3L-4

Bipartition of Biconnected Graphs

Hitoshi Suzuki, Naomi Takahashi and Takao Nishizeki

Tohoku University

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.

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.

THEOREM 1 [Gy,Lo]. If G is k-connected, then the *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.

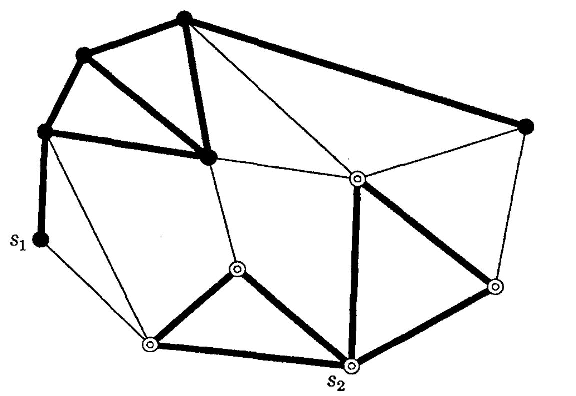

Fig. 1 An instance of the bipartition problem and a solution(thick lines depict the subgraphs induced from V_1 and V_2).

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). In this paper, ancestors and descendants of $v \in V$ include v itself. 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 its child, an edge of G joins an ancestor of the grandparent of v and a descendant of v. ■

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 child v of s_2, s_1 is adjacent to a vertex in DES(v)}
(2.1) **if** $|\mathrm{DES}(a) \cup \{s_2\}| \leq n_2$ **then**
begin {include DES(a) into V_2}
$V_2 := \mathrm{DES}(a)$;
$G_{21} := G - V_2$;
$T_{21} := T - V_2$;
$(V_1, V_2') :=$ 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, $|(\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}) \cup \{s_1\}| < n_1\}$
begin {include $\mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}$ into V_1}
$V_1 := \mathrm{DES}(s_2) - \mathrm{DES}(a) - \{s_2\}$;
$G_{22} := G - V_1$;
$T_{22} := T - V_1$;
$(V_1', V_2) :=$ 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

(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** $|\mathrm{DES}(b) \cup \{s_1\}| \le n_1$ **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}
$T_{311} := T - V_1$;
$(V_1', V_2) :=$ 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 $\mathrm{DES}(a) - \mathrm{DES}(b)$ into V_2}
$V_2 := \mathrm{DES}(a) - \mathrm{DES}(b)$;
$G_{312} := (G - V_2)/(s_2, a)$;
$T_{312} := (T - V_2)/(s_2, a)$;
$(V_1, V_2') :=$ 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** $|\mathrm{DES}(b) \cup \{s_2\}| \le n_2$ **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, $|(\mathrm{DES}(a) - \mathrm{DES}(b)) \cup \{s_2\})| < n_1\}$
begin {include $\mathrm{DES}(a) - \mathrm{DES}(b)$ 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;

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}. ■

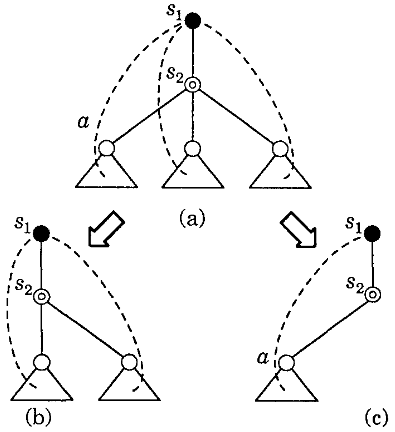

Fig. 2 (a) G , (b) G_{21} and (c) G_{22}

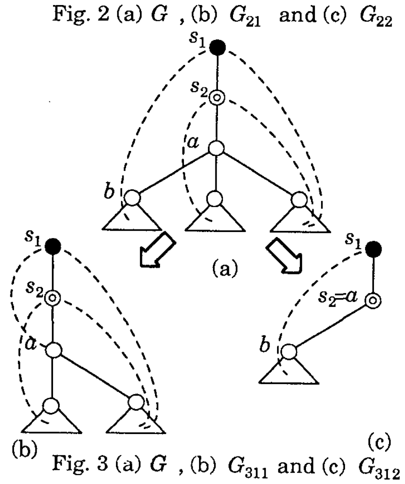

Fig. 3 (a) G , (b) G_{311} and (c) G_{312}

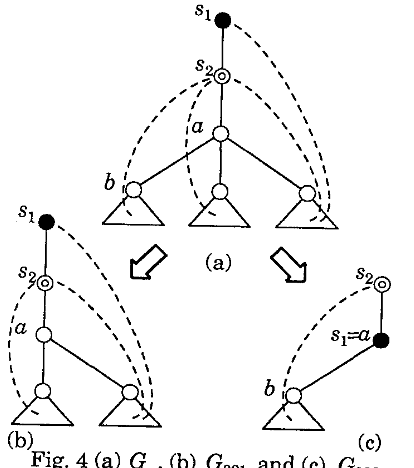

Fig. 4 (a) G , (b) G_{321} and (c) G_{322}

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

Clearly one can implement the algorithm above so that it runs in $O(m)$.

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.