Core Path in Special Classes of Graphs

A Project Report

submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Balasubramanian S (CS04B040)

under the guidance of

Prof. C. Pandu Rangan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2008

Certificate

This is to certify that the thesis entitled **Core Path in Special Classes of Graphs** submitted by **Balasubramanian S** in partial fulfilment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering, is a bona-fide record of work carried out by him under my guidance and supervision in the Department of Computer Science and Engineering, Indian Institute of Technology, Madras.

Place : Chennai

Prof. C. Pandu Rangan

Acknowledgments

I am deeply indebted to my guide Prof. C. Pandu Rangan for introducing me to the field of Algorithms and for being a source of inspiration throughout my stay here. His clarity in teaching and his excitement in communicating the subject matter have been the driving forces in me to take up this field. I would like to specially thank him for introducing me to research and guiding my first steps. He meticulously ensures that the lab has state-of-the-art books in all topics of interest. Besides, the freedom he has given to all of us in accessing every nook and corner of the lab at any time of the day, is something I knew not before.

I am very grateful to my intern mentor Dr. Ravi Sundaram, for introducing me to numerous tools and techniques in approximation algorithms during my three months stay in Boston. Observing his mind at work while solving a problem, was an important training on thinking for me. Besides, his readiness for discussion at any point of the day, and willingness to stay till late in the night for many discussions, speak volumes about the attentive care he gave me. I would also like to thank Dr. Rajmohan Rajaraman for the many brain-storming sessions that I had with him during my stay there.

My special thanks are due to Prof. Balakrishnan for providing me the opportunity to attend two of his marvelous courses. Words are grossly insufficient to express the quality of his classes. Looking at the subject through his eyes is one of the greatest privileges one could have while learning Physics. He was so warm-hearted that I had the rare privilege of interacting with him for many hours on various topics including my graduate studies.

I thank Dr. Shankar Balachandran and Dr. Kasi Viswanathan for being my teacher and friend, and providing me valuable guidance whenever I sought it. I also thank Prof. Choudum, Dr. Arvind, Dr. Kamakoti and all my other teachers for their dedication to teaching. Dr. Arvind's classes were always amazing and I benefited a lot from his courses. I thank my Head of the Department for providing me access to excellent infrastructure in the department. I also thank my faculty advisors Prof. Janakiram and Dr. Ravindran for being supportive throughout my stay here.

I thank Harini for being an excellent friend and co-researcher to work with. I benefited a lot in my thesis work from the discussions I had with her. I thank Ashish and Arpita for being very supportive and helpful seniors in the lab. I thank Vivek and Sharmila for being very caring. I thank Vinod, Srini, Sai, Tirumala, Shayer, Chandrasekhar, Ambika, Masilamani and Varad for making my stay in the TCS lab enjoyable.

I thank Ashwin for the many stimulating discussions I had with him and also for his willingness to help me as soon as I approach him. Ragavendran's (Kurma) cheerful willingness to assist me whenever in need has been a great source of hope for me. I thank Naresh for being someone whom I always felt comfortable confiding in. I thank George for being one of my most memorable friends in IIT Madras, from whom I learnt a lot. I also thank Bandi, Arun, Vibhav, Nikhil, Setia and Santhosh for making my stay in IIT memorable. My special thanks are due to Vivekananda Study Circle, IIT Madras and Siva Teja for adding a lot of meaning to my stay in IIT Madras.

I am very fortunate to have a friend like Suresh, to whom I owe a lot for what he has given me over the past 19 years. Most importantly, I thank my parents and grandmother, for their unconditional love, sacrifice and blessings, without which I wouldn't have been here.

Abstract

We consider the problem of locating a core/median path in proper interval graphs and threshold graphs. We give a polynomial time algorithm to compute the core path in both these classes of graphs, when vertices are assigned arbitrary positive and real weights and edges are assigned unit weights. Additionally, we establish the NP-Completeness of locating the core path in both these graph classes when edges are assigned arbitrary positive weights, even when vertices are assigned unit weights. A variant of the above problem is called the conditional core problem, where the requirement is to locate the core path in presence of some already established facilities. We give a polynomial time algorithm for solving the conditional core problem in threshold graphs.

Table of Contents

CHAPTER 1

Introduction

1.1 Introduction

Structures that can be represented as graphs are ubiquitous, and many problems of practical interest can be represented by graphs. The link structure of a website could be represented by a directed graph: the vertices are the web pages available at the website and a directed edge from page A to page B exists if and only if A contains a link to B. A similar approach can be taken to problems in travel, biology, molecular chemistry, computer chip design, and many other fields. The development of algorithms to handle graphs is therefore of major interest in computer science.

Unfortunately, if the input graph is an arbitrary graph, many problems in graph theory remain intractable. We call a problem as intractable if it takes time exponential in size of the input to solve the problem.The problems which are labeled NP-Complete are those for which only exponential algorithms are known. Inspite of the current lower bounds for NP-Complete problems being polynomial functions of very small degree, there is little hope that a polynomial time algorithm exists for any of these problems. But the crucial point is many problems that are of great practical interest fall under the category of NP-Complete problems.

Though, many problems of practical relevance are NP-Complete in arbitrary graphs, it turns out that graphs that occur in real life problems are not all that arbitrary. They possess some definite structural properties because of which they often lend themselves for polynomial time algorithms for many problems. So we have that, when the input graph belongs to a restricted class of graphs, polynomial time solutions are

possible for many problems that are NP-Complete in arbitrary graphs.

Proper interval graphs and threshold graphs are two such classes of graphs. Proper interval graphs form a sub-class of interval graphs and threshold graphs form a sub-class of comparability graphs. The study of threshold graphs began in [4] with applications to the "aggregation" of linear inequalities in integer programming and set packing problems. There have been several other application areas such as the synchronization of parallel processes in [5], [8] and [14] and to cyclic scheduling in [9]. In this report, we solve a facility location problem - the core path problem in proper interval graphs and threshold graphs.

1.2 Facility location and the core path problem

Facility location on networks has been and is still a topic of great importance in fields such as transportation, communication and computer sciences. Network facility location is concerned with the optimal selection of a site in a network. It aims to find one or a set of points for placing new facilities in order to satisfy a given demand arising from a set of potential customers. A natural extension of point facility location problems are those problems in which facilities are extensive, that is those that cannot be represented as isolated points but as some dimensional structures. The criteria for optimality that have been extensively studied in literature are the *minimax* criteria in which the distance to the farthest vertex from the site is minimized and the *minisum* criteria where the total distance to the vertices from the site is minimized. The former criterion is called the center criterion and the latter is called as the median/core criterion function.

Hakimi et al. [7] first considered both minimax and minisum optimization problems related to the location of one site or of several sites in a network, the sites being either path shaped or tree shaped, the underlying graph being either a tree or an arbitrary graph etc. and thus 64 variations of the facility location problem!! Almost all the versions remain NP-Complete in arbitrary graphs. But in the recent years, the problem that has been pursued extensively is the location of a *path* shaped facility

in *trees*. There has been growing interest in this field, since in particular several applications require the location of path-shaped facility instead of a single point or a set of points. The location of pipelines, irrigation ditches, express lanes in a highway and the design of public transportation routes, can be regarded as the location of path shaped facilities.

1.2.1 Core Path

Let $G = (V, E)$ be a simple, undirected, connected, weighted (positive, real vertex and edge weights) graph. The length of a path P, denoted by $length(P)$ is defined as sum of weights of edges in P. The distance between two vertices u and v, $d(u, v)$ is defined as the length of the shortest path between the two vertices. The set of vertices of a path P is denoted by $V(P)$ and the set of vertices of $V(P) \cap X$ for any set X, is denoted by $V_X(P)$. We extend the notion of distance between a pair of vertices in a natural way to the notion of distance between a vertex and a path. The distance between a vertex v and path P is $d(v, P) = \underset{u \in V(P)}{Min}\, d(v, u)$. If $v \in V(P)$, then $d(v, P)$ is zero. The cost of a path P denoted by $d(P)$ is $\sum_{v \in V} d(v, P)w(v)$, where $w(v)$ is the weight of the vertex v.

Definition 1.2.1 *[13] The Core path or* Median *path of a graph G is a path P in G that minimizes $d(P)$.* □

Definition 1.2.2 *[12] Let $\mathcal{P}_l$ be the set of all paths of length l in G. The Core path of length l of a graph G is a path $P \in \mathcal{P}_l$ where $d(P) \leq d(P')$ for any path $P' \in \mathcal{P}_l$.* □

We consider the following two versions of the above problem for proper interval graphs and threshold graphs :

1. Finding the core path of length l in a graph, with arbitrary positive weights assigned to the edges and unit weights to vertices.
2. Finding the core path of length l in a graph, with arbitrary positive weights assigned to the vertices and unit weights to edges.

In this report, for the above two classes of graphs, we give polynomial time algorithms for version 2 and establish the NP-Completeness of version 1. Besides, we solve a variant of the above problem called the conditional core problem in threshold graphs.

1.2.2 Related Work

In [13], Morgan and Slater give the first linear time algorithm to find the core path of a tree with unit edge weights. In [11], [12] Minieka et al. extend the notion of core path with a constraint on the length of the path. In their work, they have addressed the problem of locating path or tree shaped facilities of specified length in a tree. The algorithm runs in $O(n^3)$ time. In [15], Peng and Lo extend their work by giving an $O(nlogn)$ sequential and $O(log^2(n))$ parallel algorithm using $O(n)$ processors for finding the core path of an unweighted tree with a specified length. In [3], Becker et al. give an $O(nl)$ algorithm for the unweighted case and an $O(nlog^2n)$ algorithm for the weighted case for trees. In [1], Alstrup et al. give an $O(nmin\{logn\alpha(n,n),l\})$ algorithm for finding the core path of a tree.

The problem has been analyzed so far only in trees. Recently there has been a study of the core path in grid graphs in [2].

CHAPTER 2

Proper Interval Graphs and Threshold Graphs

2.1 Proper Interval Graphs: Definitions and Characterization

Definition 2.1.1 *A graph G is an interval graph if its vertices can be put in one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G iff their corresponding intervals have nonempty intersection [6]. F is known as the intersection model of the graph.*

Let $G = (V, E)$ be an interval graph with arbitrary positive weights for vertices and unit weights for edges. For every interval I_i corresponding to a vertex v_i of the graph, we mark its left and right end points as a_i and b_i respectively.

Definition 2.1.2 *A graph G is a proper interval graph,* iff *no interval is properly contained within another interval of the graph. Any interval graph is a proper interval graph* iff $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_n$ *and* $b_1 \leq b_2 \leq b_3 \leq \cdots \leq b_n$. *For any two vertices v_i and v_j, if $a_i \leq a_j$ and $b_i \leq b_j$ then we say that $v_i < v_j$. Such an ordering of vertices of the proper interval graph is called the proper interval graph ordering.*

There are several other interesting characterizations for proper interval graphs. We state below without proof some of the other equivalent characterizations of proper interval graphs.

Theorem 2.1.3 *(Roberts) A graph is a proper interval graph if and only if it is an interval graph that does not contain an induced subgraph isomorphic to $K_{1,3}$.*

A *unit interval graph* is defined to be the intersection graph of a family of closed intervals of the real line, all of which have the same length (which is often taken to

Figure 2.1: Two graphs that are not proper interval graphs

be one). The following theorem states the relationship between a unit interval graph and a proper interval graph.

Theorem 2.1.4 *(Roberts) A graph is a proper interval graph if and only if it is a unit interval graph.*

Characterization based on consecutive ones property A *maxclique* of a graph G is defined to be any complete subgraph that is not properly contained in another complete subgraph. For any graph G with vertices indexed by $\{1, 2, \ldots, n\}$ and maxcliques indexed by $\{1, 2, \ldots, m\}$, define the *maxclique-vertex matrix M(G)* to be the $m \times n$ matrix with entry m_{ij}=1 if the i^{th} maxclique contains the j^{th} vertex and $m_{ij} = 0$ otherwise.

A matrix has the *consecutive ones property for columns* if its rows can be permuted so as to make all the 1 entries in each column consecutive. The *consecutive ones property for rows* is defined similarly.

Theorem 2.1.5 *A graph G is a proper interval graph if and only if M(G) has the consecutive ones property for both rows and columns.*

2.2 Threshold Graphs : Definitions and Characterizations

Definition 2.2.1 *A graph G is a split graph if $V(G)$ can be partitioned into $Q \cup I$, where Q induces a complete graph and I induces an edgeless graph. Thus the graph G has $|Q|(|Q|-1)/2$ edges within Q and anywhere between zero to $|Q|.|I|$ other edges between Q and I.*

Figure 2.2: Two threshold graphs having thresholds 4 and 7, respectively, using $v_{(i)}$ to denote that vertex v has weight $w_v = i$

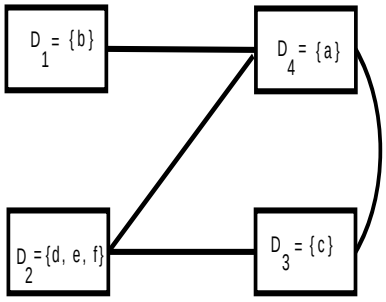

Figure 2.3: Another view of the graph on the right in 2.2

Threshold graphs are special split graphs that were introduced by Chvátal & Hammer and have been extensively studied since that time.

Definition 2.2.2 *For each vertex v of a graph G, let w_v denote a nonnegative real number, the weight of v. A graph G is a threshold graph if there is an assignment of weights to the vertices of G and a nonnegative real number t, the threshold, such that, for every $X \subseteq V(G)$, X is an independent set if and only if $\sum_{v \in X} w_v \leq t$ - in other words, if weights can be assigned to vertices of G so that a subset of vertices is independent if and only if the total weight of the set is no greater than a certain constant threshold.*

The notion of degree partition of a vertex set is crucial to the understanding of threshold graphs. Let G be a graph whose nonisolated vertices have the distinct degrees $\delta_1 < \delta_2 < \delta_3 < \cdots < \delta_m$. Set $\delta_0 = 0$ and $\delta_{m+1} = |V| - 1$, and let D_i be the set of all vertices having degree δ_i for $i = 0, 1, 2, \ldots, m$. The sequence $D_0, D_1, \ldots, D_m$ is

called the *degree partition* of G.

Example 1 *The threshold graph on the left in Figure 2.2 has $m = 2$ with $\delta_1 = 1$, $\delta_2 = 4$, $D_0 = \emptyset$, $D_1 = \{a, b, d, e\}$, and $D_2 = \{c\}$. The threshold graph on the right has $m = 4$ with $D_0 = \emptyset$, $D_1 = \{b\}$, $D_2 = \{d, e, f\}$, $D_3 = \{c\}$, and $D_4 = \{a\}$.*

Figure 2.3 shows another view of the graph on the right in Figure 2.2, with its vertices now grouped into "cells" corresponding to the degree partition. The D_i's in the left column represent independent sets, the D_i's in the right column represent complete subgraphs, and a line between cells D_i and D_j means that every vertex in D_i is adjacent to every vertex in D_j.

The graph in Example 1 is a split graph, with the union of the cells in Figure 2.3 forming the independent set I and the union of those on the right including the complete subgraph Q. It can also be seen that the open neighborhood of every vertex in the left column is contained in the open neighborhood of every vertex below it; similarly, the closed neighborhoods of every vertex in the right column is contained in the closed neighborhood of every vertex above it.

Theorem 2.2.3 *(Chvátal and Hammer) Let $G = (V, E)$ be a graph with degree partition $D_0, D_1, \ldots, D_m$. Then the following statements are equivalent:*

1. *G is a threshold graph;*
2. *for $x \in D_i$ and $y \in D_j$, $xy \in E$ if and only if $i + j > m$;*
3. *there exist nonnegative integer weights w_v and threshold t such that, for distinct vertices u and v, $uv \in E$ if and only if $w_u + w_v > t$*
4. *G does not contain P_4, C_4, or $2K_2$ as an induced subgraph;*
5. *G is a split graph where the open neighborhoods of the vertices of the independent set I can be nested with respect to set inclusion;*
6. *G can be obtained from K_1 by recursively adding either an isolated vertex or a vertex adjacent to every existing vertex.*

It follows from condition (4) of Theorem 2.2.3 that the complement of a threshold

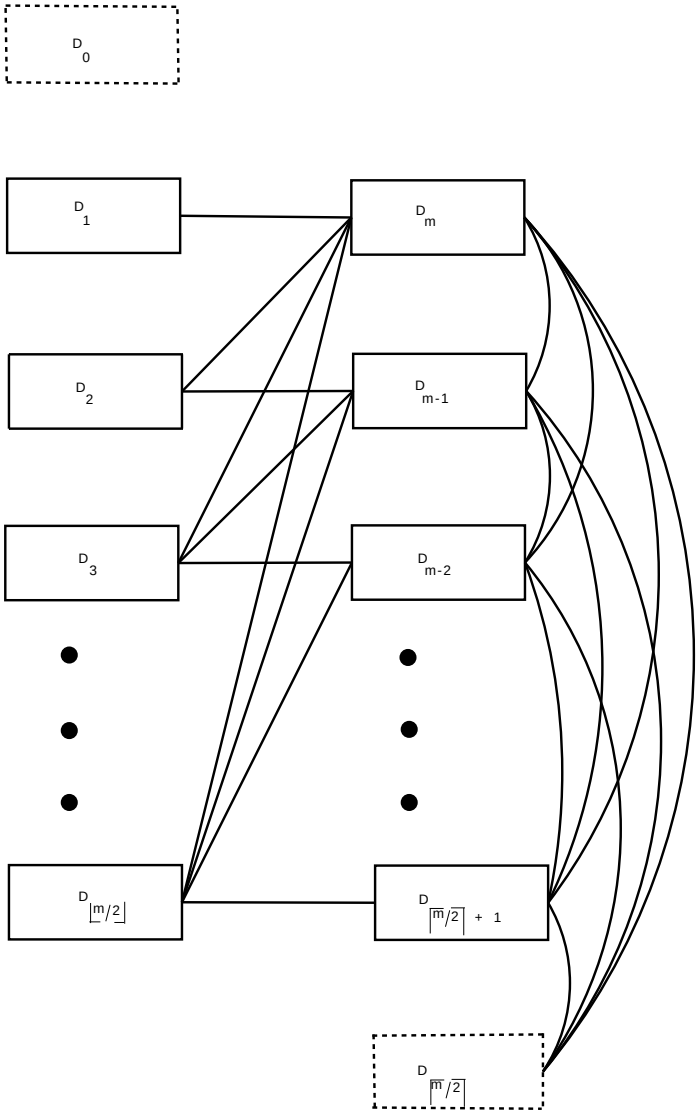

Figure 2.4: The structure of a typical threshold graph

graph is a threshold graph. This is because we know that P_4 is self-complementary and that C_4 and $2K_2$ are complements of each other.

2.3 Threshold Graphs as Intersection Graphs

Let G be an interval graph. A *threshold interval representation* for G is an interval representation for G that consists of a family of intervals $\{J_{v_i}\}$ such that J_{v_i} is either the interval $[0, r_i]$ or the trivial interval $[s_i, s_i]$ where $s_j \neq s_k$ for $j \neq k$ and where $s_j \neq r_k$ for all j and k. The following is from [10]

Theorem 2.3.1 *A graph is a threshold graph if and only if it is an interval graph with a threshold interval representation.*

Proof: Assume G is an interval graph with a threshold interval representation. Then the nontrivial intervals in the representation correspond to vertices that induce a maxclique Q of G, the trivial intervals correspond to vertices that form an independent set i in G, and the neighborhoods of vertices in I are nested in the order in which their representing trivial intervals appear along the real line. Therefore, G is a threshold graph by part (5) of Theorem 2.2.3.

Conversely, suppose that G is a threshold graph. By part (5) of Theorem 5.1, G is a split graph with $V(G)$ partitioned into the complete subgraph Q and the independent set I with the neighborhoods of I nested. Assign each vertex v in Q an interval $J_v =$ [0,r_v] such that $N[u] \subseteq N[v]$ if and only if $r_u \leq r_v$ for all u, $v \in Q$. For each $w \in I$, let $r(w) = max\{r_v : wv \in E(G)\}$. Each such w can be assigned a trivial interval J_w that is a small distance ϵ_w to the left of $r(w)$ in such a way that the intervals form a threshold interval representation for G. □

Example 2 *The threshold graph on the left in Figure* 2.2 *could receive the threshold interval representation determined by* $0 < s_a < s_b < s_d < s_e < r_c$. *The threshold graph on the right could receive the threshold interval representation determined by* $o < s_d < s_e < s_f < r_c < s_b < r_a$.

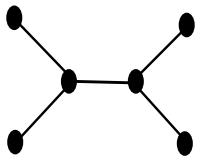

Figure 2.5: An interval graph that requires only two lengths of intervals, yet is not a threshold graph.

There are threshold graphs that are unit interval graphs (K_n for example) and others that are not unit interval graphs ($K_{1,3}$ for example). However a threshold graph will never require more than two distinct lengths of intervals in its interval representation.

Theorem 2.3.2 *Every threshold graph has an interval representation whose intervals have at most two distinct lengths.*

Proof: Let G be a threshold graph with $V(G)$ partitioned into the maxclique Q and independent set I, and suppose $\{J_v\}$ is a threshold interval representation as constructed in the proof of Theorem 2.3.1. By that construction, there exists a $z \in Q$ such that $r_x \leq r_z$ for all $x \in Q$. For each $u \in Q$ assign the interval $J'_u = [r_u - r_z, r_u]$, while for each $w \in I$ let $J'_w = J_w$. Then J'_v is an interval representation for G using only the two lengths r_z and 0. □

The converse to Theorem 2.3.2 fails since the nonthreshold graph P_4 is a unit interval graph.

CHAPTER 3

Core path of a Proper interval graph with vertex weights

3.1 Introduction

In this chapter, we provide a polynomial time algorithm for solving the core path problem in proper interval graphs in the case when arbitrary positive weights are assigned to vertices and unit weights are assigned to edges. Additionally, we prove the NP-Completeness in solving the problem when arbitrary positive weights are assigned to edges, even when unit weights are assigned to vertices.

Let $G = (V, E)$ be an interval graph with arbitrary positive weights for vertices and unit weights for edges. Recalling the notation from chapter 2, we note that for every interval I_i corresponding to a vertex v_i of the graph, we mark its left and right end points as a_i and b_i respectively.

From now on, we consider a proper interval graph G whose vertices $V = \{v_1, v_2, \ldots, v_n\}$ are labeled such that $v_i < v_j$ *iff* $i < j$.

Remark 3.1.1 *For a proper interval graph, the core path (when there is no constraint on the length of the path) is the Hamiltonian path. The path $P = v_1v_2v_3 \ldots v_n$ is a Hamiltonian path and it is also a core path with $d(P) = 0$.*

Our algorithm finds the core path with a specified length. Since we have assigned unit weights to all the edges, the length of the path is just the number of edges in the path. *In this section alone, for technical considerations, we hold the length of the path = 1 + number of edges in the path i.e. the number of vertices in the path*

3.2 NP-Completeness of Core path problem in edge-weighted proper interval graphs

PROBLEM A

INSTANCE: A complete graph, L, K

QUESTION: Is there a path P of length utmost L satisfying $d(P) \leq K$?

Theorem 3.2.1 *Problem A is NP-Complete.*

Proof: The problem of finding a Hamiltonian path in an arbitrary graph $G = (V, E)$ is known to be NP-Complete. We construct a complete graph $G' = (V', E')$ from G as follows : $V' = V \cup a$ and $E' = E \cup E_{new}$ where $E_{new} = \{(v_i, v_j) | (v_i, v_j) \notin E\} \cup \{(a, v) | v \in V' - a\}$. In G', we assign a weight of $|V|$ to all edges $e \in E_{new}$ and unit weights to all edges $e \in E' - E_{new}$. We assign unit weights to all the vertices. We set $L = |V| - 1$ and $K = |V|$

We now have to establish that G has a Hamiltonian path *iff* G' has a path of length utmost L and cost utmost K.

Suppose G had a Hamiltonian path P, then the path P in G' will have a length of $|V|$ - 1 and $d(P) = |V|$.

Suppose G' had a core path of length utmost $|V|-1$ and $d(P) \leq |V|$, then it will imply that no edge $e \in E_{new}$ was selected in the core path for otherwise, $length(P) \geq |V|$. This means that the path comprises only of the vertices from $X \cup Y$ and edges from E. Also the path should have been of length $|V|$ - 1, else $d(P) \geq |V| + 1$ ($|V|$ due to vertex a, and at least 1 due to vertices in V). Therefore we have a path of length $|V|$ - 1 comprising only of vertices from V and edges from E, which is nothing but a Hamiltonian path in the graph G. □

Corollary 3.2.2 *Since every complete graph is a proper interval graph, it follows that the problem is NP-Complete for proper interval graphs.*

3.3 Preliminaries

We call a path $P = x_1 x_2 \dots x_k$ of G *ordered, iff* $x_1 < x_2 < \cdots < x_k$. We say that an ordered path P *begins* at v_i if $v_i < v, \forall v \in V(P)$ and *ends* at v_i if $v < v_i, \forall v \in V(P)$. We use $\leq$ to denote the reflexive closure version of the relation $<$ between the vertices.

Lemma 3.3.1 *In a proper interval graph, for every path P, there exists an ordered path Q, such that $V(P) = V(Q)$.*

Proof: Let $P = x_1 x_2 \dots x_k$ be an arbitrary path. By induction hypothesis, assume that the *Lemma* is true for all paths of length utmost $k-1$. Hence there is a path $P' = y_1 y_2 \dots y_{k-1}$, such that $y_1 < y_2 < \cdots < y_{k-1}$ and $V(P') = \{x_1, x_2, \dots, x_{k-1}\}$.
Case 1: $y_{k-1} < x_k$
$P'x_k$ is the required path Q, where $P'x_k$ is the path obtained by concatenation of P' and x_k. As P' *ends* at y_{k-1}, we have that $x_{k-1} \leq y_{k-1}$. But we also know that $(x_{k-1}, x_k) \in P$ and hence $(x_{k-1}, x_k) \in E$. Since $x_{k-1} \leq y_{k-1} < x_k$ and $(x_{k-1}, x_k) \in E$, it follows from proper interval graph ordering property that $(y_{k-1}, x_k) \in E$
Case 2: $x_k < y_1$
By a similar argument as for *Case 1*, $x_k P'$ is the required path Q.
Case 3: $y_{i-1} < x_k < y_i$ for some $2 \leq i \leq k-1$
Since $(y_{i-1}, y_i) \in E$, $(y_{i-1}x_k)$, $(x_k y_i) \in E$ due to proper interval graph ordering. Hence $y_1 y_2 \dots y_{i-1} x_k y_i \dots y_{k-1}$ is the required path Q. □

In the above lemma, since $V(P)$=$V(Q)$, it follows that $d(P)$=$d(Q)$. So, for every path there is an ordered path with the same cost and same set of vertices. Hence from now on, we consider only the set of ordered paths. For every vertex $v_i \in V(G)$, we find the ordered path of length l of minimum cost that *ends* at v_i. Such a path for v_i is denoted by $P^l_{v_i}$. It is now easy to see that the core path of the graph is the path which has minimum cost among $P^l_{v_i}$ $\forall v_i$.
Let G_i be the graph induced by the vertices $\{v_1, v_2, \dots v_i\}$ in G and let G'_i be the graph induced by the vertices $\{v_i, v_{i+1}, \dots, v_n\}$ in G.

Remark 3.3.2 *Any ordered path that* ends *at v_i cannot contain a vertex v such that $v_i < v$. Hence to compute $d(P_{v_i})$, it is sufficient to consider the ordered paths in G_i.*

3.4 Algorithm

For every ordered path P that *ends* at v_i, we define $d_i(P)$ to be the cost of the path P in the graph G_i. Similarly for an ordered path P that *begins* at v_i, we define $d'_i(P)$ to be the cost of the path P in the graph G'_i. We have that $P^1_{v_i} = v_i$; Also, $d_1(P^1_{v_1}) = 0$;

Lemma 3.4.1 *$d_i(P^1_{v_i}) = d_j(P^1_{v_j}) + w(v_1) + w(v_2) + \ldots + w(v_{i-1})$, where v_j is such that $(v_j,v_i) \in E$ and $(v_k,v_i) \notin E$ $\forall k < j$.*

Proof: All the vertices from v_1 to v_{j-1} incur a total cost of $d_j(P^1_{v_j}) + w(v_1) + w(v_2) + \ldots + w(v_{j-1})$. The sum $w(v_1) + w(v_2) + \ldots + w(v_{j-1})$ is due to cost incurred in traveling the extra edge (v_j, v_i) for each vertex in $\{v_1,v_2, \ldots, v_{j-1}\}$. Vertices $v_j,v_{j+1}, \ldots, v_{i-1}$ are adjacent to v_i due to the property of proper interval graphs. Hence they contribute a cost of $w(v_j) + \ldots + w(v_{i-1})$. The total cost $d_i(P^1_{v_i}) = d_j(P^1_{v_j}) + w(v_1) + w(v_2) + \ldots + w(v_{i-1})$. □

From the above *Lemma*, it follows that the computation of $d_i(P^1_{v_i})$ $\forall i$ takes $O(n)$ time.

Lemma 3.4.2 *$\forall v_i \in V$ and $\forall 2 \leq k \leq |V|$,*

$$d_i(P^k_{v_i}) = \begin{cases} \underset{\forall v_j | v_j < v_i, (v_j,v_i) \in E}{Min} \{d_i(P^{k-1}_{v_j} v_i)\} & \text{if such } v_j\text{'s exist} \\ \infty & \text{otherwise} \end{cases}$$

Also $d_i(P^{k-1}_{v_j} v_i) = d_j(P^{k-1}_{v_j}) + w(v_{j+1}) + w(v_{j+2}) \ldots + w(v_{i-1})$.

Proof: Proof of the first statement follows from the definition of $d_i(P)$. In the second statement of *lemma*, $d_j(P^{k-1}_{v_j})$ is the sum of the costs due to vertices $v_1,v_2 \ldots v_j$. Vertices v_{j+1}, $v_{j+2} \ldots v_{i-1}$ contribute the cost equal to the sum of their respective weights as they are at a distance one from $P^{k-1}_{v_j} v_i$ and hence the proof. □

3.4.1 Finding the core path:

We shall initially compute $d_i(P^l_{v_i})$ $\forall v_i$ which is the cost of $P^l_{v_i}$ in G_i using *Algorithm* 1. But we need their costs with respect to graph G. For a path $P^l_{v_i}$, we have not added the costs due to vertices v_{i+1}, v_{i+2}, ..., v_n. Now, it is easy to see that $d(P^l_{v_i})$ $= d_i(P^l_{v_i}) + d'_i(P^1_{v_i})$. Therefore we require $d'_i(P^1_{v_i})$ for each $v_i \in V$. But $d'_i(P^1_{v_i})$ can be calculated in $O(n)$ time $\forall v \in V$ by following a procedure analogous to the one specified in *Lemma* 3.4.1. After computing $d(P^l_{v_i})$ $\forall 1 \leq i \leq n$, we find the minimum of $d(P^l_{v_i})$ $\forall 1 \leq i \leq n$. The path corresponding to this minimum cost is the core path of the graph.

Algorithm 1 : Algorithm to compute the core path of length l in a proper interval graph

---

Require: A proper interval graph $G = (V, E)$ with proper interval ordering v_1,v_2,$v_3 \ldots v_n$.

Compute $d_i(P^1_{v_i})$ $\forall v_i$ as discussed in *Lemma* 3.4.1.

Compute $d'_i(P^1_{v_i})$ $\forall v_i$ similar to the procedure discussed in *Lemma* 3.4.1.

for *length*=2 to l **do**

  for each vertex v_i, $\forall 1 \leq i \leq n$. **do**

$$d_i(P^{length}_{v_i}) = \begin{cases} \underset{\forall v_j | v_j < v_i, (v_j v_i) \in E}{Min} \{d_i(P^{length-1}_{v_j} v_i)\} & \text{if such } v_j\text{'s exist} \\ \infty & \text{otherwise} \end{cases}$$

  end for

end for

for $i = 1$ to n **do**

  $d(P^l_{v_i}) = d_i(P^l_{v_i}) + d'_i(P^1_{v_i})$.

end for

Compute $Mincost = \underset{1 \leq i \leq n}{Min} d(P^l_{v_i})$

Output the path corresponding to $Mincost$

Theorem 3.4.3 *Algorithm 1 outputs the core path of length l in a proper interval graph in $O(l|E|)$ time.*

Proof: The algorithm uses *Lemma* 3.4.2 to compute the values of $d(P^l_{v_i})$ and then finds the minimum cost path by finding $\underset{1\leq i\leq n}{Min} d(P^l_{v_i})$. Hence the correctness follows.

Time Complexity: The algorithm performs utmost $deg(v_i)$ computations to compute the value of $d_i(P^{length}_{v_i})$ for each i and each value of $length$. Therefore the algorithm does $l(deg(v_1) + deg(v_2) + \cdots + deg(v_n)) = O(l|E|)$ computations. Additionally it takes $O(|V|)$ time to find the minimum among $|V|$ elements. Hence, the total time taken is $O(|V| + l|E|) = O(l|E|)$ for connected graphs. □

CHAPTER 4

Core Path in Threshold Graphs

In this chapter, we solve the core-path problem in threshold graphs by providing a polynomial time algorithm that runs in $O(l|E|)$ time. We assume that G has arbitrarily positive vertex weights and unit edge weights. As we saw in chapter 2, every threshold graph is a split graph and hence $V(G)$ can be partitioned such that $V = X + Y$ where X is an independent set and the graph induced by Y is a clique. Let $x_1, x_2, \ldots, x_{|X|}$ be the vertices of X and $y_1, y_2, \ldots, y_{|Y|}$ be the vertices of Y, such that $deg(x_1) \leq deg(x_2) \leq \cdots \leq deg(x_{|X|}) \leq deg(y_{|Y|}) \leq deg(y_{|Y|-1}) \leq \cdots \leq deg(y_1)$. We define the following ordering on the vertices of X and Y: $x_1 < x_2 < \cdots < x_{|X|}$ and $y_1 < y_2 < \cdots < y_{|Y|}$.

The set of vertices adjacent to a vertex u is denoted by $N(u)$. $N[u] = N(u) \cup \{u\}$, is the closed neighborhood of u. By *Definition*, $N(x_1) \subseteq N(x_2) \subseteq \ldots N(x_{|X|})$ and $N[y_1] \supseteq N[y_2] \supseteq \ldots N[y_{|Y|}]$. Let $L(x_i)$ and $R(x_i)$ be the vertices with smallest and largest index in $N(x_i)$ for any $x_i \in X$ and let $L(y_i)$ and $R(y_i)$ be the vertices with smallest and largest index in $N(y_i) \cap X$ for any $y_i \in Y$ in the order defined above.

4.1 NP-Completeness of core path problem in edge-weighted threshold graphs

PROBLEM B

INSTANCE: A complete split graph, L, K

QUESTION: Is there a path P of length utmost L satisfying $d(P) \leq K$?

Theorem 4.1.1 *Problem B is NP-Complete.*

Proof: We will prove the NP-Completeness of problem B by reducing it from the problem of Hamiltonian path in bipartite graphs. The problem of finding a Hamiltonian path in a bipartite graph $G = (X, Y, E)$ is known to be NP-Complete [6]. We construct a complete split graph $G' = (X', Y', E')$ from G as follows : $X' = X \cup a$, $Y' = Y \cup b$ and $E' = E \cup E_{new}$ where $E_{new} = \{(x_i, y_j) | (x_i, y_j) \notin E, x_i \in X, y_j \in Y\}$ $\cup \{(a, y) | y \in Y'\} \cup \{(b, x) | x \in X'\} \cup \{(u, v) | u, v \in Y'\}$. In G', we assign a weight of $|V|$ to all edges $e \in E_{new}$ and unit weights to all edges $e \in E' - E_{new}$. We assign unit weights to all the vertices. We set $L = |V| - 1$ and $K = 2|V|$.

We now have to establish that G has a Hamiltonian path *iff* G' has a path of length utmost L and cost utmost K.
Suppose G had a Hamiltonian path P, then the path P in G' will have a length of $|V|$ - 1 and $d(P) = 2|V|$.
Suppose G' had a core path of length utmost $|V| - 1$ and $d(P) \leq 2|V|$, then it will imply that no edge $e \in E_{new}$ was selected in the core path for otherwise, $length(P) \geq |V|$. This means that the path comprises only of the vertices from $X \cup Y$ and edges from E. Also the path should have been of length $|V|$ - 1 else $d(P) \geq 2|V| + 1$ ($2|V|$ due to vertices a and b, and at least 1 due to vertices in V). Therefore we have a path of length $|V|$ - 1 comprising only of vertices from $X \cup Y$ and edges from E, which is nothing but a Hamiltonian path in the graph G. □

Corollary 4.1.2 *We have proved the NP-Completeness of the core-path problem in complete split graphs. Since every complete split graph is a threshold graph, it follows that the problem is NP-Complete for threshold graphs.*

4.2 Ordered Paths

A path P is said to be *ordered iff* $P = x_{a_1}y_{b_1}x_{a_2}y_{b_2}\ldots x_{a_k}(y_{b_k}y_{b_{k+1}}y_{b_{k+2}}\ldots y_{b_r}$ if they exist) such that $x_{a_1} < x_{a_2} < \cdots < x_{a_k}$, $y_{b_1} < y_{b_2} < \cdots < y_{b_r}$ or $P = y_{b_1}x_{a_1}y_{b_2}x_{a_2}\ldots y_{b_k}x_{a_k}(y_{b_k+1}y_{b_{k+2}}y_{b_{k+3}}\ldots y_{b_r}$ if they exist) such that $x_{a_1} < x_{a_2} < \cdots < x_{a_k}$ and $y_{b_1} < y_{b_2} < \cdots < y_{b_r}$. For every ordered path P, let $\alpha_r(P)$ denote the path obtained by taking the *first* r edges of the ordered path P. In case, the path does not contain r edges, then $\alpha_r(P) = \perp$. We define $d(\perp) = \infty$. Also, $d(\alpha_r(P)v) = \infty$, when $\alpha_r(P) = \perp$, where $\alpha_r(P)v$ denotes the concatenation of $\alpha_r(P)$ and vertex v. The following lemma relates every path with an ordered path with same vertex set

Lemma 4.2.1 *In a threshold graph, for every path P, there exists an ordered path Q such that $V(P)$=$V(Q)$.*

Proof: Let $V_X(P) = \{x_{a_1}, x_{a_2}, \ldots, x_{a_k}\}$ and $V_Y(P) = \{y_{b_1}, y_{b_2}, \ldots, y_{b_r}\}$ such that $x_{a_1} < x_{a_2} < \cdots < x_{a_k}$ and $y_{b_1} < y_{b_2} < \cdots < y_{b_r}$. Note that $k \leq r + 1$.
First we will prove using induction that,
if $k = r + 1$, then $(x_{a_i}, y_{b_i}) \in E\ \forall 1 \leq i \leq k - 1$ and
if $k < r + 1$, then $(x_{a_i}, y_{b_i}) \in E\ \forall 1 \leq i \leq k$.
Base case: i = 1
Suppose $(x_{a_1}, y_{b_1}) \notin E$, then because $x_{a_1} \in P$, we have that $(x_{a_1}, y_{b_j}) \in P$ for $j > 1$. But this will mean that $(x_{a_1}, y_{b_1}) \in E$ due to the threshold graph property, which is a contradiction.
By induction hypothesis, we will assume that $(x_{a_i}, y_{b_i}) \in E\ \forall 1 \leq i \leq t$ - 1.
We should now prove that $(x_{a_t}, y_{b_t}) \in E$. Assume on the contrary that $(x_{a_t}, y_{b_t}) \notin E$. This will imply that $(x_{a_t}, y_{b_j}) \notin E\ \forall j \geq t$ and $(x_{a_i}, y_{b_t}) \notin E\ \forall 1 \leq i \leq t$. So all the edges of the path P incident on $\{x_{a_1}, x_{a_2}, \ldots, x_{a_t}\}$ should be incident on $\{y_{b_1}, y_{b_2}, \ldots, y_{b_{t-1}}\}$. Now, we know that, even if both the single degree vertices of the path P are in X, the degree sum of $x_{a_1}, x_{a_2}, \ldots, x_{a_t}$ for the edges in path P is $2t$ - 2.
But $\mid (y_{b_j}, x_{a_i}) \in P, 1 \leq j \leq t-1, 1 \leq i \leq t \mid \leq 2(t-1)-1 = 2t$-3, i.e. the maximum number of edges between $\{y_{b_1}, y_{b_2}, \ldots, y_{b_{t-1}}\}$ and $\{x_{a_1}, x_{a_2}, \ldots, x_{a_t}\}$ in the path P

is 2(t - 1) - 1 = 2t - 3. The 2(t - 1) term is due to the t - 1 vertices y_{b_1}, y_{b_2}, ..., $y_{b_{t-1}}$ each of degree two. We have subtracted 1 for the following reason. For the path to be connected, at least one of the vertices in $\{x_{a_{t+1}}, x_{a_{t+2}}, \ldots, x_{a_k}\} \cup \{y_{b_t}, y_{b_{t+1}} \ldots y_{b_r}\}$ should be adjacent to at least one of the vertices in $\{x_{a_1}, x_{a_2}, \ldots x_{a_t}\} \cup \{y_{b_1}, y_{b_2}, \ldots y_{b_{t-1}}\}$. But since we have assumed that $(x_{a_t}, y_{b_t}) \notin E$, we have that $(x_{a_i}, y_{b_j}) \notin E \ \forall 1 \leq i \leq t$ and $j \geq t$. Hence at least one of the vertices of $\{y_{b_1}, y_{b_2}, \ldots, y_{b_{t-1}}\}$ is adjacent to at least one of the vertices in $\{x_{a_{t+1}}, x_{a_{t+2}}, \ldots, x_{a_k}\} \cup \{y_{b_t}, \ldots, y_{b_r}\}$ in the path P. So we have subtracted one from 2(t - 1). As 2t - 2 > 2t - 3 the degree sum does not match, leading to a contradiction. Therefore $(x_{a_t}, y_{b_t}) \in E$.

Due to the inclusion property, $(x_{a_i}, y_{b_i}) \in E$ will imply that $(x_{a_{i+1}}, y_{b_i}) \in E \ \forall 1 \leq i \leq k$ - 1. This will prove that there exists an ordered path $Q = x_{a_1} y_{b_1} x_{a_2} y_{b_2} \ldots x_{a_k} (y_{b_k} y_{b_{k+1}} \ldots y_{b_r}$ if they exist) in the graph. □

4.3 Finding the cost of any ordered path:

Remark 4.3.1 *Given any ordered path P, $V_X(P) = \{x_{a_1}, x_{a_2}, \ldots, x_{a_k}\}$ and $V_Y(P) = \{y_{b_1}, y_{b_2}, \ldots, y_{b_r}\}$ where $k \leq r + 1$ and $x_{a_1} < x_{a_2} < \cdots < x_{a_k}$, $y_{b_1} < y_{b_2} < \cdots < y_{b_r}$, we have that,*

1. *For all vertices $y_i \in Y$-$V_Y(P)$, $d(y_i, P) = 1$ as Y is a clique.*
2. *For all vertices $x_i \in X$-$V_X(P)$, if $x_{a_1} < x_i$, then $d(x_i, P)$=1 and if $x_i < x_{a_1}$, then $d(x_i, P) \leq 2$ due to the existence of the path $x_i y_1 x_{a_1}$ (as y_1 is a universal vertex). In the latter case if $L(y_{b_1}) \leq x_i < x_{a_1}$, then $d(x_i, P)$=1. Otherwise $d(x_i, P)$=2.*
3. *If $V_X(P) = \emptyset$, then $\forall x_i$ satisfying $x_1 \leq x_i < L(y_{b_1})$, $d(x_i, P) = 2$ and $\forall x_i$ satisfying $L(y_{b_1}) \leq x_i$, $d(x_i, P) = 1$.*

We define the following:

1. $W(G) = \sum_{\forall v \in V} w(v)$.

2. $W(P) = \sum_{\forall v \in V(P)} w(v)$ for any path P.

3. $U(x_i) = \{v \in X | v < x_i\}$.

4. $W_U(x_i) = \sum_{v \in U(x_i)} w(v)$.

5. $UAdj(x_i, y_j) = \{v \in X | (v < x_i \text{ and } vy_j \in E)\}$.

6. $W_{UAdj}(x_i, y_j) = \sum_{v \in UAdj(x_i, y_j)} w(v)$.

7. $USUM(x_i, y_j) = \sum_{v \in U(x_i)} min(d(v, x_i), d(v, y_j))w(v)$.

By the above definitions and *Remark* 4.3.1, we have the following *Lemma*.

Lemma 4.3.2 *1.* $W_U(x_1) = 0$, $W_U(x_i) = W_U(x_{i-1}) + w(x_{i-1})$.

2. $W_{UAdj}(x_i, y_j) = W_U(x_i)$ - $W_U(L(y_j))$.

3. $USUM(x_i, y_j) = 2W_U(L(y_j)) + W_{UAdj}(x_i, y_j)$.

The above values can be computed for all $(x_i, y_j) \in E$ *in* $O(|E|)$ *time.*

Proof: From equation 1 of the lemma, we can compute $W_U(x_i)$ $\forall x_i \in X$ in $(O|V|)$ time.

From equation 2 of the lemma and from the precomputed values of W_U, we can compute $W_{UAdj}(x_i, y_j)$ $\forall (x_i, y_j) \in E$ in $O(|E|)$ time.

We shall prove the third equation of the above lemma now. While calculating $USUM(x_i, y_j)$, we have to add the cost incurred by all the vertices of X above x_i in reaching (x_i, y_j). The vertices of X that are above $L(y_j)$ incur a cost of two and the total cost is thus $2W_U(L(y_j))$. The vertices that are in between $L(y_j)$ and x_i (including $L(y_j)$) incur a cost of one and their cost is $W_{UAdj}(x_i, y_j)$.

From the third equation and from the precomputed values of W_U and W_{UAdj}, we can compute $USUM(x_i, y_j)$ $\forall (x_i, y_j) \in E$ in $O(|E|)$ time.

So the overall time taken is $O(|V| + |E|)$ which is $O(|E|)$ for connected graphs. □

As we form the path, we can compute $W(P)$ for any path P. We show a method to do this in *Lemma* 4.4.2.

Lemma 4.3.3 *For any ordered path P, the cost of the path $d(P)$ can be computed in $O(1)$ time after $O(|E|)$ preprocessing.*

Proof: We claim that $d(P) = W(G)$ - $W(P)$ - $W_U(x_{a_1})$ + $USUM(x_{a_1}, y_{b_1})$, where (x_{a_1}, y_{b_1}) is the *first edge* of the ordered path P. $W(G)$ - $W(P)$ - $W_U(x_{a_1})$ accounts for the cost of all vertices except vertices in $U(x_{a_1})$. $USUM(x_{a_1}, y_{b_1})$, accounts for the cost due to vertices in $U(x_{a_1})$.
Also, note that if the path P does not contain any $x \in X$, then in the above equation for $d(P)$, we set $x_{a_1} = L(y_{b_1})$ where y_{b_1} is the vertex of least index in the ordered path. □

4.4 Finding the core path:

4.4.1 Algorithm

Let G_{ij} be a graph induced by the vertices $\{x_1, x_2, \ldots, x_i\}$ and $\{y_1, y_2, \ldots, y_j\}$.

Remark 4.4.1 *For every edge $(x, y) \in E$ let $path^l_{xy}$ denote the path with (x, y) as its* last edge *such that*

1. *The path is ordered and it is of length l.*
2. *It is the minimum cost ordered path of length l with (x, y) as its* last edge.

As noted by *Remark* 4.4.1, we have to find $path^l_{xy}$ $\forall xy \in E$.

1. Let $P^l_{x_i y_j}$ denote an ordered path of length l and of minimum cost among all ordered paths of length l in G_{ij} with (x_i, y_j) as the *last edge* and y_j being the vertex of degree one in the path P.
2. Let $P^l_{y_j x_i}$ denote an ordered path of length l and of minimum cost among all ordered paths of length l in G_{ij} with (x_i, y_j) as the *last edge* and x_i being the vertex of degree one in the path P.

From $P^l_{x_i y_j}$ and $P^l_{y_j x_i}$, we can compute $path^l_{x_i y_j}$ which is defined such that $d(path^l_{x_i y_j})$ = $Min\{d(P^l_{x_i y_j}), d(P^l_{y_j x_i})\}$. Let $P^l_{y_i y_j}$ be an ordered path of length l and of minimum

cost among all ordered paths of length l in G with (y_i, y_j) as the *last edge* and y_j being the degree one vertex. We first find $Min_1 = \underset{(x,y)\in E}{Min}\ d(path^l_{xy})$ and hence the path corresponding to that cost. We then find $Min_2 = \underset{(y_i,y_j)\in E|i<j}{Min}\ d(P^l_{y_iy_j})$ and the corresponding path. We have put in the restriction $i < j$ because from our definition of ordered path it follows that an ordered path cannot have (y_i, y_j) as the *last edge* such that y_j is the vertex of degree one in the path and $j < i$. Finally we find $Mincost = Min\{Min_1, Min_2\}$. The path corresponding to $Mincost$ will yield the core path of length l in a threshold graph.

$P^1_{x_iy_j} = (x_i, y_j)$,$P^1_{y_jx_i} = (y_j, x_i)$, $P^1_{y_iy_j} = (y_i, y_j)$ and their costs can be computed using *Lemma* 4.3.3. We give the dynamic programming equations and the Algorithm below. Also note that to compute $d(P)$ we need $W(P)$ which is given as a part of *Lemma* 4.4.2.

The following equations can be used to find the costs of $P^q_{x_iy_j}$, $P^q_{y_jx_i}$, $P^q_{y_jy_i}$,$\forall q \geq 2$.

Lemma 4.4.2 *In graph G_{ij} $\forall 1 \leq i \leq |X|$ and $\forall 1 \leq j \leq |Y|$ and $(x_i, y_j) \in E(G)$ we have that,*

$$d(P^q_{x_iy_j}) = \begin{cases} \underset{\forall(y_k,x_i)\in E, k<j}{Min} d(P^{q-1}_{y_kx_i}y_j) & \text{if such } y_k\text{'s exist} \\ \infty & \text{otherwise} \end{cases} \tag{4.1}$$

$$= \begin{cases} Min\{d(P^{q-1}_{y_{j-1}x_i}y_j),\ d(\alpha_{q-1}(P^q_{x_iy_{j-1}})y_j)\} & \text{if } j > 1 \text{ and } (x_i, y_{j-1}) \in E \\ \infty & \text{otherwise} \end{cases} \tag{4.2}$$

$$d(P^q_{y_jx_i}) = \begin{cases} \underset{\forall(x_k,y_j)\in E, k<i}{Min} d(P^{q-1}_{x_ky_j}x_i) & \text{if such } x_k\text{'s exist} \\ \infty & \text{otherwise} \end{cases} \tag{4.3}$$

$$= \begin{cases} Min\{d(P^{q-1}_{x_{i-1}y_j}x_i),\ d(\alpha_{q-1}(P^q_{y_jx_{i-1}})x_i)\} & \text{if } i > 1 \text{ and } (x_{i-1}, y_j) \in E \\ \infty & \text{otherwise} \end{cases} \tag{4.4}$$

In graph G $\forall 1 \leq i \leq |Y|$ *and* $\forall 1 \leq j < i$ *such that* $(y_j, y_i) \in E(G)$ *we have that*

$$d(P^q_{y_j y_i}) = Min\{\underset{\forall (y_k, y_j) \in E, k<j}{Min} d(P^{q-1}_{y_k y_j} y_i), \underset{\forall (x_k, y_j) \in E}{Min} d(P^{q-1}_{x_k y_j} y_i)\} \tag{4.5}$$

Note that in the above equation, such a x_k *or* y_k *will always exist due to the threshold graph property.*

When $i > j + 1$

$$d(P^q_{y_j y_i}) = d(\alpha_{q-1}(P^q_{y_j y_{j+1}}) y_i) \tag{4.6}$$

The weights of the corresponding paths can be computed from the following equations.

$$W(P^{q-1}_{y_k x_i} y_j) = W(P^{q-1}_{y_k x_i}) + w(y_j),\ W(P^{q-1}_{x_k y_j} x_i) = W(P^{q-1}_{x_k y_j}) + w(x_i),$$
$$W(P^{q-1}_{y_k y_j} y_i) = W(P^{q-1}_{y_k y_j}) + w(y_i),\ W(P^{q-1}_{x_k y_j} y_i) = W(P^{q-1}_{x_k y_j}) + w(y_i)$$

Proof: The proof for equations (4.1), (4.3), (4.5) and the equations for computing the weights follow from their respective definitions. So we have to prove (4.2), (4.4) and (4.6). We will first prove (4.2) and the proof for (4.4) will follow from that.

Claim: $d(\alpha_{r-1}(P^r_{x_i y_{j-1}}) y_j) = \underset{\forall (y_k, x_i) \in E | k<j-1}{Min} d(P^{r-1}_{y_k x_i} y_j)$

It is clear that if the above claim is established, then equations (4.2) and (4.1) will become the same and thus (4.2) is proven.

Let $\alpha_{r-1}(P^r_{x_i y_{j-1}})$ be such that it has (y_t, x_i) as the *last edge*.

$$\Rightarrow d(P^{r-1}_{y_t x_i} y_{j-1}) = \underset{\forall (y_{t'}, x_i) \in E | t'<j-1}{Min} d(P^{r-1}_{y_{t'} x_i} y_{j-1})$$
$$\Rightarrow d(P^{r-1}_{y_t x_i}) = \underset{\forall (y_{t'}, x_i) \in E | t'<j-1}{Min} d(P^{r-1}_{y_{t'} x_i})$$
$$\Rightarrow d(P^{r-1}_{y_t x_i} y_j) = \underset{\forall (y_{t'}, x_i) \in E | t'<j-1}{Min} d(P^{r-1}_{y_{t'} x_i} y_j)$$

which is precisely the statement of our *claim*.

We will now prove equation (4.6)

Claim : $d(\alpha_{q-1}(P^q_{y_j y_{j+1}}) y_i) = Min\{\underset{\forall (y_k y_j) \in E, k<j}{Min} d(P^{q-1}_{y_k y_j} y_i), \underset{\forall (x_k y_j) \in E}{Min} d(P^{q-1}_{x_k y_j} y_i)\}$

If the above claim is established, equations (4.6) and (4.5) will become the same and thus (4.6) is proven.

Case 1: Let $\alpha_{q-1}(P^q_{y_jy_{j+1}})$ be such that it has (x_t, y_j) as its *last edge*.

$$\Rightarrow\ d(P^{q-1}_{x_ty_j}y_{j+1}) = Min\{\underset{\forall(y_k,y_j)\in E,k<j}{Min} d(P^{q-1}_{y_ky_j}y_{j+1}), \underset{\forall(x_ky_j)\in E}{Min} d(P^{q-1}_{x_ky_j}y_{j+1})\}$$

$$\Rightarrow\ d(P^{q-1}_{x_ty_j}) = Min\{\underset{\forall(y_k,y_j)\in E,k<j}{Min} d(P^{q-1}_{y_ky_j}), \underset{\forall(x_ky_j)\in E}{Min} d(P^{q-1}_{x_ky_j})\}$$

$$\Rightarrow\ d(P^{q-1}_{x_ty_j}y_i) = Min\{\underset{\forall(y_k,y_j)\in E,k<j}{Min} d(P^{q-1}_{y_ky_j}y_i), \underset{\forall(x_ky_j)\in E}{Min} d(P^{q-1}_{x_ky_j}y_i)\}$$

But this is precisely the statement of the claim.

Case 2: Let $\alpha_{q-1}(P^q_{y_jy_{j+1}})$ be such that it has (y_t, y_j) as its *last edge* The proof in this case is similar to case 1. Hence proved. □

Theorem 4.4.3 *The* Algorithm *2 computes the core path of a threshold graph in* $O(l|E|)$ *time.*

Proof: The algorithm first computes the values of $d(path^l_{xy})$ for each edge $(x, y) \in E$ and $d(P^l_{y_iy_j})$ $\forall(y_i, y_j) \in E$ using *Lemma* 4.4.2. It then computes the minimum cost path by finding Min $\{\underset{(y_i,y_j)\in E|i<j}{Min} d(P^l_{y_iy_j}), \underset{(x,y)\in E}{Min} d(path^l_{xy})\}$ and hence the correctness follows.

*Time complexity:*For a given length and a given edge (x_i, y_j) or $(y_j, x_i) \in E$, the algorithm takes $O(1)$ time to compute the value of $d(P^{length}_{x_iy_j})$ or $d(P^{length}_{y_ix_j})$. Therefore to compute the value of $d(P^l_{x_iy_j})$ and $d(P^l_{y_ix_j})$ for all edges (x_i, y_j) and $(y_i, x_j) \in E$, it takes $O(l|E|)$ time. Also for a given length and a given edge $(y_j, y_i) \in E$, the algorithm takes $O(1)$ time to compute $d(P^{length}_{y_jy_i})$ when $i > j+1$. When $i = j+1$, the algorithm takes utmost $deg(y_j)$ time to compute $d(P^{length}_{y_jy_i})$. But there are only $|Y|$-1 edges (y_j, y_i) that satisfy $i = j+1$ and hence the total time taken for these edges is utmost $deg(y_1) + deg(y_2) + \cdots + deg(y_{|Y|-1})$ which is utmost $O(|E|)$. Therefore to compute the value of $d(P^l_{y_jy_i})$ for all edges $(y_j, y_i) \in E$, it takes $O(l|E|)$ time. □

Algorithm 2 : Algorithm to compute the core path of length l in a threshold graph.

Require: A Threshold graph $G = (V, E)$ with ordering $\{x_1, x_2 \dots x_{|X|}\}$ and $\{y_1, y_2 \dots y_{|Y|}\}$.

Compute $d(P^1_{x_i y_j})$, $d(P^1_{y_j x_i})$ and $d(P^1_{y_i y_j})$ $\forall i, j$.

for $length$=2 to l **do**

for $i = 1$ to $|X|$ **do**

for $y = L(x_i)$ to $R(x_i)$ **do**

Compute $d(P^{length}_{yx_i})$ by using equation (4.4)

end for

end for

for i=1 to $|Y|$ **do**

for $x = L(y_i)$ to $R(y_i)$ **do**

Compute $d(P^{length}_{xy_i})$ by using equation (4.2)

end for

end for

for $i = 1$ to $|Y|$-1 **do**

$d(P^{length}_{y_i y_{i+1}}) = Min\{\underset{\forall(y_k,y_i)\in E, k<i}{Min} d(P^{length-1}_{y_k y_i} y_{i+1}),\ \underset{\forall(x_k,y_i)\in E}{Min} d(P^{length-1}_{x_k y_i} y_{i+1})\}$

end for

for i=3 to $|Y|$ **do**

for $j = 1$ to $i - 2$ **do**

$d(P^{length}_{y_j y_i}) = d(\alpha_{length-1}(P^{length}_{y_j y_{j+1}}) y_i)$

end for

end for

end for

Compute $d(path^l_{x_i y_j}) = Min\{d(P^l_{x_i y_j}), d(P^l_{y_j x_i})\}$ $\forall (x_i, y_j) \in E$

Compute $Min_1 = \underset{(x,y)\in E}{Min} d(path^l_{xy})$

Compute $Min_2 = \underset{(y_i,y_j)\in E | i<j}{Min} d(P^l_{y_i y_j})$

$Mincost = Min\{Min_1, Min_2\}$

Output the path corresponding to $Mincost$

CHAPTER 5

Conditional Core in Threshold Graphs

5.1 Introduction

In this chapter, we consider a minor variant of the core path problem − the conditional core problem. In this problem, it is required to locate a path shaped facility in the network under the condition that some facilities have already been located. Here, we solve the conditional core problem in threshold graphs. We define S to be the set of facilities that have already been located. The new facility that is to be located has to be a path that does not include any of the vertices in S.

The objective here is to minimize $d(P) = \sum_{v \in V} (d(v, P), d(v, S))w(v)$. For all vertices $v \in S$, we assign $w(v) = 0$.

We use here the same approach as for the core path problem in threshold graphs i.e. for each edge, we find the *ordered path* of least cost and of length l, which ends at that edge. From these values, we compute the conditional core path by choosing the path of minimum cost. The only differences in the definition of notations that we make here are,

1. The definition of $d(P)$
2. The definition of $USUM(x_i, y_j)$.

We present below a method to compute the value of $USUM(x_i, y_j)$ under the new definition. All other notations mean the same according to their definition and usage in the previous chapter.

5.2 Conditional Core

We recall the following definitions from the previous chapter.

1. $W(G) = \sum_{\forall v \in V} w(v)$.
2. $W(P) = \sum_{\forall v \in V(P)} w(v)$ for any path P.
3. $U(x_i) = \{v \in X | v < x_i\}$.
4. $W_U(x_i) = \sum_{v \in U(x_i)} w(v)$.
5. $UAdj(x_i, y_j) = \{v \in X | (v < x_i \text{ and } vy_j \in E)\}$.
6. $W_{UAdj}(x_i, y_j) = \sum_{v \in UAdj(x_i, y_j)} w(v)$.
   For the definition of $USUM$ we make the following modification.
7. $USUM(x_i, y_j) = \sum_{v \in U(x_i)} min(d(v, x_i), d(v, y_j), d(v, S))w(v)$.
   Additionally, we define the following quantity.
8. $TOSUB(x_i) = \sum_{v \in U(x_i), d(v,S)=1} w(v)$.

Lemma 5.2.1 *1.* $TOSUB(x_1) = 0$.

2. $$TOSUB(x_i) = \begin{cases} TOSUB(x_{i-1}) + w(x_{i-1}) & \text{if } d(x_{i-1}, S) = 1 \\ TOSUB(x_{i-1}) & \text{otherwise} \end{cases}$$

3. $USUM(x_i, y_j) = 2 * W_U(L(y_j)) - TOSUB(x_i) + W_{UAdj}(x_i, y_j)$

Proof: The proof for equation on $TOSUB(x_i)$ follows from definition. We will prove the equation for $USUM(x_i, y_j)$. For computing $USUM(x_i, y_j)$, we need to add the cost incurred by vertices that are in $W_U(x_i)$ and adjacent to y_j, and the cost incurred by those vertices that are in $W_U(x_i)$ but not adjacent to y_j. The cost incurred by those vertices that are in $W_U(x_i)$ and adjacent to y_j, is $W_{UAdj}(x_i, y_j)$. Among the vertices that are in $W_U(x_i)$ but not adjacent to y_j, those vertices that are adjacent to a vertex in S incur a cost of 1, and those that are not adjacent to any vertex in S, incur a cost of 2. This value is captured by 2*$W_U(L(y_j))$ - $TOSUB(x_i)$. □

From now on, we follow the same procedure to compute the conditional core path, as was done to compute the core path i.e. with the above definition of $USUM(x_i, y_j)$, we can use the same formulas and algorithms as for the core path problem to compute the conditional core path in threshold graphs.

Lemma 5.2.2 *The cost of any ordered path $d(P)$ can be computed in $O(1)$ time, after $(O|E|)$ preprocessing.*

Proof: We claim that $d(P) = W(G) - W(P) - W_U(x_{a_1}) + USUM(x_{a_1}, y_{b_1})$, where (x_{a_1}, y_{b_1}) is the first edge of the ordered path P. $W(G)$ - $W(P)$ - $W_U(x_{a_1})$ accounts for the cost of all vertices except vertices in $U(x_{a_1})$. $USUM(x_{a_1}, y_{b_1})$, accounts for the cost due to vertices in $U(x_{a_1})$.
Also, note that if the path P does not contain any $x \in X$, then in the above equation for $d(P)$, we set $x_{a_1} = L(y_{b_1})$ where y_{b_1} is the vertex of least index in the ordered path. □

Since we require that the path located, should not contain any vertex $v \in S$, we have to run the algorithm provided in previous chapter on each component of $V - S$. If vertices in S are removed from V, the graph can either remain connected or be disconnected. In case the graph becomes disconnected and has $t(> 1)$ components, because of the properties of a threshold graph, it follows that, of these t components $t-1$ components are single vertices in X. So if at all, a path of length l were to exist, it must be in the remaining component. Let us call this component C_t and let V' and E' denote respectively the set of vertices and the set of edges in component C_t.
Let $X' = \{x' \colon x' \in X \cap C_t\}$ and let $Y' = \{y' \colon y' \in Y \cap C_t\}$. Let $x'_1, x'_2, \ldots, x'_{|X'|}$ denote the vertices of X' arranged according to the order in which they appear in the ordering $x_1 < x_2 < \cdots < x_{|X|}$ which we defined for vertices of X and let $y'_1, y'_2, \ldots, y'_{|Y'|}$ denote the vertices of Y' arranged according to the order in which they appear in the ordering $y_1 < y_2 < \cdots < y_{|Y|}$ which we defined for vertices of Y.

The following equations can be used to find the costs of $P^q_{x'_i y'_j}$, $P^q_{y'_j x'_i}$, $P^q_{y'_j y'_i}$,$\forall q \geq 2$. They are analogous to what was stated for computing these quantities for threshold

graphs in the previous chapter. But we state them below for the sake of completeness.

Lemma 5.2.3 *In graph* G'_{ij} $\forall 1 \leq i \leq |X'|$ *and* $\forall 1 \leq j \leq |Y'|$ *and* $(x'_i, y'_j) \in E'(G)$ *we have that,*

$$d(P^q_{x'_i y'_j}) = \begin{cases} \underset{\forall (y'_k, x'_i) \in E, k<j}{Min} d(P^{q-1}_{y'_k x'_i} y'_j) & \text{if such } y'_k\text{'s exist} \\ \infty & \text{otherwise} \end{cases} \tag{5.1}$$

$$= \begin{cases} Min\{d(P^{q-1}_{y'_{j-1} x'_i} y'_j),\ d(\alpha_{q-1}(P^q_{x'_i y'_{j-1}}) y'_j)\} & \text{if } j > 1 \text{ and } (x'_i, y'_{j-1}) \in E \\ \infty & \text{otherwise} \end{cases} \tag{5.2}$$

$$d(P^q_{y'_j x'_i}) = \begin{cases} \underset{\forall (x'_k, y'_j) \in E', k<i}{Min} d(P^{q-1}_{x'_k y'_j} x'_i) & \text{if such } x'_k\text{'s exist} \\ \infty & \text{otherwise} \end{cases} \tag{5.3}$$

$$= \begin{cases} Min\{d(P^{q-1}_{x'_{i-1} y'_j} x'_i),\ d(\alpha_{q-1}(P^q_{y'_j x'_{i-1}}) x'_i)\} & \text{if } i > 1 \text{ and } (x'_{i-1}, y'_j) \in E \\ \infty & \text{otherwise} \end{cases} \tag{5.4}$$

In graph G $\forall 1 \leq i \leq |Y'|$ *and* $\forall 1 \leq j < i$ *such that* $(y'_j, y'_i) \in E'(G)$ *we have that*

$$d(P^q_{y'_j y'_i}) = Min\{\underset{\forall (y'_k, y'_j) \in E', k<j}{Min} d(P^{q-1}_{y'_k y'_j} y'_i), \underset{\forall (x'_k, y'_j) \in E'}{Min} d(P^{q-1}_{x'_k y'_j} y'_i)\} \tag{5.5}$$

Note that in the above equation, such a x'_k *or* y'_k *will always exist due to the threshold graph property.*

When $i > j + 1$

$$d(P^q_{y'_j y'_i}) = d(\alpha_{q-1}(P^q_{y'_j y'_{j+1}}) y'_i) \tag{5.6}$$

We compute the conditional core path by executing *Algorithm* 3 below. The proof of correctness is identical to the proof given to the core path algorithm for threshold graphs in the previous chapter.

Algorithm 3 : Algorithm to compute the core path of length l in a threshold graph.

Require: A Threshold graph $G' = (V', E')$ with ordering $\{x'_1, x'_2 \dots x'_{|X'|}\}$ and $\{y'_1, y'_2 \dots y'_{|Y'|}\}$.

Compute $d(P^1_{x'_i y'_j})$, $d(P^1_{y'_j x'_i})$ and $d(P^1_{y'_i y'_j})$ $\forall x'_i \in X' and y'_j \in Y'$.

for $length$=2 to l **do**

for $i = 1$ to $|X'|$ **do**

for $y' = L(x'_i)$ to $R(x'_i)$ **do**

Compute $d(P^{length}_{y' x'_i})$ using equation (5.4)

end for

end for

for i=1 to $|Y'|$ **do**

for $x' = L(y'_i)$ to $R(y'_i)$ **do**

Compute $d(P^{length}_{x' y'_i})$ using equation (5.2)

end for

end for

for $i = 1$ to $|Y'|$-1 **do**

$d(P^{length}_{y'_i y'_{i+1}}) = Min\{\underset{\forall (y'_k, y'_i) \in E', k<i}{Min} d(P^{length-1}_{y'_k y'_i} y'_{i+1}),\ \underset{\forall (x'_k, y'_i) \in E'}{Min} d(P^{length-1}_{x'_k y'_i} y'_{i+1})\}$

end for

for i=3 to $|Y'|$ **do**

for $j = 1$ to $i - 2$ **do**

$d(P^{length}_{y'_j y'_i}) = d(\alpha_{length-1}(P^{length}_{y'_j y'_{j+1}}) y'_i)$

end for

end for

end for

Compute $d(path^l_{x'_i y'_j}) = Min\{d(P^l_{x'_i y'_j}), d(P^l_{y'_j x'_i})\}$ $\forall (x'_i, y'_j) \in E'$

Compute $Min_1 = \underset{(x', y') \in E'}{Min} d(path^l_{x' y'})$

Compute $Min_2 = \underset{(y'_i, y'_j) \in E' | i<j}{Min} d(P^l_{y'_i y'_j})$

$Mincost = Min\{Min_1, Min_2\}$

Output the path corresponding to $Mincost$

CHAPTER 6

Conclusion

In this report, we have solved the core path problem in proper interval graphs and threshold graphs by giving polynomial time algorithms for both. Additionally, we have established the NP-Completeness of the core path problem in the above classes of graphs, when arbitrary positive, real weights are assigned to edges and unit weights to vertices. The central theme in solving both of these problems is in identifying an ordering among vertices of the graph. This enables us to reduce the search space of the problem from exponential to polynomial. With the help of this ordering among vertices, dynamic programming equations were established to obtain the core path efficiently.

On the other hand, in higher classes of graphs like permutation graphs, where such an ordering of vertices is not found, it is not possible to use this approach to find the core path. The most interesting open problem in this domain is that, in interval graphs and permutation graphs, the complexity of the longest path problem is still unresolved. Therefore the existence of a path of length l is unresolved. Consequently, the core path problem is open in these classes of graphs.

References

[**1**] Stephen Alstrup, Peter W. Lauridsen, Peer Sommerlund, and Mikkel Thorup. Finding cores of limited length. *WADS '97: Proceedings of the 5th International Workshop on Algorithms and Data Structures*, pages 45–54, 1997.

[**2**] Ronald Becker, Isabella Lari, Andrea Scozzari, and Giovanni Storchi. The location of median paths on grid graphs. *Annals of Operations Research*, 150(1):65–78, 2007.

[**3**] Ronald I. Becker, Yen I. Chang, Isabella Lari, Andrea Scozzari, and Giovanni Storchi. Finding the l-core of a tree. *Discrete Appl. Math.*, 118(1-2):25–42, 2002.

[**4**] V. Chvátal and P. L. Hammer. Aggregations of inequalities. *Ann Discrete Math*, 1:145–162, 1977.

[**5**] M. C. Golumbic. Threshold graphs and synchronizing parallel processes. *Colloq. Math. Soc. János Bolyai*, 18:419–428, 1978.

[**6**] M. C Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press New York, 1980.

[**7**] S. L. Hakimi, E. F. Schmeichel, and Martine Labb'e. On locating path- or tree-shaped facilities on networks. *Networks*, 23(6):543 – 555, 1993.

[**8**] P. B. Henderson and Y. Zalcstein. A graph-theoretic characterization of the pv_{chunk} class of synchronizing primitives. *SIAM J. Comput.*, 6:88–108, 1977.

[**9**] G. J. Koop. Cyclic scheduling of offweekends. *Oper. Res. Lett.*, 4:259–263, 1986.

[**10**] N. V. R. Mahadev and U. N. Peled. Threshold graphs and related topics. *Ann. of Discrete Math.*, 56, 1995.

[**11**] Edward Minieka. The optimal location of a path or a tree in a tree network. *Networks*, 15:309–321, 1985.

[**12**] Edward Minieka and Niranjani H. Patel. On finding the core of a tree with a specified length. *J. Algorithms*, 4(4):345–352, 1983.

[**13**] Christine A. Morgan and Peter J. Slater. A linear algorithm for a core of a tree. *J. Algorithms*, 1(3):247–258, 1980.

[**14**] E. T. Ordman. Minimal threshold separators and memory requirements for synchronization. *SIAM J. Comput.*, 18:152–165, 1989.

[**15**] Shietung Peng and Win-Tsung Lo. Efficient algorithms for finding a core of a tree with a specified length. *J. Algorithms*, 20(3):445–458, 1996.