A Linear Time Algorithm for the Hamiltonian Problem on Distance-Hereditary Graphs

Sun-yuan Hsieh*
Department of Computer Science and Information Engineering,
National Cheng Kung University,
e-mail: hsiehsy@mail.ncku.edu.tw

Chin-Wen Ho
Department of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan
e-mail: hocw@csie.ncu.edu.tw

Tsan-sheng Hsu†
Institute of Information Science
Academia Sinica, Taipei, Taiwan
e-mail: tshsu@iis.sinica.edu.tw

Ming-Tat Ko
Institute of Information Science
Academia Sinica, Taipei, Taiwan
e-mail: mtko@iis.sinica.edu.tw

Abstract

In the literature, there are quite a few algorithms to solve problems on distance-hereditary graphs. Two well-known graphs, trees and cographs, both belong to the class of distance-hereditary graphs. In this paper, we present an $O(|V| + |E|)$-time sequential algorithm to solve the Hamiltonian problem on a distance-hereditary graph $G = (V, E)$. This algorithm is faster than the previous best result which takes $O(|V|^2)$ time.

1 Introduction

A graph is *distance-hereditary* [1, 6] if the distance stays the same between any of two vertices in every connected induced subgraph containing both (where the *distance* between two vertices is the length of a shortest path connecting them). Properties of distance-hereditary graphs are studied by many researchers [1, 2, 3, 4, 5, 6, 9, 10, 11] which resulted in algorithms to solve several interesting graph-theoretical problems on this special class of graphs.

A cycle in a graph G is called a *Hamiltonian cycle* if it contains every vertex of G exactly once. A graph is said to be *Hamiltonian* if it contains a Hamiltonian cycle. The *Hamiltonian problem* is to determine whether there exists a Hamiltonian cycle in a given graph and find one if such a cycle does exist. The previous best known result to solve the Hamiltonian problem takes $O(|V|^2)$-time [9]. In this paper, we show that the problem can be solved in $O(|V| + |E|)$-linear-time, where V (respectively, E) is the number of the vertices (respectively, edges) of the given graph.

2 Preliminaries

This paper considers finite, simple and undirected graphs $G = (V, E)$, where V and E are the vertex and edge sets of G, respectively. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *union of* G_1 *and* G_2, denoted by $G_1 \cup G_2$, is the graph $(V_1 \cup V_2, E_1 \cup E_2)$. Let $G[X]$ denote the subgraph of G induced by $X \subseteq V$.

Definition 1 [3] A graph consisting of a single vertex v is a *primitive distance-hereditary graph with the twin set* $\{v\}$. Let G_1 and G_2 be distance-hereditary graphs with the twin sets S_1 and S_2, respectively. Then,
(1) The graph obtained from G_1 and G_2 by connecting every vertex of S_1 to all vertices of S_2 is a distance-hereditary graph with the twin set $S_1 \cup S_2$;

*Part of this work was done while this author was visiting Institute of Information Science, Academia Sinica, Taipei, Taiwan.
†To whom correspondence should be addressed.

(2) The graph obtained from G_1 and G_2 by connecting every vertex of S_1 to all vertices of S_2 is a distance-hereditary graph with the twin set S_1;
(3) The union of G_1 and G_2 is a distance-hereditary graph with the twin set $S_1 \cup S_2$.

A distance-hereditary graph G is said to be formed from G_1 and G_2 by the *true twin* (respectively, *attachment*) operation, denoted by $\otimes$ (respectively, $\oplus$), if G is obtained through (1) (respectively, (2)) of Definition 1, and by the *false twin operation*, denoted by $\odot$, if G is obtained through (3) of Definition 1. Form Definition 1, a distance-hereditary graph can be represented by a binary tree form, called a *decomposition tree* D_G, in which the leaves represent $V(G)$ and the internal nodes represent three type operations mentioned above.

For a node v in D_G, let $D_G(v)$ be the subtree of D_G rooted at v, and let G_v be a subgraph of G induced by the leaves of $D_G(v)$. Also let S_v be the twin set of G_v.

Lemma 1 [3] *A decomposition tree of a distance-hereditary graph can be constructed in $O(n+m)$ sequential time.*

A *closed integer interval* is an ordered pair of integers $[t_1, t_2]$, with $t_1 \leq t_2$. The interval $[t_1, t_2]$ represents the set $\{t \in \mathbf{Z} | \ t_1 \leq t \leq t_2\}$. A *path partition* of a graph $G = (V, E)$ is a set of pairwise vertex disjoint paths such that the union of the vertices of these paths equals V. Given a distance-hereditary graph G with the twin set S, a path partition of G is said to be *crucial* if the end-vertices of each path are in S. Let $\mathcal{P}_S(G)$ denote a crucial path partition of G. Furthermore, a *crucial k-path partition of* G, $\mathcal{P}_S^k(G) = \{P_1, P_2, \ldots, P_k\}$ is a crucial path partition of G composed of exactly k paths $P_1, P_2, \ldots, P_k$. Since all path partitions under consideration throughout this paper are crucial, the subscripts S in the notations $\mathcal{P}_S(G)$ and $\mathcal{P}_S^k(G)$ are omitted if no ambiguity arises. Let $N(G) = (l_1, l_2, \ldots, l_t)$ denote the set of integers in the increasing order, i.e., $l_1 < l_2 < \cdots < l_t$, such that G has a crucial l_i-path partition. As we will show later, the elements of $N(G)$ form a segment of consecutive integers, that is, $l_{i+1} = l_i + 1$ for all $1 \leq i \leq t-1$. Thus $N(G)$ can be represented by a closed integer interval $[l_1, l_t]$. In particular, let l_1 (respectively, l_t) be the *left* (respectively, *right*) *end-point* of $N(G)$, denoted by $l(N(G))$ (respectively, $r(N(G))$). We also define $N(G) = [0, 0]$ if G does not contain any crucial path partition.

3 An Algorithm

In this section, let G be a distance-hereditary graph with the twin set S, which is formed from two distance-hereditary graphs G_1 and G_2 with the twin sets S_1 and S_2, respectively. Let $P_G[x,y]$ denote a path of G with end-vertices x and y. We allow the case of $x = y$ and in this case $P_G[x,y]$ contains exactly one vertex. For two paths P_1 and P_2 containing exactly one common vertex, let $P_1 + P_2$ denote the concatenation of P_1 and P_2. We next define three operations on two given sets of paths which are useful to our method. Let $A = \{P_{G_1}[u_1, v_1], P_{G_1}[u_2, v_2], \ldots, P_{G_1}[u_i, v_i]\}$ and $B = \{P_{G_2}[x_1, y_1], P_{G_2}[x_2, y_2], \ldots, P_{G_2}[x_j, y_j]\}$. For an integer $k \leq \textsc{Min}\{i, j\}$, define $\bowtie_k$ to be the operation on A and B, denoted by $A \bowtie_k B$, that generates a set of paths $\{P_{G_1}[u_1, v_1] + (v_1, x_1) + P_{G_2}[x_1, y_1], P_{G_1}[u_2, v_2] + (v_2, x_2) + P_{G_2}[x_2, y_2], \ldots, P_{G_1}[u_k, v_k] + (v_k, x_k) + P_{G_2}[x_k, y_k], P_{G_1}[u_{k+1}, v_{k+1}], P_{G_1}[u_{k+2}, v_{k+2}], \ldots, P_{G_1}[u_i, v_i], P_{G_2}[x_{k+1}, y_{k+1}], P_{G_2}[x_{k+2}, y_{k+2}], \ldots, P_{G_2}[x_j, y_j]\}$.

Given $\{P_{G_1}[u_1, v_1], P_{G_1}[u_2, v_2], \ldots, P_{G_1}[u_i, v_i]\}$ and $\{P_{G_2}[x_1, y_1], P_{G_2}[x_2, y_2], \ldots, P_{G_2}[x_j, y_j]\}$, let A be the one whose cardinality is larger than or equal to the other, say B. Let k be a positive integer satisfying $k < \textsc{Min}\{i, j\}$ (respectively, $k \leq \textsc{Min}\{i, j\}$) when $i = j$ (respectively, $i \neq j$). Define $\triangleright_k$ to be the operation on A and B, denoted by $A \triangleright_k B$, that generates a set of paths $\{P_{G_1}[u_1, v_1] + (v_1, x_1) + P_{G_2}[x_1, y_1] + (u_2, y_1) + P_{G_1}[u_2, v_2] + (v_2, x_2) + P_{G_2}[x_2, y_2] + (u_3, y_2) + \cdots + (v_k, x_k) + P_{G_2}[x_k, y_k] + (u_{k+1}, y_k) + P_{G_1}[u_{k+1}, v_{k+1}], P_{G_1}[u_{k+2}, v_{k+2}], P_{G_1}[u_{k+3}, v_{k+3}], \ldots, P_{G_1}[u_i, v_i], P_{G_2}[x_{k+1}, y_{k+1}], P_{G_2}[x_{k+2}, y_{k+2}], \ldots, P_{G_2}[x_j, y_j]\}$.

Given $\{P_{G_1}[u_1, v_1], P_{G_1}[u_2, v_2], \ldots, P_{G_1}[u_i, v_i]\}$ and $\{P_{G_2}[x_1, y_1], P_{G_2}[x_2, y_2], \ldots, P_{G_2}[x_j, y_j]\}$, let A be the one whose cardinality is larger than or equal to the other, say B. Let k, l be two integers satisfying $k < \textsc{Min}\{i, j\}$ and $k + l \leq \textsc{Min}\{i, j\}$ when $i = j$, or just satisfying $k + l \leq \textsc{Min}\{i, j\}$ when $i \neq j$. Define $(\triangleright_k, \bowtie_l)$ to be a pair of operations on A and B, denoted by $A(\triangleright_k, \bowtie_l)B$, that first apply $\triangleright_k$ on A and B, and then apply $\bowtie_l$ on A' and B', where $A' = \{P_{G_1}[u_1, v_1] + (v_1, x_1) + P_{G_2}[x_1, y_1] + (u_2, y_1) + P_{G_1}[u_2, v_2] + (v_2, x_2) + P_{G_2}[x_2, y_2] + (u_3, y_2) + \cdots + (v_k, x_k) + P_{G_2}[x_k, y_k] + (u_{k+1}, y_k) + P_{G_1}[u_{k+1}, v_{k+1}], P_{G_1}[u_{k+2}, v_{k+2}], P_{G_1}[u_{k+3}, v_{k+3}], \ldots, P_{G_1}[u_i, v_i]\}$ and $B' = \{P_{G_2}[x_{k+1}, y_{k+1}], P_{G_2}[x_{k+2}, y_{k+2}], \ldots, P_{G_2}[x_j, y_j]\}$.

Lemma 2 *Let $G = G_1 \otimes G_2$. A crucial q-path partition $\mathcal{P}^q(G)$ can be constructed from a crucial i-path partition $\mathcal{P}^i(G_1)$ and a crucial j-path partition $\mathcal{P}^j(G_2)$, where* $\textsc{Max}\ \{1, \textsc{Max}\{i, j\} - \textsc{Min}\{i, j\}\} \leq q \leq i + j$.

Proof: Without loss of generality, assume that $i \geq j$. The other case of $i < j$ can be shown similarly. Each desired crucial path partition can be constructed as follows. (1) $\mathcal{P}^{i+j}(G)$ can be obtained by the union of $\mathcal{P}^i(G_1)$ and $\mathcal{P}^j(G_2)$; (2) $\mathcal{P}^{i+j-k}(G)$ for $1 \leq k \leq j$, can be obtained by $\mathcal{P}^i(G_1) \bowtie_k \mathcal{P}^j(G_2)$; (3) $\mathcal{P}^{i-k}(G)$ for $1 \leq k \leq j$ (respectively, $1 \leq k \leq j-1$), can be obtained from $\mathcal{P}^i(G_1)(\triangleright_k, \bowtie_{j-k})\mathcal{P}^j(G_2)$ if $i > j$ (respectively, $i = j$). **Q.E.D.**

By constructing $\mathcal{P}^i(G_1) \triangleright_j \mathcal{P}^j(G_2)$, we have the following result.

Lemma 3 *Let $G = G_1 \oplus G_2$. A crucial path partition $\mathcal{P}^{i-j}(G)$ can be constructed from $\mathcal{P}^i(G_1)$ and $\mathcal{P}^j(G_2)$ if $i > j$.*

According to the above lemmas, we can show the following results.

Lemma 4 *Assume that $G = G_1 \otimes G_2$. Let a_i and b_i be positive integers for $i \in \{1, 2\}$. If $N(G_1) = [a_1, b_1]$ and $N(G_2) = [a_2, b_2]$, then $N(G)$ equals (1) $[1, b_1 + b_2]$ when $[a_1, b_1] \cap [a_2, b_2] \neq \emptyset$; (2) $[a_2 - b_1, b_1 + b_2]$ when $[a_1, b_1] \cap [a_2, b_2] = \emptyset$ and $a_2 > b_1$; (3) $[a_1 - b_2, b_1 + b_2]$ when $[a_1, b_1] \cap [a_2, b_2] = \emptyset$ and $a_1 > b_2$.*

Proof: By Lemma 2, a crucial path partition $\mathcal{P}^q(G)$ can be constructed from the two crucial path partitions $\mathcal{P}^i(G_1)$ and $\mathcal{P}^j(G_2)$ for $\textsc{Max}\{1, \textsc{Max}\{i, j\} - \textsc{Min}\{i, j\}\} \leq q \leq i+j$. Since $a_1 \leq i \leq b_1$ and $a_2 \leq j \leq b_2$, all possible constructions specified in the statement can be obtained.

$\mathcal{P}^q(G)$ with $q > b_1 + b_2$ does not exist in G by the following reasoning. If $i + j - |M(P^q(G))| > b_1 + b_2$, then $i + j > b_1 + b_2$. This leads to $i > b_1$ or $j > b_2$, and thus contradicts to the assumption. Therefore, the right end-point of $N(G)$ equals $b_1 + b_2$.

We next show that the left end-points specified in the statement are correct. If $[a_1, b_1] \cap [a_2, b_2] = \emptyset$ and $a_2 > b_1$, then q has the minimum value $a_2 - b_1$. The reason is explained below. Assume that $q = i+j-|M(P^q(G))| < a_2 - b_1$ holds, where $i \in [a_1, b_1]$ and $j \in [a_2, b_2]$. Clearly, $i < j$ and $0 \leq |M(\mathcal{P}^q(G))| \leq 2i$. Then, $j - i = i + j - 2i \leq i+j-|M(\mathcal{P}^q(G))| < a_2 - b_1$ and thus $0 \leq j - a_2 < i - b_1 \leq 0$, a contradiction. Therefore, $q \geq a_2 - b_1$ in this case. The case of $[a_1, b_1] \cap [a_2, b_2] = \emptyset$ and $a_1 > b_2$ can be argued similarly. **Q.E.D.**

Lemma 5 *Assume that $G = G_1 \oplus G_2$. Let a_i and b_i be positive integers for $i \in \{1, 2\}$. If $N(G_1) = [a_1, b_1]$ and $N(G_2) = [a_2, b_2]$, then $N(G)$ equals (1) $[0, 0]$ when $a_2 \geq b_1$; (2) $[1, b_1 - a_2]$ when $[a_1, b_1] \cap [a_2, b_2] \neq \emptyset$ and $a_2 < b_1$; (3) $[a_1 - b_2, b_1 - a_2]$ when $[a_1, b_1] \cap [a_2, b_2] = \emptyset$ and $a_2 < b_1$.*

Proof: From Lemma 3, $q \in N(G)$ iff there exists $i \in N(G_1)$ and $j \in N(G_2)$ such that $i - j = q$. When $a_2 \geq b_1$, since there are no $i \in N(G_1)$ and $j \in N(G_2)$ such that $i > j$. Thus, $N(G) = [0, 0]$.

We then consider the situation that $a_2 < b_1$. By Lemma 3, $\mathcal{P}^{i-j}(G)$ can be constructed from $\mathcal{P}^i(G_1)$ and $\mathcal{P}^j(G_2)$ if $i > j$. Clearly, all possible constructions specified in the statement can be obtained. The two end-points of $N(G)$ for each situation can be obtained. **Q.E.D.**

Lemma 6 *Assume that $G = G_1 \odot G_2$. Let a_i and b_i be positive integers for $i \in \{1, 2\}$. If $N(G_1) = [a_1, b_1]$ and $N(G_2) = [a_2, b_2]$, then $N(G) = [a_1 + a_2, b_1 + b_2]$.*

Theorem 1 *Let D_G be a decomposition tree of a distance-hereditary graph G. If $N(G) \neq [0, 0]$, then $N(G_v) = [a_v, b_v]$ for all nodes v in D_G, where $a_v \leq b_v \in \mathbf{Z}^+$.*

Proof: Note that $N(G_v) \neq [0, 0]$ because $N(G) \neq [0, 0]$. The theorem can be proved by induction on the height of D_G, according to Lemmas 4–6. **Q.E.D.**

According to Lemma 4 and Lemma 5, we have the following result.

Theorem 2 *Let $G = G_1 \otimes G_2$ or $G = G_1 \oplus G_2$ and let $|V(G)| \geq 3$. Then, G has a Hamiltonian cycle if and only if $N(G_1) \cap N(G_2) \neq \emptyset$ and $N(G_i) \neq [0, 0]$, $i \in \{1, 2\}$.*

Given a decomposition tree D_G as an input instance, our $O(|V| + |E|)$-time algorithm consists of two main phases. In the first phase, we compute $N(G_x)$ for each non-root internal node x of D_G based on Lemmas 4–6. During the progress of computation, if $N(G_v) = [0, 0]$ for some node v, we terminate the algorithm and exit. In the second phase, we aim at finding the set of edges to form a Hamiltonian cycle. The details are as follows.

Phase 1: Compute $N(G_v)$ for each non-root node v of D_G. Initially, let $N(G_l) = [1, 1]$

for each leaf l. By Lemmas 4–6, the desired values can be computed by a bottom-up evaluation process. During the progress of computation, if $N(G_v) = [0, 0]$ for some non-root node v, we terminate the algorithm and exit. Otherwise, all non-root nodes of D_G are associated with the desired values. Clearly, this phase takes $O(n)$ time. Before proceeding to describe the next phase, we give the following lemma which is useful in proving the correctness of the algorithm.

PHASE 2: Compute the target number $tar(v)$ for each non-root node v. Assume that y and z are the left and right children of $root(D_G)$, respectively. Select an arbitrary number $k \in N(G_y) \cap N(G_z)$ and let $tar(y) = tar(z) = k$. For each non-root internal node v with the two children u and w, select $tar(u)$ and $tar(w)$ such that $\mathcal{P}^{tar(v)}(G_v)$ can be constructed from $\mathcal{P}^{tar(u)}(G_u)$ and $\mathcal{P}^{tar(w)}(G_w)$. $tar(u)$ and $tar(w)$ can be found in $O(1)$ time. Therefore, this phase can be implemented to run in $O(n)$ time by a top-down evaluation process.

PHASE 3: Find a set of edges to form a Hamiltonian cycle of G. During a bottom-up traverse of D_G, we maintain a list $L_v = (\langle p_1, q_1\rangle, \langle p_2, q_2\rangle, \ldots, \langle p_{tar(v)}, q_{tar(v)}\rangle)$ and $M(\mathcal{P}^{tar(v)}(G_v))$ for each node v, where $\langle p_i, q_i\rangle$ represents the end-vertices of $P_{G_v}[p_i, q_i] \in \mathcal{P}^{tar(v)}(G_v)$. Initially, for each leaf l representing a primitive distance-hereditary graph $G_l = (\{v_l\}, \emptyset)$, let $L_l = (\langle v_l, v_l\rangle)$ and let $M(\mathcal{P}^1(G_l)) = \emptyset$. Also let $H_{\odot}(G)$ be the set of the internal non-$\odot$-nodes in D_G. For a node $v \in H_{\odot}(G)$ with the two children u and w, $M(\mathcal{P}^{tar(v)}(G_v))$ can be found according to Definitions **??**–**??** in $O(|S_u| \cdot |S_w|)$ time. Recall that $\mathcal{P}^{tar(u)}(G_u)$ and $\mathcal{P}^{tar(w)}(G_w)$ together with $M(\mathcal{P}^{tar(v)}(G_v))$ forms $\mathcal{P}^{tar(v)}(G_v)$. Also note that $E(G)$ equals $\cup_{v \in H_{\odot}(G)}\{S_u \times S_w |\ u \text{ and } w \text{ are the two children of } v\}$. If v is the root, we need to include some edge set, say Q, such that $\mathcal{P}^{tar(u)}(G_u)$ and $\mathcal{P}^{tar(w)}(G_w)$ together with Q form a Hamiltonian cycle. Given L_u and L_w, L_v can be constructed using the list-operations in a constant time (respectively, $O(|S_u| \cdot |S_w|)$) time if v is a $\odot$-node (respectively, non-$\odot$-node). Therefore, this phase takes $O(n + m)$ time. Note that $\cup_{v \in H_{\odot}(G)}(M(\mathcal{P}^{tar(v)}(G_v))) \cup Q$ forms a Hamiltonian cycle of G.

By Lemmas 4–6, and Theorem 2, it is not difficult to see that if the given graph is not Hamiltonian, the algorithm will detect some non-root node v associated with $N(G_v) = [0, 0]$ and then exit. Otherwise, a Hamiltonian cycle of G will be generated. By the above discussion and Lemma 1, we have the following result.

Theorem 3
The Hamiltonian problem on distance-hereditary graphs can be solved in $O(n + m)$ sequential time.

References

[1] H. J. Bandelt and H. M. Mulder, Distance-hereditary graphs, *Journal of Combinatorial Theory Series B*, 41(1):182–208, 1989.

[2] A. Brandstädt and F. F. Dragan, A linear time algorithm for connected γ-domination and Steiner tree on distance-hereditary graphs, *Networks*, 31:177–182, 1998.

[3] M. S. Chang, S. Y. Hsieh, and G. H. Chen, Dynamic programming on distance-hereditary graphs, *Proceedings of 7th International Symposium on Algorithms and Computation (ISAAC'97)*, LNCS 1350, pp. 344–353, 1997.

[4] A. D'atri and M. Moscarini, Distance-hereditary graphs, steiner trees, and connected domination, *SIAM Journal on Computing*, 17(3):521–538, 1988.

[5] P. L. Hammer and F. Maffray, Complete separable graphs, *Discrete Applied Mathematics*, 27(1):85–99, 1990.

[6] E. Howorka, A characterization of distance-hereditary graphs, *Quarterly Journal of Mathematics (Oxford)*, 28(2):417–420, 1977.

[7] R. W. Hung, S. C. Wu, and M. S. Chang, Hamiltonian cycle problem on distance-hereditary graphs, manuscript.

[8] H. Müller and F. Nicolai, Polynomial time algorithms for Hamiltonian problems on bipartite distance-hereditary graphs, *Information Processing Letters*, 46:225–230, 1993.

[9] Falk Nicolai, Hamiltonian problems on distance-hereditary graphs, Technique report, Gerhard-Mercator University, Germany, 1994.