Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Simple linear-time algorithms for counting independent sets in distance-hereditary graphs

Min-Sheng Lin

Department of Electrical Engineering, National Taipei University of Technology, Taipei 106, Taiwan, ROC

ARTICLE INFO

Article history:
Received 7 December 2016
Received in revised form 15 December 2017
Accepted 16 December 2017
Available online xxxx

Keywords:
Distance-hereditary graphs
Counting problem
Independent sets
Independent dominating sets
Independent perfect dominating sets
Linear-time algorithms

ABSTRACT

A connected graph is distance-hereditary if any two vertices have the same distance in all of its connected induced subgraphs. This paper proposes a unified method for designing linear-time algorithms for counting independent sets and their two variants, independent dominating sets and independent perfect dominating sets, in distance-hereditary graphs.

1. Introduction

A connected graph is a distance-hereditary graph if and only if the distance between any two vertices in all of its connected induced subgraphs is the same as in the original graph; alternatively, all induced paths in a distance-hereditary graph are isometric. Howorka [11] first developed distance-hereditary graphs, which have been studied extensively [1,10,9]. Studies of distance-hereditary graphs are motivated by the fact that several graph problems can be efficiently solved for distance-hereditary graphs. Numerous studies of decision problems and optimization problems for distance-hereditary graphs have been published [2,8,3,5,6,12,18,4], but few had addressed counting problems. This paper focuses on solving the problems of counting independent sets, independent dominating sets, and independent perfect dominating sets in a distance-hereditary graph.

Let $G = (V, E)$ be a graph with set of vertices V and set of edges E. An *independent set* (IS) in G is a subset D of V such that no two vertices of D are mutually adjacent. A *dominating set* in G is a subset D of V such that every vertex that is not in D is adjacent to at least one vertex in D. An *independent dominating set* (IDS) in G is a set of vertices of G that is both independent and dominating in G. An independent dominating set D is an *independent perfect dominating set* (IPDS) (or an efficient dominating set) if every vertex that is not in D is adjacent to exactly one vertex in D. Let $IS(G)$, $IDS(G)$, and $IPDS(G)$ be the collections of all ISs, IDSs, and IPDSs in G, respectively. Then, the inclusions $IPDS(G) \subseteq IDS(G) \subseteq IS(G)$ hold by definition.

Provan and Ball [20] confirmed that counting ISs is a #P-complete problem for general graphs and remains so even for bipartite graphs. Okamoto et al. [19] demonstrated that the problem of counting MISs is #P-complete for chordal graphs. Lin and Chen [15] showed that counting IPDSs remains a #P-complete problem for chordal graphs. Valiant [21] defined the class of #P problems as those that involve counting access computations for problems in NP; the class of #P-complete problems includes the hardest problems in #P. As is widely known, all exact algorithms for solving #P-complete problems have exponential time complexity so efficient exact algorithms for this class of problems are unlikely to exist. However, this

E-mail address: mslin@ee.ntut.edu.tw.

https://doi.org/10.1016/j.dam.2017.12.023
0166-218X/

complexity can be reduced by considering a restricted subclass of #P-complete problems. Some polynomial-time or linear-time algorithms for counting ISs, IDSs, or IPDSs have been found for interval graphs [13], chordal graphs [19], trapezoid graphs [14], tolerance graphs [17], triad convex bipartite graphs [15], and rooted directed path graphs [16].

The rest of this paper is organized as follows. Section 2 introduces basic definitions and notation that are used in the later sections and reviews some properties of distance-hereditary graphs. Sections 3, 4 and 5 present linear-time algorithms to count ISs, IDS, and IPDS, respectively, in a distance-hereditary graph. Finally, Section 6 provides concluding remarks.

2. Preliminary

This section presents the preliminaries on which the desired algorithms depend. Suppose $G = (V, E)$ is a graph with set of vertices V and set of edges E. Let $N(v)$ represent the neighborhood of a vertex v in G and $N[v] = \{v\} \cup N(v)$ represent the closed neighborhood of a vertex v in G. Vertices u and v are called *false twins* if $N(u) = N(v)$ and *true twins* if $N[u] = N[v]$. A *pendant vertex* is a vertex with exactly a single neighbor. Let $G[X]$ denote the subgraph of G that is induced by $X \subseteq V$.

An ordering $v_1 < v_2 < \cdots < v_n$ of V is called a *one-vertex-extension ordering* of G if v_i is a pendant vertex that is attached to, or is a (true or false) twin of, some other vertex in $G[V_i]$ for $2 \le i \le n$, where $V_i = \{v_1, v_2, \ldots, v_i\}$ and n is the number of vertices in G. It is well known that a graph is distance-hereditary if and only if it has a one-vertex-extension ordering [1,10].

Chang et al. [4] introduced the one-vertex-extension tree based on one-vertex-extension ordering. Given a one-vertex-extension ordering $v_1 < v_2 < \cdots < v_n$ of a distance-hereditary graph G, the *one-vertex-extension tree*, denoted by $ET(G)$, is obtained as follows. First, let v_1 be the root of $ET(G)$. Next, nodes are added to $ET(G)$ from v_2 to v_n. For each $2 \le j \le n$, by one-vertex-extension ordering, either vertex v_j is a pendant vertex that is attached to vertex v_i or vertices v_j and v_i are (true or false) twins in $G[V_j]$ for some vertex v_i with $i < j$. Now, let v_j be the child of node v_i in $ET(G)$. Finally, assume that the ordering of children of a node from left to right in $ET(G)$ is the same as the one-vertex-extension ordering of V. Let $[v_i, v_j]$ denote an edge of $ET(G)$, where v_i is the parent of v_j. An edge $[v_i, v_j]$ is called a *P edge* in $ET(G)$ if v_j is a pendant vertex that is attached to v_i in $G[V_j]$. An edge $[v_i, v_j]$ is called a *T edge* or *F edge* in $ET(G)$ if v_i and v_j are true twins or false twins in $G[V_j]$, respectively. Fig. 1 presents an example of a distance-hereditary graph and its one-vertex-extension tree.

Let $ET(i)$ denote the subtree of $ET(G)$ that is rooted at v_i, and let $V(i)$ be the set of all nodes in $ET(i)$. The *twin set* of node v_i, denoted by $TS(i)$, is the set of nodes in $ET(G)$ that contains v_i itself and all descendants v_j of v_i such that all edges of the path that connects v_i with v_j in $ET(G)$ are T edges or F edges.

Suppose that v_i is an internal node in $ET(G)$ whose children ordered from left to right are $v_{h1}, v_{h2}, \ldots, v_{hk}$. Then, let $ET(i, h_j)$ be a subtree of $ET(G)$ that is induced by v_i, $V(h_j)$, $V(h_{j+1})$, …, and $V(h_k)$. Let $V(i, h_j)$ be the set of all nodes in $ET(i, h_j)$ and $TS(i, h_j) = TS(i) \cap V(i, h_j)$.

Suppose that $[v_i, v_j]$ is an edge in $ET(G)$. To simplify the notation, the rest of this paper will use v_{j^*} to refer to the child of v_i right next to v_j in $ET(G)$. Notably, if v_j is the rightmost child of v_i, then $TS(i, j^*)$ contains only one node, v_i.

According to the above definitions, the following remarks are easily verified.

Remark 1. Let $[v_i, v_j]$ be an edge in $ET(G)$. The vertex set $V(i, j)$ can be partitioned into two disjoint subsets $V(i, j^*)$ and $V(j)$, and successively partitioned into four disjoint sets $TS(i, j^*)$, $V(i, j^*) \setminus TS(i, j^*)$, $TS(j)$, and $V(j) \setminus TS(j)$.

Remark 2. If $[v_i, v_j]$ is a T or F edge in $ET(G)$, then $TS(i, j)$ is the disjoint union of $TS(i, j^*)$ and $TS(j)$. If $[v_i, v_j]$ is a P edge in $ET(G)$, then $TS(i, j) = TS(i, j^*)$.

Let X and Y represent two disjoint subsets of vertices in a graph G. X and Y are said to form a *join* in G if every vertex of X is adjacent to any vertex of Y in G. X and Y are said to be *separated* in G if no vertex of X is adjacent to any vertex of Y in G.

Lemma 1 ([4]). *Suppose that $[v_i, v_j]$ is a P or T edge in $ET(G)$. $TS(j)$ and $TS(i, j^*)$ form a join in G.*

Lemma 2 ([4]). *Suppose that $[v_i, v_j]$ is an F edge in $ET(G)$. $V(j)$ and $V(i, j^*)$ are separated in G.*

Lemma 3 ([18,4]). *Suppose that $[v_i, v_j]$ is an edge in $ET(G)$. $V(j)$ and $V(i, j^*) \setminus TS(i, j^*)$ are separated in G, and $V(i, j^*)$ and $V(j) \setminus TS(j)$ are separated in G.*

The following notation will be used in the rest of this paper. Let $X \uplus Y$ denote the *disjoint union* of two sets X and Y. Given two collections of sets A and B, the operation $\otimes$ is defined by $A \otimes B = \{X \cup Y : X \in A, Y \in B\}$. Clearly, if for each $X \in A$ and each $Y \in B$, X and Y are disjoint, then $|A \otimes B| = |A| \times |B|$. To prove Lemmas 4–12, given an IS S in $G[V(i, j)]$, let $S_1 = S \cap V(i, j^*)$ and $S_2 = S \cap V(j)$. Clearly, by Remark 1, S_1 and S_2 form a partition of S. That is, $S = S_1 \uplus S_2$.

3. Counting independent sets in a distance-hereditary graph

This section provides a linear-time algorithm for counting ISs in a distance-hereditary graph. First, the four collections of ISs that are used to derive the main algorithm are defined as follows. Suppose that v_i is a node in $ET(G)$. Define the following.

$IS_a(i)$: collection of all ISs S of $G[V(i)]$ such that $S \cap TS(i) \neq \varnothing$.
$IS_b(i)$: collection of all ISs S of $G[V(i)]$ such that $S \cap TS(i) = \varnothing$.

Please cite this article in press as: M. Lin, Simple linear-time algorithms for counting independent sets in distance-hereditary graphs, Discrete Applied Mathematics (2018), https://doi.org/10.1016/j.dam.2017.12.023.