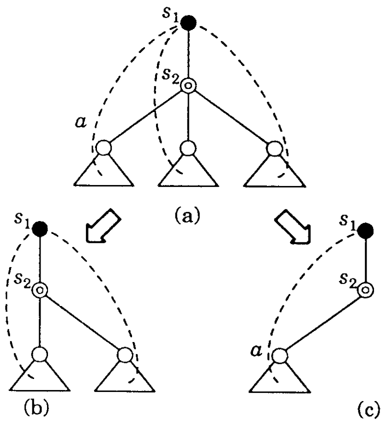

Fig. 2 (a) G , (b) G_{21} and (c) G_{22}

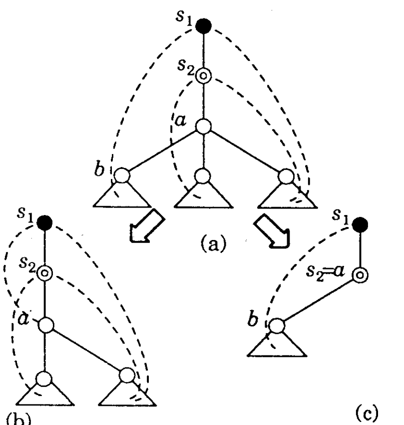

Fig. 3 (a) G , (b) G_{311} and (c) G_{312}

(3) **else** $\{s_2$ has exactly one child$\}$

begin

let b be an arbitrary grandchild of s_2;

(3.1) **if** s_1 is adjacent to a vertex in DES(b) **then** {see Fig. 3}

(3.1.1) **if** $|\mathrm{DES}(b) \cup \{s_1\}| \leq n_1$ **then**

begin {include DES(b) into V_1}

$V_1 := \mathrm{DES}(b);$

$G_{311} := G - V_1 + (s_1, a);$ {since all vertices in DES(b) are included into V_1, we may assume that a, the parent of b, is adjacent to s_1}

$T_{311} := T - V_1;$

$(V_1', V_2) := \mathrm{PART2}(G_{311}, T_{311}, s_1, s_2, |V(G_{311})| - n_2, n_2);$

return $(V_1 \cup V_1', V_2)$

end

(3.1.2) **else** $\{|\mathrm{DES}(b) \cup \{s_1\}| > n_1$, that is, $|(\mathrm{DES}(a) - \mathrm{DES}(b)) \cup \{s_2\})| < n_2\}$

begin {include DES(a) − DES(b) into V_2}

$V_2 := \mathrm{DES}(a) - \mathrm{DES}(b);$

$G_{312} := (G - V_2)/(s_2, a);$

$T_{312} := (T - V_2)/(s_2, a);$

$(V_1, V_2') := \mathrm{PART2}(G_{312}, T_{312}, s_1, s_2, n_1, |V(G_{312})| - n_1);$

return $(V_1, V_2 \cup V_2')$

end

(3.2) **else** $\{s_1$ is adjacent to no vertex in DES(b), and hence s_2 is adjacent to a vertex in DES(b). see Fig. 4$\}$

(3.2.1) **if** $|\mathrm{DES}(b) \cup \{s_2\}| \leq n_2$ **then**

begin {include DES(b) into V_2}

$V_2 := \mathrm{DES}(b);$

$G_{321} := G - V_2;$

$T_{321} := T - V_2$;
$(V_1, V_2') :=$ PART2$(G_{321}, T_{321}, s_1, s_2,$
$n_1, |V(G_{321})| - n_1)$;
return $(V_1, V_2 \cup V_2')$
end
(3.2.2) **else** $\{|\mathrm{DES}(b) \cup \{s_1\}| > n_2$, that is, $|(\mathrm{DES}(a) - \mathrm{DES}(b)) \cup \{s_2\})| \leq n_1\}$
begin {include DES(a) − DES(b) into V_1}
$V_1 := \mathrm{DES}(a) - \mathrm{DES}(b)$;
$G_{322} := (G - (V_1 - \{a\}))/(s_1, a)$;
$T_{322} := (T - (V_1 - \{a\}))/(s_1, a)$;
{although (s_1, a) is not an edge in T, $/(s_1, a)$ is to identify two vertices s_1 and a. Select s_2 as the root of T_{322}}
$(V_2, V_1') :=$ PART2$(G_{322}, T_{322},$
$s_2, s_1, n_2, |V(G_{322})| - n_1)$;
return $(V_1 \cup V_1', V_2)$
end
end
end;

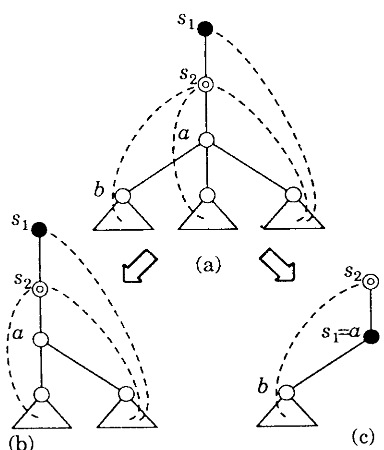

Fig. 4 (a) G , (b) G_{321} and (c) G_{322}

The following lemma can be easily proved from Lemma 1.

LEMMA 2. Modified graphs G_{21}, G_{22}, G_{311}, G_{312}, G_{321} and G_{322} in PART2 are biconnected, $T_{21}, T_{22}, T_{311}, T_{312}$ and T_{321} are depth first search trees with s_1 as the root in G_{21}, G_{22}, G_{311}, G_{312} and G_{321}, respectively, and T_{322} is a depth first search tree with s_2 as the root in G_{322}. ■

One can easily prove the correctness of the algorithm by using Lemma 2.

In order to implement the algorithm above so that it runs in $O(m)$ time, we use low(v) and id(v): for each vertex $v \in V$, low(v) is defined to be the vertex u adjacent to a vertices in DES(v) such that the depth first number of u is minimum, and

$$\mathrm{id}(v) = \begin{cases} 0, & \text{if } v \notin V_1 \cup V_2; \\ 1, & \text{if } v \in V_1; \text{ and} \\ 2, & \text{if } v \in V_2. \end{cases}$$

Then we can determine whether s_1 is adjacent to a vertex in DES(b) (in (3.1)) by checking whether id(low(b)) = 1. For each $v \in V - \{s_1, s_2\}$, id(v) is initially set to be zero and must be updated according to proceeding of the algorithm. However, it is not necessary to update low(v). Although, for example, after an execution of (3.2.2), for some vertices v in DES(a) − DES(b), id(low(v)) may become incorrect, the vertices are included into V_1 and hence will not be selected as b. Therefore, we need to compute low(v) for all $v \in V$ only once at the beginning of the algorithm. Furthermore, moving s_2 (or s_1) to a instead of contracting edge (s_2, a) (resp. (s_1, a)), we can implement the algorithm so that it does not modifies G nor T.

A depth first search tree of a graph can be found in $O(m)$ time. Furthermore low(v) for all vertices $v \in V$ can be computed also in $O(m)$

time. All the other tasks can be done in $O(n)$ time. Thus the bipartition problem for a biconnected graph can be solved in $O(m)$ time.

Remark

A slightly extended problem can be solved by a similar algorithm. The problem is the following.

Input: (1) an undirected biconnected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges;
(2) $s_1, s_2, x, y \in V$, s_1, s_2, x and y are all distinct; and
(3) two natural numbers $n_1, n_2 \in N$ such that $n_1 + n_2 \geq n$.

Output: a partition (V_1, V_2) of vertex set V such that
(a) $s_1 \in V_1$ and $s_2 \in V_2$;
(b1) $|V_1| = n_1$ or $(\{x, y\} \cap V_1 \neq \phi$ and $|V_1| < n_1)$;
(b2) $|V_2| = n_2$ or $(\{x, y\} \cap V_2 \neq \phi$ and $|V_2| < n_2)$; and
(c) V_1 and V_2 induce connected subgraphs of G.

Our algorithm PART2* which solves the problem above is similar to PART2, but sets of vertices which will be included included into V_1 or V_2 are chosen more carefully. The part of PART2* corresponding to (2.1) and (2.2) in PART2 is the following.

(2.1)**if** $(|\mathrm{DES}(a) \cup \{s_2\}| \leq n_2$ and $|\mathrm{DES}(a) \cap \{x, y\}| \neq 2)$ or $|(\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}) \cup \{s_1\}| \geq n_1)$ **then**
  begin {include DES(a) into V_2}
    $V_2 := \mathrm{DES}(a)$;
    $G_{21} := G - V_2$;
    $T_{21} := T - V_2$;
    if $|\mathrm{DES}(a) \cap \{x, y\}| = 0$ **then**
    $(V_1, V_2') := \mathrm{PART2}^*(G_{21}, T_{21}, s_1, s_2, n_1, n_2 - |V_2|)$;
    else $\{|\mathrm{DES}(a) \cap \{x, y\}| = 1$ or $2\}$
     begin
      $n_1' := \min\{n_1, |V(G_{21})| - 1\}$;
      $(V_1, V_2') := \mathrm{PART2}(G_{21}, T_{21}, s_1, s_2, n_1', |V(G_{21})| - n_1')$;
      {if $n_1' < n_1$, then $|\mathrm{DES}(a) \cap \{x, y\}| = 1$ and $|V(G_{21})| - n_1' = 1$, and hence V_1 will include x or y}
     end;
    return $(V_1, V_2 \cup V_2')$
  end

(2.2)**else** $\{(|\mathrm{DES}(a) \cup \{s_2\}| > n_2$ or $|\mathrm{DES}(a) \cap \{x, y\}| = 2)$ and $|(\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}) \cup \{s_1\}| < n_1\}$
  begin {include $\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}$ into V_1}
    $V_1 := \mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}$;
    $G_{22} := G - V_1$;
    $T_{22} := T - V_1$;
    if $|\mathrm{DES}(a) \cap \{x, y\}| = 2$ **then**
     $(V_1', V_2) := \mathrm{PART2}^*(G_{21}, T_{21}, s_1, s_2, n_1 - |V_1|, n_2)$;
     else $\{|\mathrm{DES}(a) \cap \{x, y\}| = 0$ or $1\}$
     $(V_1', V_2) := \mathrm{PART2}(G_{21}, T_{21}, s_1, s_2, |V(G_{22})| - n_2, n_2)$;
     $\{|V(G_{22})| - n_2 > 1$, since $|\mathrm{DES}(a) \cap \{x, y\}| \neq 2$ and hence $|V(G_{22})| - 1 = |\mathrm{DES}(a) \cup \{s_2\}| > n_2\}$
     return $(V_1 \cup V_1', V_2)$
  end

The remaining part of PART2* can be similarly derived from PART2. $|\mathrm{DES}(v) \cap \{x, y\}|$

for all vertices v can be computed in $O(n)$ time. Thus the execution time of PART2* is $O(m)$.

Acknowledgment

We would like to thank Professors A. Saito, N. Saito and S. Ueno for helpful discussion. This research is partly supported by Grant in Aid for Scientific Research of the Ministry of Education, Science, and Culture, under grant number: General Research (C) 01550275 (1989).

References

[DF] M. E. Dyer and A. M. Frieze, On the complexity of partitioning graph into connected subgraphs, Discrete Appl. Math., 10, 1985, pp.139-153.

[Gy] E. Györi, On division of connected subgraphs, Combinatorics (Proc. Fifth Hungarian Combinatorial Coll., 1976, Keszthely), Bolyai-North-Holland, 1978, pp.485-494.

[Lo] L. Lovász, A homology theory for spanning trees of a graph, Acta Math. Acad. Sci. Hunger. 30, 1977, pp.241-251.