Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)

A Linear Time Algorithm for Cubic Subgraph of Halin Graphs

Dingjun Lou and Junfu Liu
Department of Computer Science
Sun Yat-sen University
Guangzhou 510275，P. R. China
Email: issldj@mail.sysu.edu.cn

***Abstract*—In this paper, we design a linear time algorithm to determine whether a Halin graph H has a cubic subgraph H*. If H has, then the algorithm finds a cubic subgraph H* in H; otherwise the algorithm answers "No".**

Keywords-Linear time algorithm; cubic subgraph; Halin graph

I. INTRODUCTION

A Halin graph H is defined as follows: First, we embed a tree T in the plane such that each inner vertex of T has degree at least 3; then we draw a cycle C through all leaves of T to form a planar graph. Then H = T ∪ C is called a Halin graph, where T is called the characteristic tree of H and C is called the accompanying cycle of H. The simplest Halin graphs are wheels, where T has only one inner vertex and the other vertices are leaves of T. Suppose a Halin graph H is not a wheel. If w is an inner vertex of T such that all neighbours $v_1, v_2. ..., v_k$ of w except one neighbour are leaves of T, then the induced subgraph H[{w} ∪ { $v_1, v_2, \ldots, v_k$ }] is called a fan of H and w is called the center of the fan, where the induced subgraph G[S] of a graph G on a subset S of vertices in G is a subgraph of G consisting of the vertices in S and the edges of G with both ends in S.

Halin graphs were introduced by German mathematician Halin [6] as minimally 3-connected planar graphs. It can be used as a model of a network with minimum cost and fault tolerance.

A graph G is Hamiltonian if G has a cycle through all vertices of G. A graph G is 1-Hamiltonian, if G is Hamiltonian and deleting each vertex from G, the graph is still Hamiltonian. A graph G is Hamiltonian connected if, for each pair of vertices u and v, there is a Hamiltonian path P from u to v in G, where P goes through all vertices of G. A graph G is pancyclic, if G has a cycle C of length L for each integer L such that 3 ≤L ≤ |V(G)|.

Bondy [2] proves that every Halin graph H is 1-Hamiltonian. Then Bondy and Lovász [3] prove that, for each integer L such that 3 ≤ L ≤ |V(H)| except possibly for an even integer, a Halin graph H has a cycle of length L. Lou [8] proves that every Halin graph is Hamiltonian connected.

Let G be a weighted graph with each edge having a positive weight. The weight of a subgraph K of G is the sum of weights of all edges of K. The Traveling Salesman Problem is to find a Hamiltonian cycle C with minimum weight among all Hamiltonian cycles in G.

The TSP problem for a general graph is an NP—hard problem. However, Cornuejols, Naddef and Pulleyblank [4] give a linear time algorithm to solve TSP for a weighted Halin graph. Li, Lou and Lu [7] design a linear time algorithm to find a Hamiltonian path with minimum weight between each pair of vertices in a weighted Halin graph.

The Bottleneck TSP of a weighted graph G is to find a Hamiltonian cycle C with the weight of each edge of C less than or equal to a given number B. The Bottleneck TSP is also an NP—Complete problem.

Phillips, Punnen and Kabadi [11] design a linear time algorithm to solve the BTSP for a weighted Halin graph. Lou and Dou [10] design a linear time algorithm to find a Hamiltonian cycle satisfying the bottleneck restriction and having minimum weight in a weighted Halin graph.

Lou and Zhu [9] also give a linear time algorithm to solve another NPC problem, the Max-leaves Spanning Tree Problem, for Halin graphs.

The problem to determine whether a general graph G has a cubic subgraph G* such that for every vertex w of G*, $d_{G^*}(w)$ = 3 is an NPC problem (see [5]). However, for a Halin graph H, the problem to determine whether H has a cubic subgraph H* can be solved in linear time. In this paper, we design a linear time algorithm to determine whether a Halin graph H has a cubic subgraph H*. If H has, then the algorithm finds a cubic subgraph H*; otherwise the algorithm answers "No". We also prove the correctness of the algorithm and analyze the time complexity of the algorithm. The algorithm is optimal.

In [4], it is mentioned that given a Halin graph H, we can find the characteristic tree T and accompanying cycle C in O(n) time. The main idea of this algorithm is as following:

1. Find a planar embedding H' of H;
2. For each face F of H', search the boundary cycle C of F:

If all vertices on C have degree 3 and deleting the edges of C from H, the resulting graph is a tree T, then T is the characteristic tree and C is the accompanying cycle.

For terminology and notation not defined in this paper, the reader is referred to [1]

II. THE ALGORITHM

First, we give an algorithm to determine whether a Halin graph H has a cubic subgraph H* as following:

Algorithm 1:
1. Choose an inner vertex u as the root of the characteristic tree T of the input Halin graph H;
2. Do the postorder traversal of T rooted at u as following:
3. If the currently visited vertex v is the center of a fan but not u, then
(3.1) If v has at least 4 children in the current T, then H has no cubic subgraph, and the algorithm answers "No" and exits; else
(3.2) If v has precisely 3 children in the current T, then the algorithm deletes the edge between v and its father in T; else
(3.3) If v has precisely 2 children in the current T, then the algorithm keeps the edge between v and its father in T; else
4. If the currently visited vertex v is an inner vertex of the original T but not u, then
(4.1) If v has at least 4 children in the current T, then H has no cubic subgraph, and the algorithm answers "No" and exits; else
(4.2) If v has precisely 3 children in the current T, then the algorithm deletes the edge between v and its father in T; else
(4.3) If v has precisely 2 children in the current T, then the algorithm keeps the edge between v and its father in T; else
(4.4) If v has precisely 1 child in the current T, then H has no cubic subgraph, and the algorithm answers "No" and exits; else
(4.5) If v has no child in the current T, then the algorithm deletes the edge between v and its father in T and also deletes v from T; else
5. If the currently visited vertex v is u, then
(5.1) If v has precisely 3 children or no child (if no child, the algorithm deletes v from T), then H has a cubic subgraph H* =T ∪ C , where T is currently obtained by the algorithm.
(5.2) Otherwise H has no cubic subgraph, and the algorithm answers "No" and exits.

III. CORRECTNESS AND TIME COMPLEXITY

Next, we prove the correctness of Algorithm 1.

Theorem 1: If a Halin graph H has a cubic subgraph, then Algorithm 1 succeeds to find a cubic subgraph H* of H; otherwise Algorithm 1 gives answer "No".

Proof. Let u be the root of the characteristic tree T of H with the root u at the top and the tree T below. Let the level number of the lowest leaves in T be 0, the level numbers from bottom to top in T be 0, 1, 2, …, L, where L is the level number of u. If a vertex v is at level l, then all of its children are at level l−1. We proceed by induction on level number l to prove that when Algorithm 1 visits a vertex v at level l, either the degree of v becomes 3 or 0 (if 0, then v is deleted from T) or H has no cubic subgraph. We prove Claim 1 first.

Claim 1: If H has a cubic subgraph H*, then all leaves of the original T are in H*.

Since in H, every leaf of T has degree 3, if T has a leaf x not belonging to H*, then the leaf y of T adjacent to x in H has degree less than 3, and hence y does not belong to H*. If y does not belong to H*, then the leaf z of T adjacent to y in H will have degree less than 3, and hence z does not belong to H*, and so on. Then all leaves of T do not belong to H*. But deleting all leaves from T, only an isolated vertex of T remains or T has a vertex of degree 1 (a new leaf). The new leaf does not belong to H* since it has degree 1. Repeatedly deleting new leaf from T, in the end, only one isolated vertex of T remains. So H has no cubic subgraph. By the above argument, if H has a cubic subgraph, then all leaves of T are in H*.

Now we make induction on the level number l of currently visited vertex v of T.

When l = 0, the vertex v at level 0 is a leaf of the original T. When Algorithm 1 visits v, it does not do anything, and v has degree 3 in T∪C

Assume that when l ⩽ k and Algorithm 1 visits a vertex v at level l, either the degree of v becomes 3 or 0 (if 0, v is deleted from T) or H has no cubic subgraph.

If H has no cubic subgraph, according to Algorithm 1, it will not visit any vertex at level k+1 in T. Now suppose that Algorithm 1 visits a vertex v at level k+1 in T. We have 3 cases:

Case 1: v is a leaf of the original T.

Then Algorithm 1 does nothing, so v remains in T and has degree 3 in T∪C.

Case 2: v is an inner vertex of the original T but not the root u.

By induction hypothesis, all descendants of v have degree 3 or 0 (if 0, it is deleted from T) by the process of Algorithm 1. Suppose that after the process of Algorithm 1, v has p children $w_1, w_2, \ldots, w_p$.

Case (2.1): p ⩾ 4.

But in H*, v has to be of degree 3. So one edge between v and its child w_q must be deleted. By induction hypothesis, w_q and all its descendants (including some leaves of the original T) have degree 3 in current T∪C. Deleting the edge vw_q, the degree of w_q becomes less than 3. So the edges between w_q and its children must be deleted. Then the children of w_q have degree less than 3 respectively and the edges between them and their children must be deleted. Repeatedly to do this, in the end, one leaf of the original T which is a descendant of v has degree less than 3 and must be deleted from H*. By Claim 1, H has no cubic subgraph.

Case (2.2): p = 3.

In this case, Algorithm 1 deletes the edge between v and its father in T, so v has degree 3 in the current T.

Case (2.3): p = 2.

Now Algorithm 1 keeps the edge between v and its father, so v has degree 3 in the current T.

Case (2.4) p = 1.

Including the edge between v and its father, v has degree 2, so v does belong to H*, and we must delete the edge between v and its child w_q. Applying the argument in Case (2.1), w_q and all its descendants (including some leaves of the original T) must be deleted from H*, by Claim 1, H has no cubic subgraph.

Case (2.5): p = 0.

Now Algorithm 1 deletes the edge between v and its father, so v has degree 0 and is deleted from H*.

Case 3: v is the root u of T.

Suppose that after the process of Algorithm 1, v has p children in the current T.

Case (3.1): p = 3 or 0.

By induction hypothesis, the descendants of v in the original T have level number l < k+1, their degrees are either 3 or 0 (if 0, it is deleted from T) and all leaves of the original T has degree 3. So H has a cubic subgraph H* = T ∪ C, where T is currently obtained (by deleting v if v has 0 child).

Case (3.2): p ≠ 3 or 0.

Then Algorithm 1 deletes at least one edge between v and its child w_q. By the argument of Case (2.1), w_q and all its descendants (including some leaves of the original T) must be deleted from H*, by Claim 1, H does not have cubic subgraph. []

Now we analyze the time complexity of Algorithm 1.

Theorem 2: In the worst case, Algorithm 1 has time complexity O(n), where n is the number of vertices of H.

Proof. Algorithm 1 does postorder traversal of the characteristic tree T and visits each vertex once. When it visits a leaf of T, it does nothing. When it visits an inner vertex v of T, it visits at most all vertices adjacent to v once, and it needs O($d_T(v)$) time. For visiting the whole tree T, it needs O($\sum_{v\in V(T)} d_T(v)$) = O(2m(T)) = O(2(n−1)) = O(n) time, where m(T) is the number of edges of T and n = |V(T)| = |V(H)|. []

The space that Algorithm 1 needs is also O(n).

REFERENCES

[1] J. A. Bondy and U. S. R. Murty, Graph Theory with Applications, Macmillan Press, London, 1976.

[2] J. A. Bondy, “Pancyclic graphs: Recent Results”, Infinite and Finite Sets, Coll. Math. Soc. János Bolyai, vol. 10, pp. 181−187, 1975.

[3] J. A. Bondy and L. Lovász, “Lengths of cycles in Halin graphs”, Journal of Graph Theory, vol. 8, pp. 397−410, 1985.

[4] G. Cornuejols, D. Naddef and W. Pulleyblank, “Halin graphs and the traveling salesman problem”, Math. Programming, vol. 16, pp. 287−294, 1983.

[5] M. R. Garey and D. S. Johnson, Computers and Intractability−A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[6] R. Halin, “Studies on minimally n-connected graphs”, Combinatorial Mathematics and its Applications, Academic Press, London, pp. 129−136, 1971

[7] Y. Li, D. Lou and Y. Lu, “Algorithms for the optimal Hamiltonian path in Halin graphs”, Ars Combinatoria, vol. 87, pp. 235−255, 2008.

[8] Dingjun Lou, “Hamiltonian paths in Halin graphs”, Mathematica Applicata (Chinese), vol. 8, pp. 158−160, 1995.

[9] Dingjun Lou and Huiquan Zhu, “A note on max-leaves spanning tree problem in Halin graphs”, Australasian Journal of Combinatorics, vol. 29, pp. 95−97, 2004.

[10] Dingjun Lou and Hongke Dou, “A linear time algorithm for optimal bottleneck traveling salesman problem on a Halin graph”, Proc. 2011 International Conference on Computer, Communication and Information Technology (ICCCIT 2011), 2011, pp. 60−62.

[11] J. M. Phillips, P. Punnen and S. N. Kabadi, “A linear time algorithm for the bottleneck traveling salesman problem on a Halin graph,”, Information Processing Letter, vol. 67, pp. 105−110, 1998.