Bull. Korean Math. Soc. **37** (2000), No. 1, pp. 121–125

MATCHINGS IN LINE GRAPHS

YUNSUN NAM

ABSTRACT. In this paper, we obtain an algorithm for finding a maximum matching in the line graph $L(G)$ of a graph G. The complexity of our algorithm is $O(|E|)$, where E is the edge set of G ($|E|$ is equal to the number of vertices in $L(G)$).

1. Introduction

Let $G = (V, E)$ be a graph. The *line graph* $L(G)$ *of* G is the graph whose vertex set can be put in one-to-one correspondence with the edge set of G such that two vertices of $L(G)$ are adjacent if and only if the corresponding edges of G are adjacent. A *matching* is a spanning subgraph M of G such that the degree of each vertex in M is less than or equal to one. A *maximum matching* is a matching with maximum number of edges. In this paper, we present an algorithm for finding a maximum matching in a line graph $L(G)$, which runs in $O(|E|)$. Since the edge set of G has one-to-one correspondence with the vertex set of $L(G)$, the algorithm runs in linear time in the number of vertices of $L(G)$. If we apply Micali and Vazirani's algorithm to $L(G)$, then it takes $O(\sqrt{|E|}\cdot m)$ to obtain a maximum matching, where m is the number of edges of $L(G)$. Our algorithm is relatively fast. Throughout this paper, we assume that G is connected. If G is not connected, we can obtain a maximum matching of G by applying our algorithm to each component of G.

Our algorithm deals with G, not with $L(G)$. We use the idea of Chartrand, *et al.* [1]. They show that finding a maximum matching in $L(G)$ is equivalent to decomposing G into even trails and possibly one

Received May 31, 1998.

1991 Mathematics Subject Classification: 05C70, 68R10.

Key words and phrases: matching, line graph.

The author wishes to acknowledge the financial support of the Korea Research Foundation made in the program year of 1998.

odd trail. Decompose each trail into the complete bipartite graphs $K_{2,1}$'s. An even trail is decomposed into $K_{2,1}$'s, but an odd trail is decomposed into $K_{2,1}$'s and one edge. Thus G is decomposed into $K_{2,1}$'s and possibly one edge. Let M be the set of edges in $L(G)$ joining the pair of vertices corresponding to the edges in each $K_{2,1}$. The set M is a matching in $L(G)$ of size $\lfloor \frac{|E|}{2} \rfloor$, and thus it is a maximum matching (Note that $|E|$ is equal to the number of vertices of $L(G)$). We obtain an algorithm for decomposing G into even trails and possibly one odd trail, which runs in $O(|E|)$.

In Section 2, we present our algorithm for decomposing G into even trails and possibly one odd trail. In Section 3, we prove the validity of the algorithm and compute its complexity.

2. Algorithm

In this section, we present an algorithm for decomposing G into even trails and possibly one odd trail. The algorithm consists of two steps. The first step decomposes G into trails. The second step pairs up odd trails and transforms them into even trails.

Algorithm

Step 1. [Decompose G into trails.]

1.1 Let $n \leftarrow 0$, and $E \leftarrow E(G)$.

1.2 Let $n \leftarrow n + 1$, a queue $T_n \leftarrow \emptyset$, and $k_n \leftarrow 0$.

1.3 If $E = \emptyset$, then go to Step 2. Otherwise, select one vertex v which has at least one incident edge in E and add v to T_n.

1.4 If there is an edge e in E incident upon v, then let w be the other end of e; add w to T_n; set $k_n \leftarrow k_n + 1$, $E \leftarrow E \setminus \{e\}$, and $v \leftarrow w$; go to the beginning of this Step 1.4. Otherwise, go to Step 1.2.

Step 2. [Pair up odd trails and transform them into even trails.]

2.1 If none of k_i is odd, STOP. For $i = 1, \ldots, n$, if k_i is odd, then shrink T_i into a pseudovertex. (If k_i and k_j are odd and $T_i \cap T_j \neq \emptyset$, then add an edge between pseudovertices of T_i and T_j.) Let G' be the resulting graph, and let $E' \leftarrow E(G')$. Choose an arbitrary vertex r.

2.2 [Tree growing from r.]
(1) For each vertex $v \neq r$, let $label(v) \leftarrow 0$ and $p(v) \leftarrow \emptyset$. Add r to the list of unscanned vertices; let $label(r) \leftarrow 1$ and $\ell \leftarrow 1$.
(2) If the list of unscanned vertices is empty, then go to Step 2.3. Otherwise, select one vertex v from the list.
(3) For each $(v, w) \in E'$, let $E' \leftarrow E' \setminus \{(v, w)\}$; if $label(w) = 0$, then add w to the list of unscanned vertices and let $p(w) \leftarrow v$, $label(w) \leftarrow label(v) + 1$.
(4) If $\ell < label(v)$, let $\ell \leftarrow label(v)$. Consider v to be scanned and delete v from the list of unscanned vertices. Go to (2).

2.3 For each vertex v with $label(v) = \ell$, if v has more than one pseudovertex among its children and itself then pair them up; delete from the tree the edges in the paths between the pairs of pseudovertices; for each of the pairs (T_i, T_j), EVEN(T_i, T_j).

2.4 If $\ell \neq 1$, then let $\ell \leftarrow \ell - 1$ and go to Step 2.3. Otherwise STOP.

Subroutine EVEN(T_i, T_j)

Substep 1. Let $P_{i,j} = T_i v_1 \dots v_{n-1} T_j$ be the path in the tree between T_i and T_j. Let v_0 (resp. v_n) be the real vertex in T_i (resp. T_j) which is adjacent to v_1 (resp. v_{n-1}). (Note that v_i $(i = 1, 2, \dots, n-1)$ is a real vertex.)

Substep 2. For $k = 0$ to $n - 1$, do the following:
(1) Find a trail T containing (v_k, v_{k+1}).
(2) Divide T_i into two subtrails $T_i^{(1)}$ and $T_i^{(2)}$ such that $T_i^{(1)}$ is from the beginning of T_i to v_k and $T_i^{(2)}$ is from v_k to the end of T_i.
(3) Do (2) replacing T_i by T.
(4) If the lengths of $T_i^{(1)}$ and $T^{(1)}$ have the same parity, then let $T \leftarrow T_i^{(1)} \cup T^{(1)}$ and $T_i \leftarrow T_i^{(2)} \cup T^{(2)}$. Otherwise $T \leftarrow T^{(1)} \cup T_i^{(2)}$ and $T_i \leftarrow T_i^{(1)} \cup T^{(2)}$.

Substep 3. Divide each of T_i and T_j into two subtrails in the same way as in Step 2. If the lengths of $T_i^{(1)}$ and $T_j^{(1)}$ have the same parity, then let $T_i \leftarrow T_i^{(1)} \cup T_j^{(1)}$ and $T_j \leftarrow T_i^{(2)} \cup T_j^{(2)}$. Otherwise, let $T_i \leftarrow T_i^{(1)} \cup T_j^{(2)}$ and $T_j \leftarrow T_i^{(2)} \cup T_j^{(1)}$. Return.

We illustrate the algorithm using the graph G in Figure 1. After

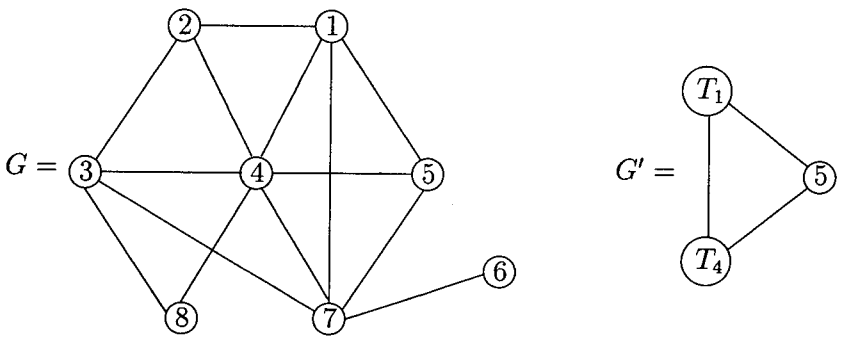

FIGURE 1. Graphs G and G'

Step 1, several different trail decompositions of G can be obtained. Let's assume that the following trail decomposition is obtained: $T_1 = [1, 2, 3, 8, 4, 2]$, $T_2 = [1, 4, 3, 7, 1, 5, 4]$, $T_3 = [5, 7, 4]$ and $T_4 = [6, 7]$. Then T_1 and T_4 are odd trails. We obtain G' in Figure 1 after Step 2.1 (shrinking T_1 and T_4 into pseudovertices). If we choose $r = 5$, then $p(5) = \emptyset$, $label(5) = 1$, $p(T_1) = p(T_4) = 5$ and $label(T_1) = label(T_4) = 2$ after Step 2.2. In Step 2.3, EVEN(T_1, T_4) is called. Then $P_{1,4} = [T_1, 5, T_7]$. Vertex 5 is adjacent to vertices 1 and 4 in T_1, and to vertex 7 in T_4. If $v_0 = 1$ is chosen, then we obtain the following trail decompostion at the end of algorithm: $T_1 = [4, 5, 7]$, $T_2 = [1, 4, 3, 7, 1]$, $T_3 = [5, 1, 2, 3, 8, 4, 2]$ and $T_4 = [6, 7, 4]$.

3. Validity and Efficiency of the Algorithm

In this section, we prove the validity of the algorithm given in Section 2 and compute its complexity. Lemma 1 shows the subroutine EVEN transforms two odd trails T_i and T_j into even trails. Using Lemma 1, the validity of the algorithm is proved (Theorem 2).

LEMMA 1. *The subroutine EVEN(T_i, T_j) transforms two odd trails T_i and T_j into even tails.*

Proof. After performing Substep 2, T has even length and T_i has odd length, and the distance between T_i and T_j is decreased by one. Thus the distance between T_i and T_j is 0 after performing Substep 2 n times, where n is the distance between T_i and T_j at the beginning of the subroutine. After performing Substep 3, both T_i and T_j have even lengths. □

Now we prove the validity of our algorithm by using the above lemma.

THEOREM 2. *At the end of the algorithm given in Section 2, G is decomposed into trails such that at most one of them has odd length.*

Proof. At the end of Step 1, G is decomposed into trails. Now we have to show that at most one trail is of odd length at the end of the algorithm. Assume that there is more than one trail of odd length. Let T_i and T_j be two trails remaining odd until the end of the algorithm. When the nearest common ancestor of T_i and T_j is scanned in Step 2.3, T_i and T_j must be paired up and sent to the subroutine EVEN(T_i, T_j). This completes the proof. □

Now we analyze the efficiency of the algorithm.

THEOREM 3. *The complexity of the algorithm given in Section 2 is $O(|E|)$.*

Proof. Step 1 takes $O(|E|)$ since it scan each edge exactly once. Since the number of trails of odd length is less than or equal to $|E|$, Step 2.1 takes $O(|E|)$. In Step 2.2, tree growing takes $O(|E|)$. For each pair (T_i, T_j), the subroutine EVEN(T_i, T_j) takes $O(n)$ where n is the distance between T_i and T_j in the tree. Since the paths between all the pairs (T_i, T_j) which are paired up in Step 2.3 are edge-disjoint, Step 2.3 takes $O(|E|)$. □

References

[1] G. Chartrand, A. D. Polimeni and M. J. Stewart, *The existence of 1-factors in line graphs, squares and total graphs*, Indag. Math. **35** (1973) 228-232.

[2] S. Micali and V. Vazirani, *An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding a maximum matching in general graphs*, in Proc. 21st Annual IEEE Symposium on Foundations of Computer Science (1980) 17-27.

MATHEMATICS DEPARTMENT, EWHA WOMANS UNIVERSITY, SEOUL 120-750, KOREA
E-mail: namy@math.ewha.ac.kr