HAMILTONICITY AND LONGEST PATH PROBLEM ON SPECIAL CLASSES OF GRAPHS

A THESIS

submitted by

ESHA GHOSH

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

July 2011

Certificate

This is to certify that the thesis entitled **Hamiltonicity and Longest Path Problem on Special Classes of Graphs** submitted by **Esha Ghosh** to the Indian Institute of Technology, Madras, for the award of the Degree of Master of Science (by research) is a record of bonafide research work carried out by her under my supervision and guidance. The contents of this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

IIT Madras
Chennai 600036

Research Guide

Date:

(Prof. C Pandu Rangan)

To all lovers of Graph Theory

ACKNOWLEDGEMENTS

It is a pleasure to thank those who made this thesis possible. First and foremost, I would like to thank my advisor Prof. C. Pandu Rangan for his immense support and guidance. I feel privileged to be among his students. He celebrates research in the truest sense of the term. I have been lucky enough to sit through his "Joy of Algorithms" lecture series; it was a divine experience. I found the perfect embodiment of knowledge, wisdom and compassion in him. His philosophical depth, mathematical maturity and profound insight about every little thing, as trifling as they might seem to appear, (not to mention his funny bone) has shaped my perception. He is the quintessential teacher who brings back the spirit of traditional "Gurukul Siksha" of India. My deepest gratitude to him for being my advisor. I also thank him for giving me the opportunity of research collaboration with Dr. N.S. Narayanaswamy (IIT Madras) and Mr. Subhas K. Ghosh (Siemens Information Systems limited).

I thank Dr. N.S. Narayanaswamy and Mr. Subhas K. Ghosh for being such wonderful collaborators. They were extremely patient and perceptive during our discussions and flooded me with insightful ideas. They helped in technical reports writing as well. I am looking forward to further collaboration.

I would like to thank all my General Test Committee members, Prof. S. A. Choudam, Dr. N.S. Narayanaswamy and Prof. Kamala Krithivasan for their valuable advices and suggestions. I specially thank Prof. S. A. Choudam for his enlightening and motivating lectures on Graph Theory. It was of great help to get my foundations in basic graph theory.

I thank all members of CS-Theory group of our department and Dr. Jayalal Sarma for spearheading this group. It was a great experience to be a part of this group.

I would like to thank Arpita, Ashish, Amjed, Saikrishna, Thirumala and Sadagopan for being wonderful seniors. They were encouraging and supportive throughout and helped me in every possible way, whenever I reached out.

My sincere thanks to Chaya, Preethi and Sudarshan for their valuable comments and suggestions which helped me write my thesis better. I thank Bharathi Priyaa for fruitful discussion at the initial stage of my research.

I thank all my labmates Sharmila, Vivek, Subhasini, Bhargav, Billy, Akash, Chaya, Preetha madam, Prateek and Sangeetha who made my stay at TCS lab fond and special. I thank all my friends and hostelmates who made my stay at IIT Madras a very pleasant one. A very special thanks to Chaya and Preethi, without whom, the stay would not have been half as enjoyable.

I thank department of computer science, for facilitating the research and Siemens Information Systems Limited - Corporate Research, Bangalore for funding my research during the five months internship period.

Last, but not the least, I feel privileged to belong to a family which taught me the joy of being. I thank my family for what not.

ABSTRACT

KEYWORDS: Hamiltonian Cycle, Optical Transpose Interconnection Systems, Fault Tolerance, Independent Spanning Tree, Longest Path, Absolute Approximation, Biconvex Graphs

In this thesis we will discuss about two well known graph theoretic problems: Hamiltonicity and Longest Path Problem.

Hamiltonicity is an important property of any graph and is of special interest in parallel and distributed computation. Existence of Hamiltonian cycle allows efficient emulation of distributed algorithms on a network wherever such algorithm exists for linear-array and ring. Hamiltonicity can also be used for construction of independent spanning tree and leads to designing fault tolerant protocols. Optical Transpose Interconnection Systems (OTIS) is a widely studied interconnection network topology with n^2 nodes is constructed by taking n copies of an n-node basis graph, and applying simple rule for connecting inter-cluster edges. Surprisingly, to our knowledge, only one strong result is known regarding Hamiltonicity of OTIS - showing that OTIS graph built of Hamiltonian basis graph are Hamiltonian. In this work we consider Hamiltonicity of OTIS networks, built on Non-Hamiltonian bases and answer some important questions. Firstly, we prove that Hamiltonicity of base graph is not a necessary condition for the OTIS to be Hamiltonian. We further show that, it is not sufficient for the basis graph to have Hamiltonian path for the OTIS constructed on it to be Hamiltonian. We thoroughly investigate the Hamiltonicity of OTIS graphs formed on generalized Butterfly graphs as basis. We also give constructive proof of Hamiltonicity for a large class of Butterfly-OTIS. This proof also leads to an alternate linear time algorithm for two rooted independent spanning trees on this class of OTIS graphs.

The longest path problem is the problem of finding a simple path of maximum length in a graph. Longest path algorithms find various applications across diverse fields. The longest path in program activity graph is known as critical path, which represents the sequence of program activities that take the longest time to execute. Longest path algorithm is required to calculate critical paths. The well-known Traveling Salesman problem is also a special case of Longest Path problem. Polynomial solutions for this problem are known only for special classes of graphs, while it is NP-hard on general graphs. In this work we are proposing an $O(n^6)$ time algorithm to find the longest path on biconvex graphs, where n is the number of vertices of the input graph. We also propose an $O(n^4)$ time absolute approximation algorithm for the same problem.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Hamiltonian Cycle Problem

1.1.1 Dining Table Dilemma

Let us consider an interesting piece of puzzle. Alf, Bob, Colin, Dave, Euan, Frank and George are holding a dinner party. They are to sit in a circular table and dine. But, sadly enough, all of them are not ready to sit next to each other. Alf does not like sitting next to Bob or Colin. Bob will not sit next to Colin, Euan or George. Colin minds sitting next to Dave or George. Euan will not sit next to George. Apart from these constraints, they are happy sitting next to anyone else. The question now is, is there any feasible way to seat them around the table at all? Let us represent the compatibility network as a graph where the nodes represent the people (A)lf, (B)ob, (C)olin, (D)ave, (E)uan, (F)rank and (G)eorge. Two nodes are connected by an edge if they have no problem in sitting next to each other (Fig 1.1(a)). Now, if we can find a cycle in the graph which covers all the nodes exactly once, we basically find an arrangement around the table! (Fig 1.1(b))

The problem we just introduced is a very well known problem in Graph Theoretic literature. This is known as *Hamiltonian Cycle Problem*. Named after Sir William Rowan Hamilton, traces its origins to the 1850s. The problem was known as "The Travelers Dodecahedron". We find its description in a chapter on Hamilton's Game in volume 2 of *É*douard Lucas' "R*é*cr*é*ations Math*é*matiques" and another mention in the 3rd edition of Ahrens' German work on Recreational Mathematics. The original puzzle is as follows:

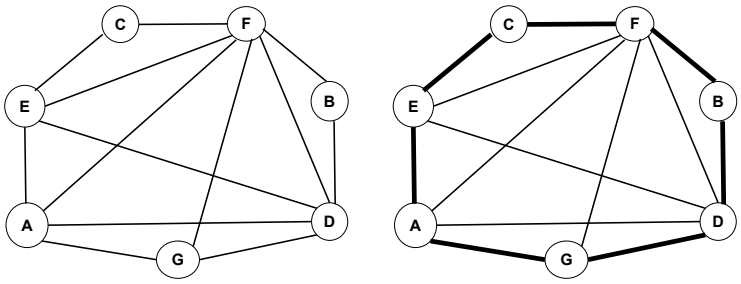

Figure 1.1: (a)The Graph representing the dinner table situation (b)The Cycle is shown in bold edges

The graph in Fig 1.2 is a two-dimensional projection on the plane of a dodecahedron (a three-dimensional solid with twelve pentagonal faces). Each vertex on the graph represents the respective vertex of the dodecahedron, and each line between any two points - the respective edge. The object of this puzzle is to visit all the twenty points on the graph starting at any point, visiting every point exactly once, and coming back to the starting point.

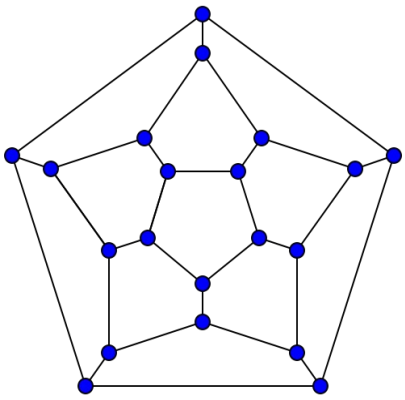

Figure 1.2: Dodecahedron

Now we will formally introduce the problem.

1.1.2 Formal Introduction

A Hamiltonian cycle is a spanning cycle in a graph, i.e., a cycle through every vertex, and a Hamiltonian path is a spanning path.[1] A graph containing a Hamiltonian cycle is said to be Hamiltonian. It is clear that every graph with a Hamiltonian cycle has a Hamiltonian path but the converse is not necessarily true. The study of Hamiltonian cycles and Hamiltonian paths in general and special graphs has been motivated by practical applications and by the issues of complexity. The problem of finding whether a graph is Hamiltonian is proved to be NP-complete for general graphs. [Garey and Johnson, 1979] The problem remains NP-Complete, even when restricted to special graph classes, like bipartite graphs, planar, cubic, 3-connected, and has no face with fewer than 5 edges, square of a graph, or even when a Hamiltonian path for the graph is given as part of the input instance. [Garey and Johnson, 1979] The problem of finding Hamiltonian path is also NP-Complete on the first two mentioned special classes. There are several other classes on which both these problems remain intractable. Naturally, this problem has been an active area of research in Graph Theory and Algorithms. The problem, apart from being of immense theoretical interest, is of great practical importance too.

There are different lines of research in the area of Hamiltonicity. One line of attempt is to show that a graph is Hamiltonian, without finding a Hamiltonian Cycle. Proving that a graph is Non-Hamiltonian is also Hard. Several necessary and sufficient conditions are known for Hamiltonicity [Gould, 1991]. However, no necessary and sufficient condition is known except Ore's condition [Broersma, 2002]. Let $G = (V, E)$ be a simple graph with n vertices and u, v be distinct nonadjacent vertices of G with $degree(u) + degree(v) \geq n$. Ore's condition states that G is Hamiltonian if and only if $G + (u, v)$ is Hamiltonian. Another important point to note is that, most

[1] All definitions and notations used here and in the subsequent sections are defined in Chapter 2

of the sufficiency conditions hold for dense graphs. Very few results are known for Hamiltonicity on sparse graphs.

A different line of research in this field is to design efficient randomized algorithms for Hamiltonicity detection with very high probability. A very recent result by Andreas Björklund in FOCS 2010 presents a Monte Carlo algorithm for Hamiltonicity detection in an n-vertex undirected graph running in $O^*(1.657^n)$ time [Bjöandrklund, 2010]. This is the first superpolynomial improvement on the worst case runtime for the problem since the $O^*(2^n)$ bound established for TSP almost fifty years ago (Bellman 1962, Held and Karp 1962). For bipartite graphs, the bound is improved to $O^*(1.414^n)$ time. Both the bipartite and the general algorithm can be implemented to use space polynomial in n .

1.1.3 Importance of Hamiltonian Cycle in Distributed Computing

Fault tolerance is an important aspect of parallel and distributed systems. Two major kinds of hardware faults that can occur in networks are dead processor fault (due to failure of processor or support chip) and dead interprocessor communication (due to failure of communication hardware). These faults can be abstracted as the failure of nodes and of edges in the underlying network graph. Hence, an important parameter to measure the fault tolerance of a distributed system, is to count the number of Independent Spanning Trees in the graph, which ensures the presence of parallel, node-disjoint paths between nodes of the network. Let us briefly define Independent Spanning Trees here. For a tree T and $x, y \in V$, let $T[x, y]$ denote the unique path from x to y in T . A rooted tree is a tree with a specified vertex called the root of T . Let G be a graph, let $r \in V$, and let T and T' be trees of G rooted at r. We say that T and T' are independent if for every $x \in V(T) \cap V(T')$, the paths $T[r, x]$, $T'[r, x]$ have no vertex in common except r and x. Hamiltonicity is an important property in any hierarchical interconnection network that is closely related to fault

tolerance, as, the presence K edge disjoint Hamiltonian cycles in a network implies $2K$ Independent Spanning Trees in that network. Hamiltonicity is also important to ensure deadlock freedom in some routing algorithms [Carpenter, 1990] and to allow efficient emulation of linear-array and ring algorithms. Algorithms, such as all-to-all broadcasting or total exchange, relies on a Hamiltonian cycle for its efficient execution [Parhami, 2006].

1.2 Longest Path Problem

1.2.1 Entanglement Puzzle game

One of the most popular computer games, hosted by Google chrome webstore is the Entanglement puzzle game made by the Gopherwood Studios. The game requires players to try to make the longest path possible by rotating and placing hexagonal tiles etched with paths to extend the path without running into walls. This is constrained version of the well known Longest Path problem.

1.2.2 Formal Introduction

The longest path problem is the problem of finding a simple path of maximum length in a graph. It is easy to see that Hamiltonian Path problem is a special case of Longest Path problem. As Hamiltonian path problem is NP-Hard on general graphs, so obviously solving the longest path problem is also NP-Hard. In fact, it has been shown that there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [Karger *et al.*, 1997].

The Hamiltonian Path problem remains NP-complete even when restricted to some small classes of graphs such as split graphs, chordal bipartite graphs, strongly chordal graphs, circle graphs, planar graphs, and grid graphs [Golumbic, 2004] [Ioannidou *et al.*, 2009]. However it becomes polynomial time solvable on certain classes of

graphs, like, interval graphs, co-comparability graphs, circular-arc graphs and convex bipartite graphs [Ioannidou *et al.*, 2009] and it is meaningful to investigate the Longest Path problem on special graph classes, for which Hamiltonian Path problem is polynomial time solvable. But surprisingly, there are a very few known polynomial time solutions for the Longest Path problems.

1.2.3 Application of Longest Path Algorithms

Longest path algorithms find various applications across diverse fields. The well-known Traveling Salesman problem is also a special case of Longest Path problem [Hardgrave and Nemhauser, 1962]. The longest path in program activity graph is known as critical path (Fig 1.4) which represents the sequence of program activities that take the longest time to execute. Longest path algorithm is required to calculate critical paths. The critical path method is as follows:[Sedgewick and Wayne, 2002]
We consider the parallel precedence-constrained job scheduling problem: Given a

job	*duration*	*must complete before*
0	41.0	1 7 9
1	51.0	2
2	50.0	
3	36.0	
4	38.0	
5	45.0	
6	21.0	3 8
7	32.0	3 8
8	32.0	2
9	29.0	4 6

A job-scheduling problem

Figure 1.3: Jobs and precedence

set of jobs of specified duration to be completed, with precedence constraints that specify that certain jobs have to be completed before certain other jobs are begun, how can we schedule the jobs on identical processors (as many as needed) such that they are all completed in the minimum amount of time while still respecting the constraints? This problem can be solved by formulating it as a longest paths problem

in an edge-weighted Directed Acyclic Graph (DAG): Create an edge-weighted DAG with a source s, a sink t, and two vertices for each job (a start vertex and an end vertex). For each job, add an edge from its start vertex to its end vertex with weight equal to its duration. For each precedence constraint v→w, add a zero-weight edge from the end vertex corresponding to v to the beginning vertex corresponding to w. Also add zero-weight edges from the source to each job's start vertex and from each job's end vertex to the sink. Now, schedule each job at the time given by the length of its longest path from the source.

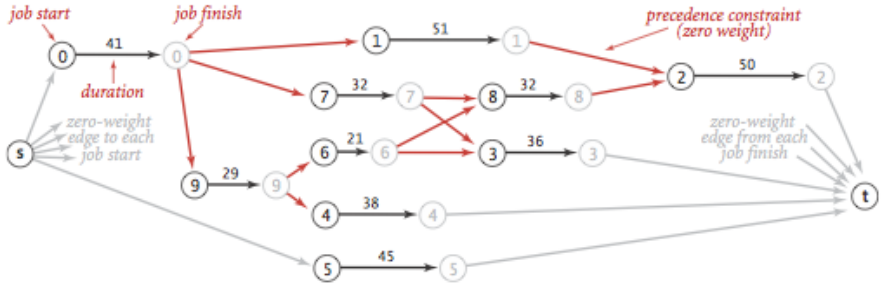

Figure 1.4: Critical Path on Program Activity Graph

1.3 Contribution of this Thesis

We organize the thesis in two parts[2].

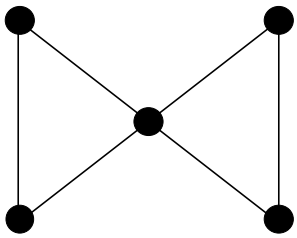

Figure 1.5: Butterfly or Bowtie network

[2]All definitions and notations used here and in the subsequent sections are defined in Chapter 2

- In the first part, we address some important aspects of **Hamiltonicity of Optical Transpose Interconnection System networks.**

  - We investigate whether Hamiltonicity of base graph is also a necessary condition for the OTIS to be Hamiltonian. We answer this in negative.

  - We further investigate whether it is sufficient for the base graph to have Hamiltonian path, for the OTIS to be Hamiltonian. We answer this also in negative.

  - Two kinds of butterfly graphs are known in literature. The first one is a 5-vertex graph which is also known as bowtie graph. We consider the generalization of this butterfly/bowtie graph, where we consider two cycles C_m, C_n connected at a cutvertex and denote in as $BF(n, m)$. We consider $BF(n, m)$ as base network, and investigate the Hamiltonicity on the OTIS network. To avoid ambiguity, we denote the OTIS network formed on $BF(n, m)$ as Bowtie-OTIS.

  - A different type of butterfly graph of dimension n is defined as a 4-regular graph, $BF(n)$, on $n2^n$ vertices as follows [Barth and Raspaud, 1994]:

    * The vertex set, $V(BF(n))$ is the set of couples $(\alpha; x_{n-1}, \ldots, x_0)$, where $\alpha \in \{0, \ldots, (n-1)\}$ and $x_i \in \{0, 1\}, \forall i \in \{0, \ldots, (n-1)\}$.
    * $[(\alpha; x_{n-1}, \ldots, x_0), (\alpha'; x_{n-1}', \ldots, x_0')]$ is an edge of $BF(n)$ if $\alpha' \equiv \alpha + 1 (\text{mod } n)$ and if $x_i = x_i' \forall i \neq \alpha'$.

    We also investigate Hamiltonicity on the OTIS network built on this base network.

  - We give constructive proofs for Hamiltonicity, on Bowtie-OTIS of $BF(2m+1, 2n+1)$ and $BF(2m+1, 2k)$, where $m, n, k \in \mathbb{N}$. This construction leads to an efficient alternate linear time rooted Independent Spanning Tree construction algorithm on this class of Bowtie-OTIS graphs. This algorithm is linear in the number of vertices, as opposed to the generalized tree construction algorithm was proposed by Itai and Rodeh [Itai and

Rodeh, 1988], which is linear in number of edges of the graph. So if we make $BF(2m+1, 2n+1)$ or $BF(2m+1, 2k)$ denser by introducing chords inside the cycles C_{2m+1}, C_{2n+1} or C_{2k} such that at least one of the vertices retain degree 2, our algorithm shows better performance than the generalized one.

- In the second part, we present **A Polynomial Time Algorithm for Longest Paths in Biconvex Graphs.** We also present a more efficient approximation algorithm with Constant additive error to find longest path on biconvex graphs. More formally, we can say, let L_{max} be the longest path length declared by the approximation algorithm and L'_{max} be the correct longest path length on the given graph. Then either $L_{max} = L'_{max}$ or $L_{max} = L'_{max} - 1$

  Most of the graph theoretic problems of practical interest, in their fullest generality, are NP-Complete, which means, it is very unlikely that efficient polynomial time solution can be found for these problems. One of the most popular and effective approaches to tackle with NP-Completeness is to restrict the class of inputs. Exploiting this approach, we can see that certain intractable problems become polynomial time solvable on some special graph classes.On a similar line of approach, we present a polynomial time algorithm [$O(n^6)$ time algorithm] to find the Longest Path on Biconvex Graphs, where n is the number of vertices of the input graph. Polynomial solutions for this problem are known only for special classes of graphs, while it is NP-hard on general graphs. Biconvex graphs are superclass of bipartite permutation graphs,[Brandstädt *et al.*, 1999] on which the Longest Path problem is polynomial time solvable, and subclass of chordal bipartite graphs [Spinrad *et al.*, 1987], on which, this problem is NP-Hard. Naturally it was interesting to investigate the status of the problem on Biconvex graph class.

1.4 Outline

In this section we present a brief outline of the thesis. In Chapter 2, we discuss the basics of complexity classes, approximation algorithms, some basics of graph theory and special graph classes and their characteristics. In Chapter 3, we present the literature survey and our results on Fault Tolerance and Hamiltonicity of Optical Transpose Interconnection Systems. In Chapter 4, we present literature survey and our results on the Longest Path problem on Biconvex graphs. In Chapter 5 we conclude mentioning some related open problems and further direction that can be pursued.

CHAPTER 2

Preliminaries

In this chapter, we discuss basic definitions and theorems in computational complexity and graph theory. In Section 2.1, we discuss about basic definitions in graph theory. In Section 2.2, we discuss about basic complexity classes P and NP. In Section 2.3, we briefly discuss about approximation algorithms. In Section 2.4 we define Biconvex Graphs and in Section 2.5 we define Optical Transpose interconnection Systems.

2.1 Graph Theoretic Preliminaries

Now we move to graph theoretic basics. In this thesis we work with simple and undirected graphs.

A graph $G = (V, E)$ consists of a finite set $V(G)$ of vertices and a collection $E(G)$ of 2-element subsets of $V(G)$ called edges. An undirected edge is a pair of distinct vertices $u, v \in V(G)$, and is denoted by uv. We say that the vertex u is adjacent to the vertex v or, equivalently, the vertex u sees the vertex v, if there is an edge $uv \in E(G)$. Let S be a set of vertices of a graph G. Then, the cardinality of the set S is denoted by $|S|$ and the subgraph of G induced by S is denoted by $G[S]$. The set $N(v) = \{u \in V(G) : uv \in E(G)\}$ is called the neighborhood of the vertex $v \in V$ in G, the set $N[v] = N(v) \cup \{v\}$ is called the closed neighborhood of the vertex $v \in V(G)$. A simple path of a graph G is a sequence of distinct vertices $v_1, v_2, \cdots, v_k$ such that $v_i v_{i+1} \in E(G)$ $\forall 1 \leq i \leq k-1$ and is denoted by $(v_1, v_2, \cdots, v_k)$. Throughout the paper all paths considered are simple. For a vertex $v \in V(G)$, by $deg(v)$ we shall denote the degree of v in G. The maximum degree among the vertices of G is denoted

by $\Delta(G)$ and the minimum degree by $\delta(G)$. $diam(G)$ denote the diameter of G and it is defined as the maximal distance between any two nodes in G. The connectivity of G, $\kappa(G)$ denotes the minimum number of vertices, which when removed, disconnects G.

2.2 Basic Complexity Classes P and NP

The theory of NP-Completeness is designed to be applied to *decision problems* only. Before formally introducing the notion of NP-Completeness, let us briefly discuss about decision and optimization version of a problem.

Decision problems have only two possible solutions, either the answer is "yes" or the answer is "no". If the optimization problem asks for a structure of certain type that has "minimum" cost among all such structures, we can associate with that problem, the decision problem that includes a numerical bound k as an additional parameter and asks whether there exists a structure of the required type having cost *no more than* k. Let us illustrate with an example. Vertex Cover is a well known graph theoretic problem. The problem is the following:

Definition: A vertex-cover of an undirected graph $G = (V, E)$ is a subset of V' of V such that $\forall(u, v) \in E$ either u or v (or both) belongs to V' .

Problem: Finding minimum size vertex cover in a given undirected graph.

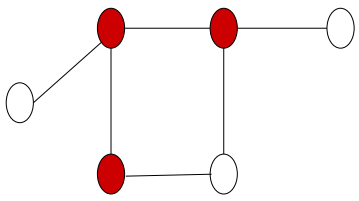

Figure 2.1: The vertices shown in dark shade constitute the minimum vertex cover

The *optimization version* of this problem is: What is the size of minimum Vertex Cover for a given graph G?
The *decision version* of this problem is: Does there exist a Vertex Cover of size k for a given graph G, where k is a fixed integer?

Decision problems can be derived from maximization problems in an analogous way, simple by replacing "no more than" by "at least".

2.2.1 NP-Hardness and NP-Completeness

The notations followed in this section are from [Papadimitriou, 1994]. For basic definitions like Turing machines refer [Papadimitriou, 1994; Garey and Johnson, 1979]. The set of problems for which polynomial time algorithms exists belong to class P or the set of all languages decidable in polynomial time by Turing machines is denoted by P [Papadimitriou, 1994] i.e., $P = \mathbf{TIME}(n^k)$ where n is size of input and $k \geq 1$. The set of languages decided by nondeterministic Turing machines within time n^k is denoted by NP i.e., $NP = \mathbf{NTIME}(n^k)$ where n is the size of the input and $k \geq 1$. Famous open question is $P \stackrel{?}{=} NP$.

Definition 2.1 *[Papadimitriou, 1994] Let $\mathcal{C}$ be a complexity class. We say that L is **$\mathcal{C}$-hard** if any language $L' \in \mathcal{C}$ can be reduced to L.*

Definition 2.2 *[Papadimitriou, 1994] Let $\mathcal{C}$ be a complexity class, and let L be a language in $\mathcal{C}$. We say that L is **$\mathcal{C}$-complete** if any language $L' \in \mathcal{C}$ can be reduced to L.*

A problem p is NP-complete if $p \in NP$ and p is NP-hard. Cook introduced the first NP-complete problem, the **Boolean Satisfiability**, in 1971. Now there exists thousands of NP-complete problems. A polynomial time lower bound for any one of these problems would imply that $P = NP$. A super polynomial lower bound for any one of these problems would imply that $P \neq NP$.

2.3 Approximation Algorithms

In this section, we look at some basic definitions related to approximation algorithms. If we know that a problem is NP-hard then we can work on the problem in the following ways:

1. Restricted class of Inputs: Here the input is restricted to a subset of the original class. Sometimes a problem can be solved in polynomial time if we restrict the input to subset of the original class. For example VERTEX COVER is NP-complete on general graphs. If we restrict our input to bipartite graphs this can be solved in polynomial time.

2. Approximation Algorithms: Here we design an algorithm which finds an approximate solution with a small error bound instead of finding the exact solution.

Let P be a problem and I be an instance of the problem and $F^*(I)$ be the value of an optimal solution to I. An approximate algorithm generally produces a feasible solution to I whose values $F(I)$ is less than (or greater than) $F^*(I)$ if P is a maximization (minimization) problem.

Definition 2.3 *[Sahani* et al.*, 2007] Let P be a problem and I be an instance of the problem and $F^*(I)$ be the value of an optimal solution to I. An approximate algorithm generally produces a feasible solution to I whose values $F(I)$ is less than (or greater than) $F^*(I)$ if P is a maximization (minimization) problem.*

An optimization problem can be either a maximization or a minimization problem.

Definition 2.4 *[Sahani* et al.*, 2007] A is an absolute approximation algorithm for problem P if and only if for every instance I of P, $|F^*(I) - F(I)| \leq C$ for some constant C.*

Definition 2.5 *[Sahani* et al., *2007] A is an $\rho(n)$-approximation algorithm for problem P, if and only if for every instance I of size n, $|F^*(I) - F(I)|/F^*(I) \leq \rho(n)$ for $F^*(I) > 0$.*

For some problems it is possible to develop approximation algorithms with constant approximation difference and for some problems the best possible polynomial time approximation ratio depends on the size of the input.

2.4 Biconvex Graphs

A bipartite graph G=(S, T, E) is convex on the vertex set S if S can be ordered so that for each element t in the vertex set T the elements of S connected to t form an interval of S; this property is called the adjacency property. G is biconvex if it is convex on both S and T [Abbas and Stewart, 2000*a*]. Fig 2.2 demonstrates an example. The vertices which violate adjacency property, i.e., whose neighborhood does not form an interval in the other partition, are shown in bold. The dotted edges are not present in the graph. This illustrates one possible ordering and there can be other possible orderings as well. However recognition of this graph class and achieving an ordering preserving the adjacency property are polynomial time [Abbas and Stewart, 2000*a*].

2.5 Optical Transpose Interconnection Network

We will use standard graph theoretic terminology. Let $G = (V, E)$ be a finite undirected simple graph with vertex set $V(G)$ and edge set $E(G)$.

The OTIS network denoted as $OTIS(G)$, derived from the base or basis or factor graph $G_B = (V_B, E_B)$, is a graph with vertex set:

$$V(OTIS(G_B)) \triangleq \{\langle u, v\rangle \,|u, v \in V(G_B)\}\,,$$

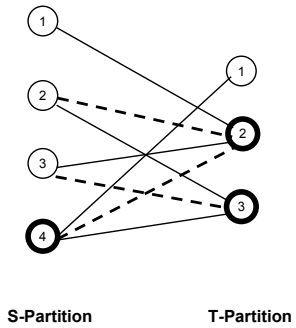

Figure 2.2: The vertices which violate adjacency property are shown in bold.

And edge set:

$$E(OTIS(G_B)) \triangleq \{(\langle v,u\rangle, \langle v,u'\rangle) | v \in V(G_B), (u,u') \in E(G_B)\} \cup \{(\langle v,u\rangle, \langle u,v\rangle) | u,v \in V(G_B), u \neq v\}.$$

If the basis network G_B has n nodes, then $OTIS(G)$ is composed of n node-disjoint subnetworks called clusters, each of which is isomorphic to G_B (Fig 2.3). We assume that the processor/nodes of the basis network is labeled $[n] = \{1, \ldots, n\}$, and the processor/node label $\langle g, u\rangle$ in $OTIS$ network $OTIS(G)$ identifies the node indexed u in cluster g, and this corresponds to vertex $\langle g, u\rangle \in V((G_B))$. Subsequently, we shall refer to g as the cluster address of node $\langle g, u\rangle$ and u as its processor address.

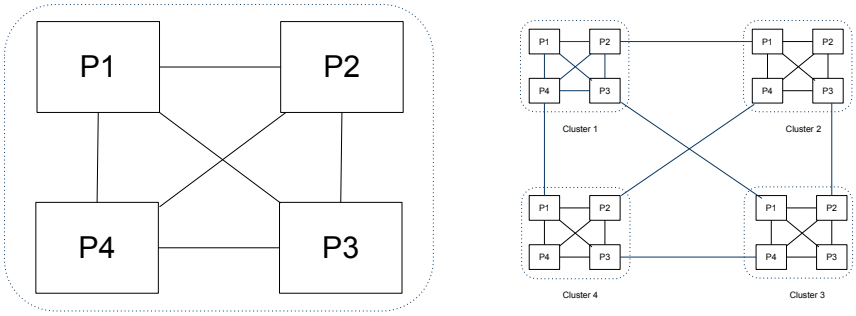

Figure 2.3: (a) Base Network(b) OTIS network

CHAPTER 3

Fault Tolerance and Hamiltonicity of the Optical Transpose Interconnection System

3.1 Optical Transpose Interconnection Systems

Optical Transpose Interconnection Systems (OTIS) is a widely studied interconnection network topology in parallel and distributed computing. OTIS(Swapped) Network was first proposed by Marsden et al. in 1993 [Marsden *et al.*, 1993]. A number of computer architectures have subsequently been proposed in which the OTIS concept was used to connect new optoelectronic computer architectures efficiently exploiting both optical and electronic technologies. In this architecture, processors are divided into groups (called clusters), where processors within the same group are connected using electronic interconnects, while optical interconnects are used for intercluster communication. The OTIS architecture has been used to propose interconnection networks for multiprocessor systems. Krishnamoorthy et al. have shown that the power consumption is minimized and the bandwidth rate is maximized when the number of processors in a cluster equals the number of clusters [Krishnamoorthy *et al.*, 1992].

3.1.1 Related Results

OTIS (Swapped) have been extensively studied. Chen et al. have shown if the base graph is k connected than OTIS will have k-vertex disjoint paths between any pair of vertices, and this is defined as a notion of maximal fault tolerance by them [Chen *et al.*, 2009]. Surprisingly, to our knowledge, very few results are known regarding Hamiltonicity of OTIS networks. The only significant result known about Hamiltonicity of OTIS, is by Parhami, that proves that OTIS networks built of Hamiltonian basis networks are Hamiltonian [Parhami, 2005]. The result by Hoseinyfarahabady et al. [Hoseinyfarahabady and Sarbazi-Azad, 2007] shows that the OTIS-Network is Pancyclic and hence Hamiltonian, if its base network is Hamiltonian-connected. However, by the fact that any Hamiltonian connected base graph is definitely Hamiltonian, this is a weaker result.

3.1.2 Our Contribution

We address some important aspect of Hamiltonicity on OTIS graphs.

- We investigate whether Hamiltonicity of base graph is also a necessary condition for the OTIS to be Hamiltonian. We answer this in negative.
- We further investigate whether it is sufficient for the base graph to have Hamiltonian path, for the OTIS to be Hamiltonian. We answer this in negative as well.

- Two kinds of butterfly graphs known in literature. The first one is a 5-vertex graph (Fig 3.1) which is also known as bowtie graph. We consider the generalization of this butterfly/bowtie graph, where we consider two cycles C_m, C_n connected at a cutvertex and denote in as $BF(n, m)$. We consider $BF(n, m)$ as base network, and investigate the Hamiltonicity on the OTIS network. To avoid ambiguity, we denote the OTIS network formed on $BF(n, m)$ as Bowtie-OTIS.

- A different type of butterfly graph of dimension n is defined as a 4-regular graph, $BF(n)$, on $n2^n$ vertices as follows [Barth and Raspaud, 1994]:

  - The vertex set, $V(BF(n))$ is the set of couples $(\alpha; x_{n-1}, \ldots, x_0)$, where $\alpha \in \{0, \ldots, (n-1)\}$ and $x_i \in \{0, 1\}, \forall i \in \{0, \ldots, (n-1)\}$.

  - $[(\alpha; x_{n-1}, \ldots, x_0), (\alpha'; {x_{n-1}}', \ldots, {x_0}')]$ is an edge of $BF(n)$ if $\alpha' \equiv \alpha+1 (\mathrm{mod}\ n)$ and if $x_i = {x_i}' \forall i \neq \alpha'$.

  We also investigate Hamiltonicity on the OTIS network built on this base network.

- We give constructive proofs for Hamiltonicity, on Bowtie-OTIS of $BF(2m + 1, 2n + 1)$ and $BF(2m + 1, 2k)$, where $m, n, k \in \mathbb{N}$. This construction leads to an efficient alternate linear time Independent Spanning Tree construction algorithm on this class of Bowtie-OTIS graphs. This algorithm is linear in the number of vertices, as opposed to the generalized tree construction algorithm proposed by Itai and Rodeh [Itai and Rodeh, 1988], which is linear in number of edges of the graph. So if we make $BF(2m+1, 2n+1)$ or $BF(2m+1, 2k)$ denser

by introducing chords inside the cycles C_{2m+1}, C_{2n+1} or C_{2k} such that at least one of the vertices retain degree 2, our algorithm shows better performance than the generalized one.

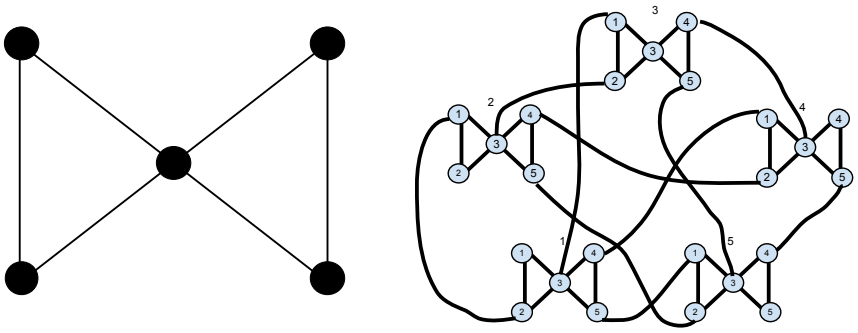

Figure 3.1: (a) Butterfly or Bowtie Graph $BF(3,3)$ (b) OTIS on $BF(3,3)$

3.1.3 Organization

We will discuss our results under the following sections. In section 3.2, we give some preliminaries, in section 3.3 we give a brief outline of our work, in section 3.4, we discuss the proof for Hamiltonicity on $OTIS(BF(2m+1,2n+1))$ and $OTIS(BF(2m+1,2k))$ thereby, proving that Hamiltonicity of base graph is not a necessary condition for the OTIS to be Hamiltonian. We further show that OTIS network built on the other class of Butterfly graphs, [Barth and Raspaud, 1994] is Hamiltonian. In section 3.5 we prove that $OTIS(BF(4,4))$ and $OTIS(BF(4,6))$ are non-Hamiltonian, which proves that it is not sufficient for the base graph to have Hamiltonian path, for the OTIS to be Hamiltonian. We discuss our algorithm to create two Independent Spanning Trees in time linear in the number of vertices in section 3.6.

3.2 Preliminaries

We will use standard graph theoretic terminology. Let $G = (V, E)$ be a finite undirected simple graph with vertex set $V(G)$ and edge set $E(G)$. For a vertex $v \in V(G)$, by $deg(v)$ we shall denote the degree of v in G. The maximum degree among the vertices of G is denoted by $\Delta(G)$ and the minimum degree by $\delta(G)$. $diam(G)$ denote the diameter of G and it is defined as the maximal distance between any two nodes in G. The connectivity of G, $\kappa(G)$ denotes the minimum number of vertices, which when removed, disconnects G. The OTIS network denoted as $OTIS(G)$, derived from the base or basis or factor graph $G_B = (V_B, E_B)$, is a graph with vertex set:

$$V(OTIS(G_B)) \stackrel{\Delta}{=} \{\langle u, v\rangle \,|u, v \in V(G_B)\}\,,$$

And edge set:

$$E(OTIS(G_B)) \stackrel{\Delta}{=} \{(\langle v, u\rangle\,, \langle v, u'\rangle)|v \in V(G_B), (u, u') \in E(G_B)\} \cup$$
$$\{(\langle v, u\rangle\,, \langle u, v\rangle)|u, v \in V(G_B), u \neq v\}\,.$$

If the basis network G_B has n nodes, then $OTIS(G)$ is composed of n node-disjoint subnetworks called clusters, each of which is isomorphic to G_B. We assume that the processor/nodes of the basis network is labeled $[n] = \{1, \ldots, n\}$, and the processor/node label $\langle g, u\rangle$ in $OTIS$ network $OTIS(G)$ identifies the node indexed u in cluster g, and this corresponds to vertex $\langle g, u\rangle \in V((G_B))$. Subsequently, we shall

refer to g as the cluster address of node $\langle g, u\rangle$ and u as its processor address.

The vertices of the base graph $BF(m,n)$, $[m,n \in \mathbb{N}]$ of $OTIS(BF(m,n))$, is labeled with indices $\{1,2,\ldots,c,c+1,\ldots,i\} \subset \mathbb{N}$, where c denotes the label of the cutvertex, and i denotes the label of the last vertex in the base graph and hence, $i = |V(G_B)|, m = c, n = i - c + 1$. (Fig 3.2)

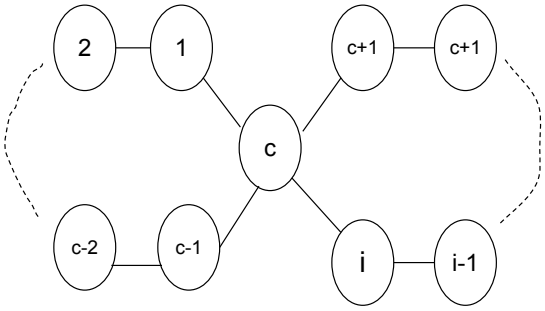

Figure 3.2: Labeling $BF(m,n)$, where $i = |V(G_B)|, m = c, n = i - c + 1$

To enhance readability, sometimes we will mention a cluster g and denote edges for which, both endpoints are within g, as (x,y) which denote edges $(\langle g,x\rangle, \langle g,y\rangle)$. A vertex is called "saturated" if its Hamiltonian neighbours, i.e, neighbours in a Hamiltonian Cycle, are explicitly identified.

Based on the existing results following properties hold for $OTIS(G)$:

Proposition 3.2.1 ([Chen *et al.*, 2009]) *Given basis graph $G = (V,E)$, with $|V| = n$, $\Delta(G) = \Delta$, $\delta(G) = \delta$, $diam(G) = d$, and $\kappa(G) = k$, following holds for $OTIS(G)$:*

1. $\deg(\langle u,v\rangle) = \deg(v) + 1$ *when* $u \neq v$, *and* $\deg(v)$ *otherwise.*

2. $\Delta(OTIS(G)) = \Delta + 1$.

3. $\delta(OTIS(G)) = \delta$.

4. $diam(OTIS(G)) = 2d + 1$.

3.3 Outline of the Work

We first investigate the Hamiltonicity of $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$, where $m, n, k \in \mathbb{N}$ and prove that both of them are Hamiltonian. We give explicit constructions of Hamiltonian Cycles on these two classes. Thus we answer the question that, the base graph need not be Hamiltonian, for the OTIS-network to be Hamiltonian, as the generalized bowtie graphs, $BF(2m+1, 2n+1)$ and $BF(2m+1, 2k)$ are clearly Non-Hamiltonian.

Lemma 3.1 *Number of edge-disjoint Hamiltonian Cycles on a simple graph with minimum degree* δ *is at most* $\lfloor \frac{\delta}{2} \rfloor$.

Proof: Any vertex v has $deg(v)$ number of edges incident on it. If possible, let there be H_i number of edge-disjoint Hamiltonian cycles on the graph. Each Hamiltonian Cycle will use exactly two of the $deg(v)$ edges incident on vertex v. Hence, v can be included in at most $\frac{deg(v)}{2}$ Hamiltonian Cycles, if $deg(v)$ is even, $\frac{deg(v)-1}{2}$ Hamiltonian Cycles, if $deg(v)$ is odd. Hence it is easily seen that H_i is upperbounded by $\lfloor \frac{\delta}{2} \rfloor$.

The crucial observation that will be exploited for Hamiltonian Cycle construction on $OTIS(BF(2m + 1, 2n + 1))$ and $OTIS(BF(2m + 1, 2k))$ is the following:

Observation 3.3.1 *There are only 4 kinds of vertex-degrees in the Bowtie-OTIS,* $2, 3, 4, 5$ *and the Bowtie-OTIS is 2-edge connected. Also there is exactly one vertex of degree* 4*, namely* $\langle c, c\rangle$*, exactly* $(|V_B| - 1)$ *vertices of degree* 2 *(*$\langle x, x\rangle$ *where* $x \neq c$*) and* $(|V_B| - 1)$ *vertices of degree 5 (*$\langle x, c\rangle$ *:* $\forall x \in (\{1, 2, \ldots, |V_B|\}\backslash c)$*) . Rest of the vertices are all of degree* 3*.*

The correctness of this observation follows from Proposition 3.2.1

Using Lemma 3.1 and Observation 3.3.1, it is easily seen the number of edge-disjoint Hamiltonian Cycles on $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$ can be at most $\lfloor \frac{2}{2} \rfloor = 1$. We give construction for Hamiltonian Cycle and discuss how these constructions can be used to generate two Independent Spanning Trees on $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$ in time linear in the number of vertices of the OTIS-network (in section 3.6).]

Next we address the question whether it is sufficient for the base graph to have Hamiltonian path, for the OTIS to be Hamiltonian. We answer this in negative, proving that the $OTIS(BF(4, 6))$ and $OTIS(BF(4, 4))$ are both Non-Hamiltonian. It is easy to see that the base graph, in both the cases, admits Hamiltonian Path.

Lastly, we consider the the OTIS network built of butterfly graph mentioned in [Barth and Raspaud, 1994].

3.4 Proof of Hamiltonicity of $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$

We give constructive proofs for both $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$. First we state two Inference Rules that will be used to construct Hamiltonian Cycles.

IR 1: If a vertex of degree ≥ 3, gets saturated, the rest of its edges, not used in the saturation, becomes Non-Hamiltonian edges and are deleted from the graph.

IR 2: If $(\langle g_1, u\rangle, \langle g_2, v\rangle)$ is an edge between the vertices $\langle g_1, u\rangle$ and $\langle g_2, v\rangle$, both of degree 3, and if the edge $(\langle g_1, u\rangle, \langle g_2, v\rangle)$ is identified as Non-Hamiltonian, then all other edges incident to the vertices $\langle g_1, u\rangle$ and $\langle g_2, v\rangle$ are forced to be Hamiltonian.

Once the edge $(\langle g_1, u\rangle, \langle g_2, v\rangle)$ is identified as Non-Hamiltonian, it is dropped from the potential set of edges required to construct Hamiltonian cycle. So now, exactly 2 potential edges are incident to each $\langle g_1, u\rangle$ and $\langle g_2, v\rangle$ and hence are forced to be Hamiltonian edges.

The steps in the construction are as follows:

Step 1: We identify the key Non-Hamiltonian edges whose endpoints lies within the same cluster, explicitly and delete them.

Step 2: In this process some vertices becomes saturated; we apply IR 1 on these vertices.

Step 3: The previous step, in turn decides Hamiltonian edges of the remaining vertices(due to IR 2).

Observation 3.4.1 *The constructions can be implemented as algorithm to construct Hamiltonian Cycles on* $OTIS(BF(2m+1, 2n+1))$ *and* $(OTIS(BF(2m+1, 2k))$ *in time* $O(m|V_B|)$*, i.e., in time linear in the number of vertices of the OTIS graph.* $[|V_B| =$ *number of clusters].*

This observation follows from the fact that, the number of intracluster edges, deleted per cluster, in these constructions, is $O(m)$, assuming $m > n, m > k$, without any loss of generality. In Step 1 of the construction, non-Hamiltonian intracluster edges are explicitly identified for all the clusters. Therefore, this step takes time proportional to the number of clusters, i.e., $O(|V_B|)$. Hence the Hamiltonian Cycle construction takes time $O(m|V_B|)$, i.e., in time linear in the number of vertices in the base graph.

3.4.1 Hamiltonicity of $OTIS(BF(2m + 1, 2n + 1))$

We identify the key non-Hamiltonian edges (Step 1 of the construction) in three parts. First we identify the key Non-Hamiltonian edges for $OTIS(BF(3, 2n + 1))$, $n > 1$. Then identify the key Non-Hamiltonian edges for $OTIS(BF(2m+1, 2m+1))$, $m > 1$. [1] Lastly, we identify the key Non-Hamiltonian edges for any $OTIS(BF(2m+1, 2n + 1))$, where $m > 1$, $n > 3$ and $n > m$.[2] This completes the proof that

[1] We give explicit construction for $m = 1$

[2] We give explicit construction for $m = 2, n = 3$

any $OTIS(BF(2m+1, 2n+1))$ $[m, n \in \mathbb{N}]$ is Hamiltonian. Note that, in these computations, the label 0 is same as label c.

Here we show the construction of $OTIS(BF(2m+1, 2m+1))$, $m > 1$ and argue its correctness. The correctness of the constructions for $OTIS(BF(3, 2n+1))$, $n > 1$ and $OTIS(BF(2m+1, 2n+1))$, where $m > 1$, $n > 3$ and $n > m$ can be argued similarly.

Key non-Hamiltonian edges for $OTIS(BF(c = 3, 2n+1))$

We determine the key non-Hamiltonian intracluster edges for each cluster.

Cluster 1: The set $S_1 = \{(c{+}2, c{+}3), (c{+}4, c{+}5), \ldots, (i{-}2, i{-}1)\}$ and $(c, i), (c, c{-}1)$ and $(c, c+1)$.

Cluster 2: $(c, 1)$ and $(c, c+1)$ and the set $S_2 = \{(6, 7), (i-3, i-2)\}$ iff $7 > i$, else ignore this set.

Cluster 3: $(c, 1), (c, c+1)$.

Cluster $(c+1)$**:** $(c, 1), (c, c-1), (c, i)$ and $(c+2, c+3), (i-2, i-1)$

Cluster $(i-1)$**:** $(c, 1), (c, i)$ and the set $S_{(i-1)} = \{(4, 5), (6, 7), (i-3, i-2)\}$ iff $5 < (i-4)$, else ignore this set.

Also $\forall$ cluster $x, 1 \leq x \leq i$, delete edges $(x-2, x-1)$ and $(x+1, x+2)$.

Key non-Hamiltonian edges for $OTIS(BF(2m+1, 2m+1))$, $m > 1$.

We determine the key non-Hamiltonian intracluster edges for each cluster.

Cluster 1: The sets $S_1 = \{(2,3),(4,5),\ldots,(c,c-1)\}$, $S_2 = \{(c+2,c+3),(c+4,c+5),\ldots,(i-2,i-1)\}$ and $(c,i),(c,c-1)$ and $(c,c+1)$.

Cluster $(c+1)$: The sets $S_1 = \{(2,3),(4,5),\ldots,(c,c-1)\}$, $S_2 = \{(c+2,c+3),(c+4,c+5),\ldots,(i-2,i-1)\}$ and $(c,i),(c,c-1)$and $(c,1)$.

Cluster $(c-1)$: The set $S_3 = \{(2,3),(4,5),\ldots,(c-3,c-2)\}$, $(c,c+1),(c,1)$ and $(c+3,c+4)$ iff $(c+4) \neq i$.

Cluster $(c-2)$: The set $S_4 = \{(1,2),(3,4),\ldots,(c-1,c)\}$, $(c,c+1)$ and $(c+3,c+4)$ iff $(c+4) \neq i$.

Cluster i: The set $S_5 = \{(c+2,c+3),\ldots,(c-3,c-2)\}$, $(c,c+1),(c,1)$ and $(3,4)$ iff $(c-1) \neq 4$.

Cluster $(i-1)$: The set $S_6 = \{(c+1,c+2),\ldots,(c,i)\}$, $(c,1)$ and $(3,4)$ iff $(c-1) \neq 4$.

For Clusters $\{2,4,6,\ldots(c-1)\}$ and $\{(c+2),(c+4),\ldots,i\}$, delete edges $(c,1)$ and $(c,c+1)$.

For Clusters $\{1,3,5,\ldots(c-2)\}$ delete edges $(c,c-1)$ and $(c,c+1)$.

For Clusters $\{(c+1),(c+3),\ldots(i-1)\}$ delete edges $(c,1)$ and (c,i).

Also $\forall$ cluster $x, 1 \leq x \leq i$, delete edges $(x-2,x-1)$ and $(x+1,x+2)$.

Correctness Argument for Hamiltonicity for $OTIS(BF(2m+1, 2m+1))$, $m > 1$.

Claim 3.1 *All the Hamiltonian edges of $OTIS(BF(2m+1, 2m+1))$ can be inferred by deleting the Key edges mentioned and using the inference rules IR 1 and IR 2.*

Proof: First we concentrate on the clusters $\{2, 4, 6, \ldots (c-1)\}$.

1. We mark the intercluster edges $(\langle 1, p\rangle, \langle p, 1\rangle)$, $(\langle c-1, p\rangle, \langle p, c-1\rangle)$ and $(\langle c-2, p\rangle, \langle p, c-2\rangle)$ as Hamiltonian edges (Using IR 1) $\forall p \in \{2, 4, 6, \ldots (c-1)\}$.

2. In clusters $\{3, 5, 7, \ldots (c-2)\}$, applying IR 2 for the vertex 2, we infer that the intercluster edges $(\langle 2, p\rangle, \langle p, 2\rangle)$ $\forall p \in \{3, 5, 7, \ldots (c-2)\}$ are Hamiltonian edges.

3. We also know that $(\langle p, x-2\rangle, \langle x-2, p\rangle)$, $(\langle p, x-1\rangle, \langle x-1, p\rangle)$, $(\langle p, x+2\rangle, \langle x+2, p\rangle)$, $(\langle p, x+1\rangle, \langle x+1, p\rangle)$ and the edges $(\langle p, x-3\rangle, \langle p, x-2\rangle)$, $(\langle p, x+3\rangle, \langle p, x+2\rangle)$ are Hamiltonian edges (Using IR 2) $\forall p \in \{2, 4, 6, \ldots (c-1)\}$.

4. Using (2) and (3) and IR 2, we infer set of non-Hamiltonian edges in clusters $\{2, 4, 6, \ldots (c-1)\}$:

   - The set $S_{e1} = \{(x-2, x-1), (x-4, x-3), \ldots (2, 3)\}$ when $(x-2) \neq c$. Else ignore this set.[3]
   - The set $S_{e2} = \{(x+1, x+2), (x+3, x+4), \ldots (c-2, c-1)\}$ when $(x+1) \neq c$. Else ignore this set. [4]

[3] For Cluster 2, this set is ignored.
[4] For Cluster $(c-1)$, this set is ignored.

This completes the description of non-Hamiltonian edges within the clusters $\{2, 4, 6, \ldots (c-1)\}$, which decides all the Hamiltonian neighbours of the vertices within these clusters. Below we illustrate with the example of Cluster 2.

- Hamneighbour$\langle 2, 1\rangle = \langle 2, 2\rangle, \langle 1, 2\rangle$.
- Hamneighbour$\langle 2, 2\rangle = \langle 2, 1\rangle, \langle 1, 3\rangle$.
- Hamneighbour$\langle 2, 3\rangle = \langle 2, 2\rangle, \langle 3, 2\rangle$.
- Hamneighbour$\langle 2, 4\rangle = \langle 2, 5\rangle, \langle 4, 2\rangle$.
- Hamneighbour$\langle 2, 5\rangle = \langle 2, 4\rangle, \langle 5, 2\rangle$.

  $\vdots$

- Hamneighbour$\langle 2, (c-3)\rangle = \langle 2, (c-2)\rangle, \langle (c-3), 2\rangle$.
- Hamneighbour$\langle 2, (c+3)\rangle = \langle 2, (c+2)\rangle, \langle (2, (c+4)\rangle$.

  $\vdots$

- Hamneighbour$\langle 2, (i-1)\rangle = \langle 2, (i-2)\rangle, \langle (2, i\rangle$.
- Hamneighbour$\langle 2, i\rangle = \langle 2, (i-1)\rangle, \langle (2, c\rangle$.

By similar arguments, we infer set of non-Hamiltonian edges in clusters $\{1, 3, 5, \ldots (c-2)\}$, which are as follows:

- The set $S_{o1} = \{(x-2, x-1), (x-4, x-3), \ldots (1, 2)\}$ when $(x-2) \neq c$. Else ignore this set.[5]

[5]For Cluster 1, this set is ignored.

- The set $S_{o2} = \{(x+1, x+2), (x+3, x+4), \ldots (c-3, c-2)\}$ when $(x+1) \neq c$.

  Else ignore this set.

This completes the description of non-Hamiltonian edges within the clusters $\{1, 3, 5, \ldots (c-2)\}$. By symmetry, we can infer the set of non-Hamiltonian edges in clusters $\{(c+2), (c+4), \ldots, i\}$ and $\{(c+1), (c+3), \ldots (i-1)\}$.

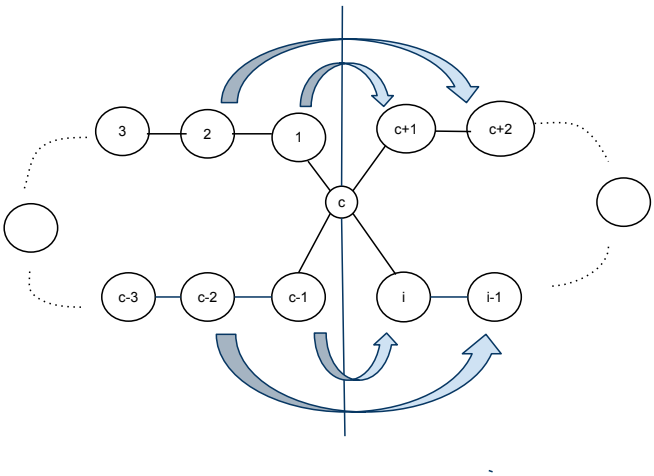

Figure 3.3: $BF(2m+1, 2m+1)$: This figure shows how the labels of the vertices with with symmetric local and global behavious can be mapped to each other. The bold solid arrows represent the mapping.

Hence, all the Hamiltonian edges of $OTIS(BF(2m+1, 2m+1))$ can be inferred using be deleting the Key edges mentioned and using the inference rules IR 1 and IR 2.

Key non-Hamiltonian edges for $OTIS(BF(2m+1, 2n+1))$, where $m > 1$, $n > 3$ and $n > m$

Cluster 1: $(c, c-1), (c, c+1), (c, i)$ and the sets $S_{1^l} = \{(2,3), (4,5), \ldots, (c-1, c)\}$. $S_{1^r} = \{(c+2, c+3), (c+4, c+5), \ldots, (i-2, i-1)\}$.

Cluster 2: $(c, 1), (c, i), (c-2, c-1)$and the set $S_2 = \{(c+5, c+6), (c+7, c+8), \ldots, (i-5, i-4)\}$ where $(i-5) \geq (c+5)$. Else ignore the set S_2.

Cluster 3: $(i-1, i-2), (c, c-1), (c, c+1)$ and the set $S_3 = \{(c+5, c+6), (c+7, c+8), \ldots, (i-5, i-4)\}$ where $(i-5) \geq (c+5)$. Else ignore the set S_3.

Cluster $4, \ldots, (c-3)$**:** $(c, 1), (c, c+1), (i-2, i-1)$

Cluster $(c-2)$**:** $(c, c-1), (c, c+1), (i-2, i-1)$.

Cluster $(c-1)$**:** $(c, 1), (c, i), (2, 3)$ and the set $S_6 = \{(c+4, c+5), (c+6, c+7), \ldots, (i-2, i-1)\}$

Cluster c**:** $(c, 1), (c, c+1)$.

Cluster $(c+1)$**:** $(c, 1), (c, c-1), (c, i)$ and $(c+2, c+3), (i-2, i-1)$.

Cluster $(c+2)$**:** $(c, c-1), (c, c+1), (i-1, i)$.

Cluster $(c+3)$**:** $(c, 1), (c, c+1), (i-1, i)$.

Cluster $(c+4)$**:** $(c, c-1), (c, 1), (i-1, i)$.

Cluster $(c+5)$ **to** $(i-4)$**:** $(2, 3)$ [only where $(i-5) \geq (c+5)$, else do not delete this edge.] $(c, 1), (c, c-1)$ and $(i, i-1)$.

Cluster $(i-3)$**:** $(c,1),(c,c-1),(i-1,i)$.

Cluster $(i-2)$**:** $S_{(i-2)} = \{(3,4),(5,6),\ldots,(c-2,c-1)\}$ and $(c,1),(c,c+1)$ and $(i,i-1)$.

Cluster $(i-1)$**:** The set $S_{(i-1)^l} = \{(3,4),(5,6),\ldots,(c-2,c-1)$ and $(c,1),(c,i)$ and the set $S_{(i-1)^r} = \{(c+1,c+2),(c+3,c+4),\ldots,(i-3,i-2)\}$.

Cluster i**:** $(1,2),(c,c-1),(c,c+1)$ and the set $S_i = \{(c+2,c+3),(c+4,c+5),\ldots,(i-2,i-1)\}$.

In addition to this, $\forall$ cluster $x, 1 \leq x \leq (c-1)$, delete edges $(x-2,x-1)$ and $(x+1,x+2)$.

3.4.2 Construction of Hamiltonian Cycle for $OTIS(BF(2m+1,2k))$

Here we identify the key non-Hamiltonian edges in two parts.

First we identify the key Non-Hamiltonian edges for $OTIS(BF(3,2k))$ and then for $OTIS(BF(2m+1,2k))$, where $m > 1$.

Key Non-Hamiltonian edges for $OTIS(BF(3,2k))$

Cluster 1: $(3,2),(3,4),(3,i)$

Cluster 2: $(3,1),(3,4)$ And the set $S_2 = \{(5,6),(7,8),\ldots,(i-1,i)\}$

Cluster 3: $(3,1),(3,4)$

Cluster 4: $(3,1),(3,2),(3,i)$

Cluster $5,6,\ldots,(i-1)$**:** $(3,2),(3,i)$

Cluster i**:** $(3,1),(3,2)$ and the set $S_i = \{(4,5),(6,7),\ldots,(i-2,i-1)\}$

Now join the intercluster edges at the at both endvertices of the deleted intercluster edges. This completes the Hamiltonian Cycle.

Illustration with $OTIS(BF(3,2k))$**,** $k=2$

The graph $OTIS(BF(3,4))$ is shown in Fig 3.4. First the specified set of intracluster edges as mentioned in 3.4.2 are removed (Fig 3.5). So the intracluster edges shown in Fig 3.5 are all forced edges, i.e., edges which are identified as Hamiltonian edges. Now, as we can see in Fig 3.6, IR 1 applies to all unsaturated vertices of Fig 3.5 and hence we complete the Hamiltonian Cycle by adding the intercluster edges forced by IR 1.

Key Non-Hamiltonian edges for $OTIS(BF(2m+1,2k))$**, where** $m>1$

Cluster 1: $(c,c-1),(c,c+1),(c,i)$ and the set $S_1 = \{(2,3),(4,5),(6,7),\ldots,(c-1,c)\}$.

Cluster 2: $(c,1),(c,c+1),(c-2,c-1)$ and the set $_2 = \{(c+2,c+3),(c+4,c+5),\ldots,(i-3,i-2)\}$ where $i \geq (c+3)$. Else ignore the set S_2.

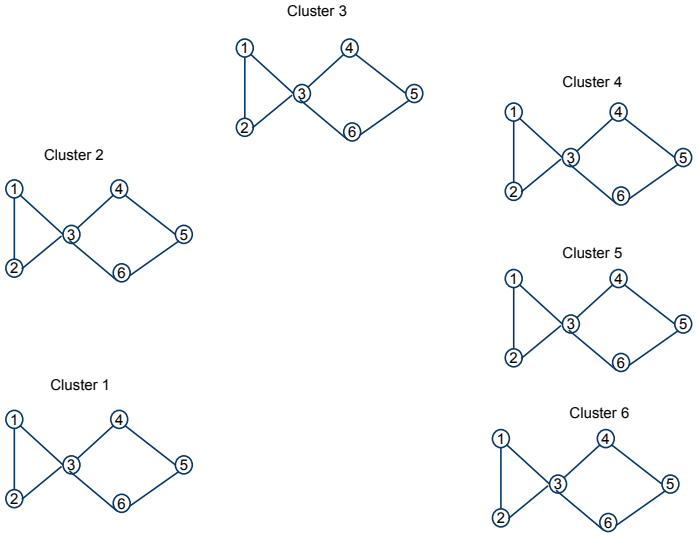

Figure 3.4: Intercluster edges are not shown explicitly to maintain clarity

Cluster 3: $(c, c-1), (c, c+1)$ and the set $S_3 = \{(c+2, c+3), (c+4, c+5), \ldots, (i-3, i-2)\}$ where $i \geq (c+3)$. Else ignore the set S_3. Delete $(i-1, i)$ if $4 < (c-1)$.

Cluster $3, 4, \ldots, (c-3)$**:** $(i-1, i)$ if $4 < (c-1)$

Cluster $4, 5, \ldots, (c-3)$**:** $(c, 1), (c, c+1)$ if $4 < (c-3)$

Cluster $(c-2)$**:** $(c, c-1), (c, c+1)$

Cluster $(c-1)$**:** $(c, 1), (c, i), (2, 3)$ and the set $S_{(c-1)} = \{(c+1, c+2), (c+3, c+4), \ldots, (i-2, i-1)\}$

Cluster c**:** $(c, 1), (c, c+1)$.

Cluster $(c+1)$**:** $(c, 1), (c, c-1), (c, i)$ and the set $S_{(c+1)} = \{(2, 3), (4, 5), \ldots, (c-3, c-2)\}$

Cluster $(c+2)$ **to** $(i-2)$**:** $(c, c-1), (c, i)$ and the edge $(2, 3)$ if $(c+3) < i$.

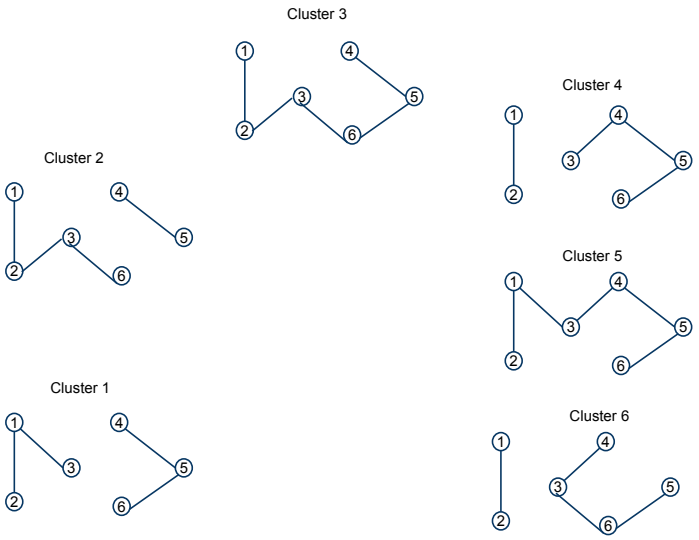

Figure 3.5: After execution of Step 1

Cluster $(i-1)$**:** $(c, c-1), (c, i)$ and the set $S_{i-1} = \{(3,4), (5,6), \ldots, (c-4, c-3)\}$ if $4 < (c-1)$

Cluster i**:** $(c, 1), (c, c-1)$ and the sets $S_{i^l} = \{(3,4), (5,6), \ldots, (c-4, c-3)\}$ and $S_{i^r} = \{(c+1, c+2), (c+3, c+4), \ldots, (i-2, i-1)\}$

In addition to this, $\forall$ cluster $x, 1 \leq x \leq (c-1)$, delete edges $(x-2, x-1)$ and $(x+1, x+2)$.

3.4.3 Explicit constructions for $OTIS(BF(3,3))$ and $OTIS(BF(5,7))$

We show the explicit constructions for $OTIS(BF(3,3))$ (Fig 3.7) and $OTIS(BF(5,7))$ (Fig 3.8).

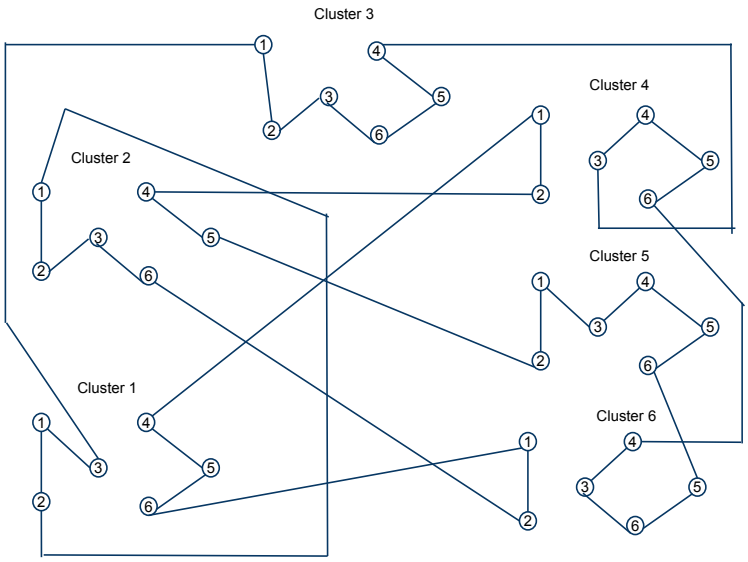

Figure 3.6: After execution of Step 2

3.4.4 $OTIS(BF(n))$ is Hamiltonian

It has been shown in the paper[Barth and Raspaud, 1994] that $BF(n)$ has two edge-disjoint Hamiltonian Cycles, by giving a recursive method of construction of the cycles. Hence the base network of $OTIS(BF(n))$ is Hamiltonian. Combining this result, with [Parhami, 2005], it is easily seen that $OTIS(BF(n))$ is Hamiltonian.

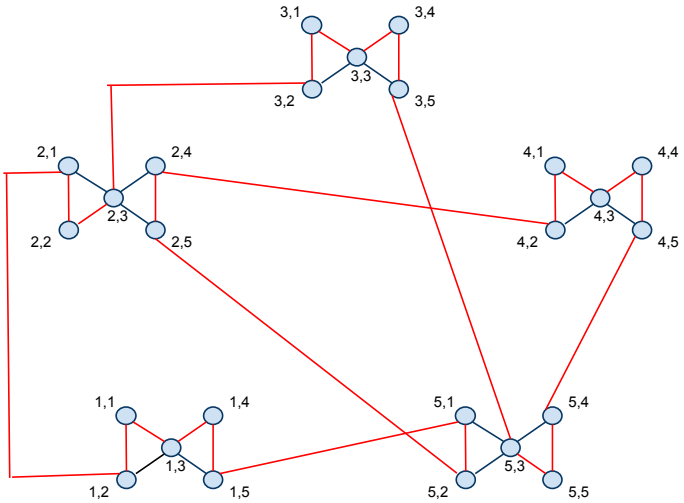

Figure 3.7: Hamiltonian Cycle on $OTIS(BF(3,3))$ shown in red color

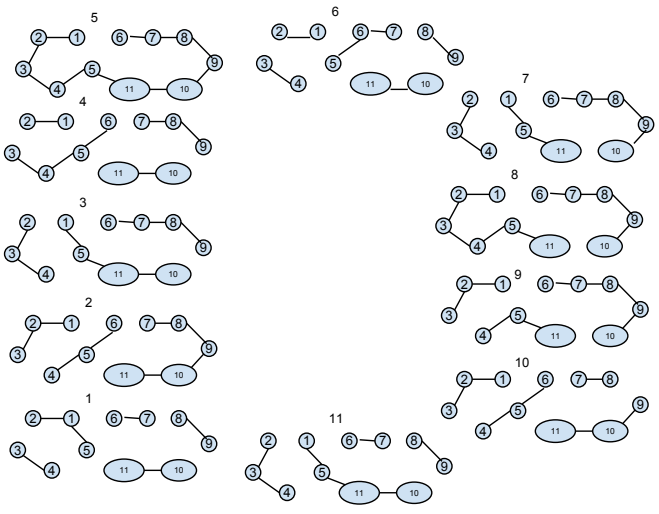

Figure 3.8: Joining the intercluster edges incident on the unsaturated vertices completes the Hamiltonian Cycle on $OTIS(BF(5,7))$

3.5 Proof of Non-Hamiltonicity of $OTIS(BF(4,4))$ and $OTIS(BF(4,6))$

3.5.1 Proof that $OTIS(BF(4,4))$ is not Hamiltonian

We present the proof of non-Hamiltonicity using counting argument.

First, let us count the total number of edges in the graph. Total number of edges in $OTIS(BF(4,4))$ is $\frac{\sum_{i=1}^{n} d_i}{2} = 77$, where d_i denotes degree of vertex i and $n = |V(OTIS(BF(4,4)))|$. If a Hamiltonian Cycle exists, it will use up 49 edges, as $n = 49$. So there are exactly $(77 - 49) = 28$ non-Hamiltonian edges in the graph.

Now, let us count the number of non-Hamiltonian edges in a different way. Let us look into $OTIS(BF(4,4))$ carefully. There are six degree 5 vertices, namely, $< 1,4 >, < 2,4 >, < 3,4 >, < 5,4 >, < 6,4 >$ and $< 7,4 >$. Neighbours of these six vertices are disjoint. Also there is exactly one, degree 4 vertex: $< 4,4 >$ These vertices together will contribute to $6 \cdot 3 + 2 = 20$ non-Hamiltonian edges. Now let us look into the subgraph induced by vertices of degree 3 only, which do not have any degree 5 or degree 4 neighbour (Fig 3.9). Maximum Independent Subset induced by these vertices is of cardinality = 9. Hence, these vertices, accounts for 9 non-Hamiltonian edges which have not been counted yet. Hence, the total number of non-Hamiltonian edges $= 20 + 9 = 29$, which does not agree with the previous count, 28. Hence a contradiction. So $OTIS(BF(4,4))$ is not Hamiltonian.

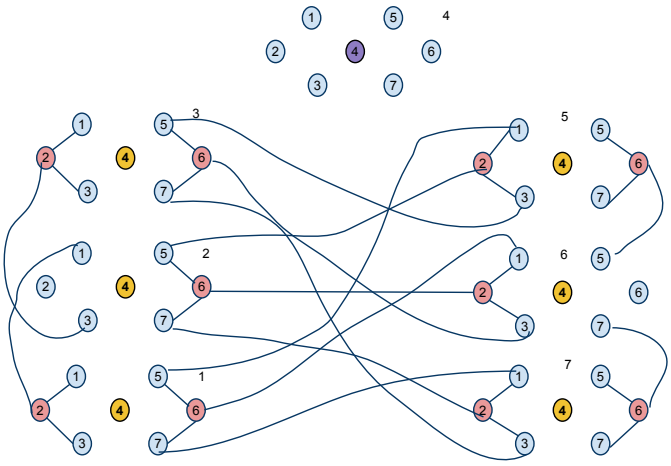

Figure 3.9: $OTIS(BF(4,4))$:Vertices of degree 3 vertices, with no degree 5 and 4 neighbours are shown in red.

However, this argument cannot be used to prove that $OTIS(BF(4,6))$ is Non-Hamiltonian, as $OTIS(BF(4,6))$ has a cycle cover of length 2.

3.5.2 Proof that $OTIS(BF(4,6))$ is not Hamiltonian

We prove our claim in parts. Firstly we identify the forced edges in a Hamiltonian Cycle, assuming that one exists. Then we make a choice of picking one edge as Hamiltonian from an option of two, without any loss of generality. Finally, we arrive at a contradiction.

Claim 3.2 *Vertex* $\langle 4,4\rangle$ *cannot have both* $(\langle 4,1\rangle,\langle 4,4\rangle)$ *and* $(\langle 4,3\rangle,\langle 4,4\rangle)$ *as Hamiltonian edges.*

Proof: If possible let both $\langle 4,1\rangle$ and $\langle 4,3\rangle$ be Hamiltonian neighbours of $\langle 4,4\rangle$.Now, vertex $\langle 4,2\rangle$ has to have either of $(\langle 4,1\rangle,\langle 4,2\rangle)$ and $(\langle 4,2\rangle,\langle 4,3\rangle)$ as its Hamiltonian

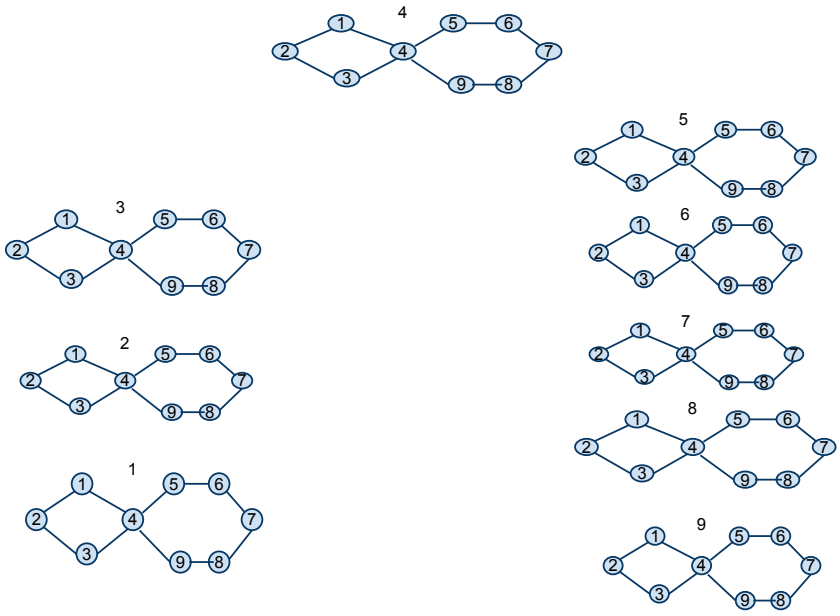

Figure 3.10: $OTIS(BF(4,6))$: The intercluster edges are not shown to maintain clarity

edges. Without loss of generality, let $(\langle 4,2\rangle, \langle 4,3\rangle)$ be the Hamiltonian edge.This forces the following edges,

Cluster 1: $(2,3)$

Cluster 2: $(1,4)$

Intercluster edges: $(\langle 4,1\rangle, \langle 1,4\rangle), (\langle 4,2\rangle, \langle 2,4\rangle), (\langle 1,3\rangle, \langle 3,1\rangle)$

This in turn, forces the following edges:

Cluster 3: $(1,4)$

Intercluster edges: $(\langle 3,2\rangle, \langle 2,3\rangle)$

This clearly forms a forced subcycle. So both $(\langle 4,1\rangle, \langle 4,4\rangle)$ and $(\langle 4,3\rangle, \langle 4,4\rangle)$ cannot be Hamiltonian edges of $\langle 4,4\rangle$. Hence the only possibility is either of $(\langle 4,1\rangle, \langle 4,4\rangle)$ and $(\langle 4,3\rangle, \langle 4,4\rangle)$ is Hamiltonian edge. Without loss of generality (due to the symmetric structure), let $(\langle 4,3\rangle, \langle 4,4\rangle)$ be the Hamiltonian edge.

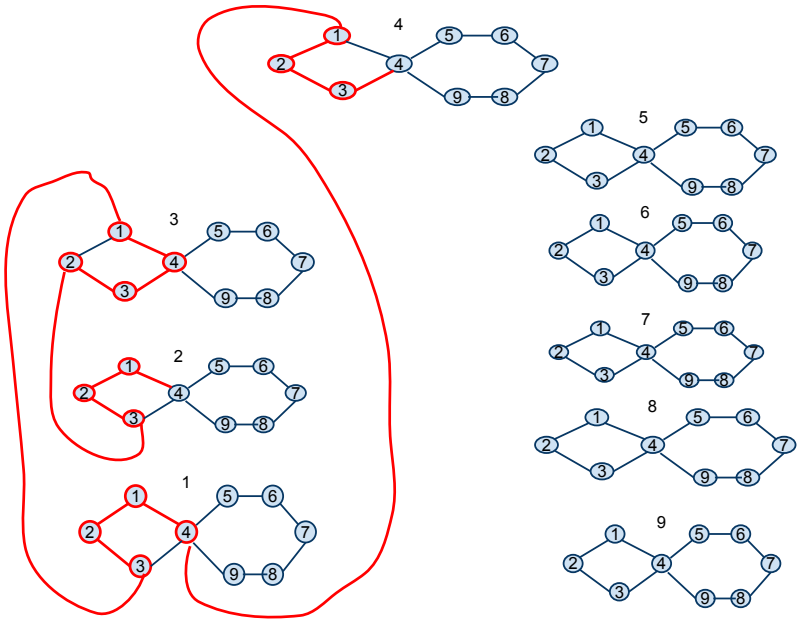

Figure 3.11: $OTIS(BF(4,6))$:The vertices and edges shown in red are saturated vertices and forced edges respectively

Now, in Fig 3.11 the vertices which have already obtained both of its Hamiltonian neighbours, i.e., saturated vertices, are shown in red. We see that $\langle 2,4\rangle$ and $\langle 4,4\rangle$ are the vertices which have obtained exactly one of its Hamiltonian neighbours and rest of the vertices are either completely saturated, or not saturated. So a Hamiltonian cycle exists if and only if, there exists a path between $\langle 2,4\rangle$ and $\langle 4,4\rangle$ spanning all the unsaturated vertices.

Vertex $\langle 4, 4\rangle$ has to have either of $\langle 4, 5\rangle$, $\langle 4, 9\rangle$ as its Hamiltonian neighbour. Without any loss of generality, let us assume $\langle 4, 9\rangle$ is its Hamiltonian neighbour. This again forces a set of edges and makes certain vertices saturated. In the following figure, the forced edges and saturated vertices are marked in red (Fig 3.12).

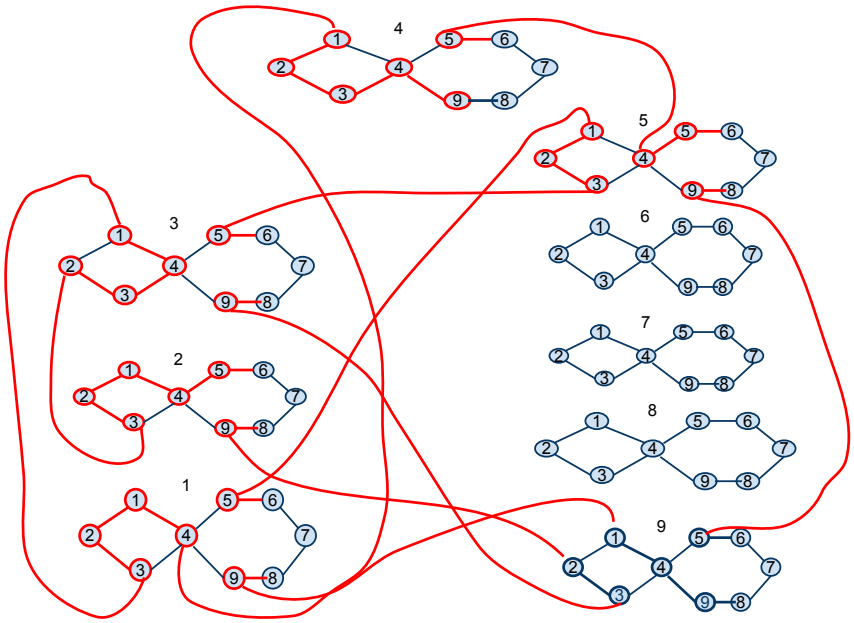

Figure 3.12: $OTIS(BF(4, 6))$:The vertices and edges shown in red are saturated vertices and forced edges respectively

Now, we see in cluster 9, vertex 2, has to have either of $\langle 9, 1\rangle$ or $\langle 9, 3\rangle$ as it Hamiltonian edge. Without loss of generality, let $\langle 9, 3\rangle$ be its Hamiltonian edge. This again forces a set of edges (Fig 3.13).

Now, vertex $\langle 2, 6\rangle$ has to adopt any one of $\langle 2, 7\rangle$, $\langle 6, 2\rangle$ as its Hamiltonian neighbours. Let us study both the cases.

Case 1: Let vertex $\langle 2, 6\rangle$ choose $\langle 6, 2\rangle$ as its Hamiltonian neighbour. Then the following edges become forced:

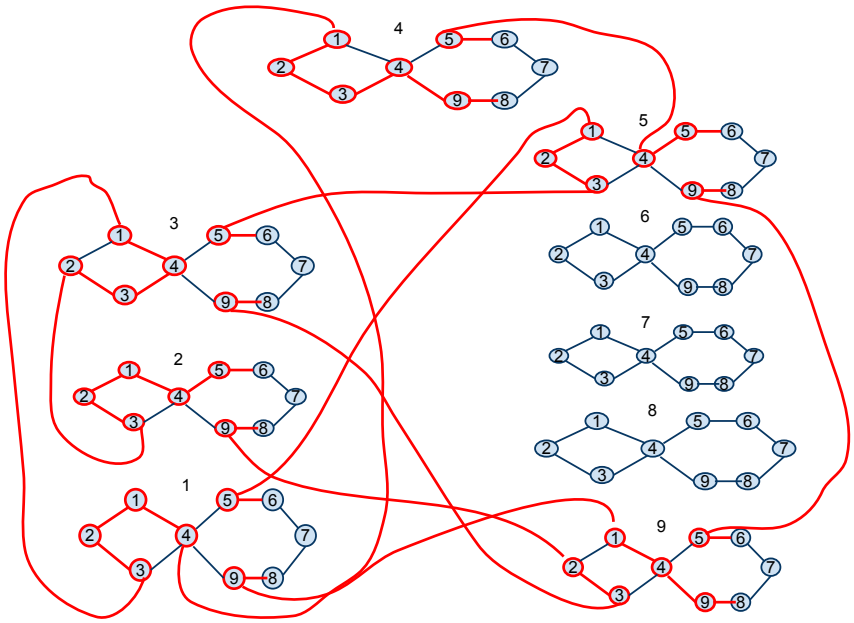

Figure 3.13: ($OTIS(BF(4,6))$:The vertices and edges shown in red are saturated vertices and forced edges respectively

Cluster 2: $(7,8)$

Cluster 8: $(2,1),(2,3)$

Intercluster edges: $(\langle 2,7\rangle,\langle 7,2\rangle)$

Now, In cluster 6, vertex 2, has to have either of $\langle 2,1\rangle$ and $\langle 2,3\rangle$ as its Hamiltonian edge. Without loss of generality, let us take $\langle 2,1\rangle$ as its Hamiltonian edge. This, in turn forces the following edges:

Cluster 6: $(3,4)$.

Intercluster edges: $(\langle 3,6\rangle,\langle 6,3\rangle)$ and hence $(\langle 3,7\rangle,\langle 7,3\rangle)$. This forces the following edges:

Cluster 3: $(7,8)$

Cluster 8: $(3,2),(3,4)$. Hence $(1,4)$ cannot be an edge in this cluster. This forces the following set of edges:

Intercluster edges: $(\langle 8,1\rangle,\langle 1,8\rangle)$, and hence $(\langle 1,7\rangle,\langle 7,1\rangle)$ and the following edges:

Cluster 1: $(7,6)$

Cluster 6: $(1,4),(5,6),(8,9)$

Intercluster edges: $(\langle 5,6\rangle,\langle 6,5\rangle)$ and $(\langle 6,9\rangle,\langle 9,6\rangle)$. This in turn forces the following edges:

Cluster 4: $(6,7)$

Cluster 9: $(7,8)$

Intercluster edges: $(\langle 7,9\rangle,\langle 9,7\rangle)$

Cluster 8: $(4,9)$. Note that, $(9,8)$ is already a forced edge as this edge is incident to a vertex of degree 2. Therefore, this forces the edge $(5,6)$, and the following edge:

Intercluster edge: $(\langle 5,8\rangle,\langle 8,5\rangle)$

Now, since both the vertices $\langle 5,6\rangle$ and $\langle 5,8\rangle$ have become saturated, the edges $(\langle 5,6\rangle,\langle 5,7\rangle)$ and $(\langle 5,8\rangle,\langle 5,7\rangle)$ have to be dropped from the Hamiltonian Cycle assuming one exists. This leaves vertex $\langle 5,7\rangle$ with degree 1 and hence a Hamiltonian Cycle is not possible.

Case 2: Let vertex $\langle 2,6\rangle$ choose $\langle 2,7\rangle$ as its Hamiltonian neighbour. Then the edges

$(\langle 6,2\rangle, \langle 6,1\rangle)$ and $(\langle 6,2\rangle, \langle 6,3\rangle)$ become forced. Now notice that in Cluster 6, both the vertices $\langle 6,1\rangle$ and $\langle 6,3\rangle$ cannot have intercluster edges incident on then as Hamiltonian edges or as non-Hamiltonian edges, as both these cases forces subcycle formation. So exactly one of them has to have the intercluster edge incident on it, as Hamiltonian edge. Without loss of generality, let vertex $\langle 6,1\rangle$ has the edge $(\langle 6,1\rangle, \langle 1,6\rangle)$ as Hamiltonian. Now, this forces the following edges:

Cluster 6: $(3,4)$

Cluster 1: $(7,8)$

Intercluster edges: $(\langle 1,7\rangle, \langle 7,1\rangle)$

Cluster 8: $(1,2), (1,4)$

Intercluster edges: $(\langle 3,7\rangle, \langle 7,3\rangle)$ and $(\langle 3,8\rangle, \langle 8,3\rangle)$ [In Cluster 3, $(7,8)$ cannot be an edge, as this would force subcycle formation in Cluster 8]

Now vertex $\langle 5,7\rangle$ can choose any two of its three incident edges as Hamiltonian edges. e consider all three possible cases and show that Hamiltonian Cycle formation is impossible.

Case 1: $\langle 5,7\rangle$ **chooses** $\langle 5,6\rangle$ **and** $\langle 5,8\rangle$ **as its Hamiltonian neighbours.** In this case, the edges $(5,4)$ and $(5,6)$ becomes forced in Cluster 6, 7 and 8. In clusters 6 and 8, this saturates vertex 4. This, in turn forces the intercluster edges $(\langle 6,9\rangle, \langle 9,6\rangle)$ and $(\langle 8,9\rangle, \langle 9,8\rangle)$. Now, since both the vertices $\langle 9,6\rangle$ and

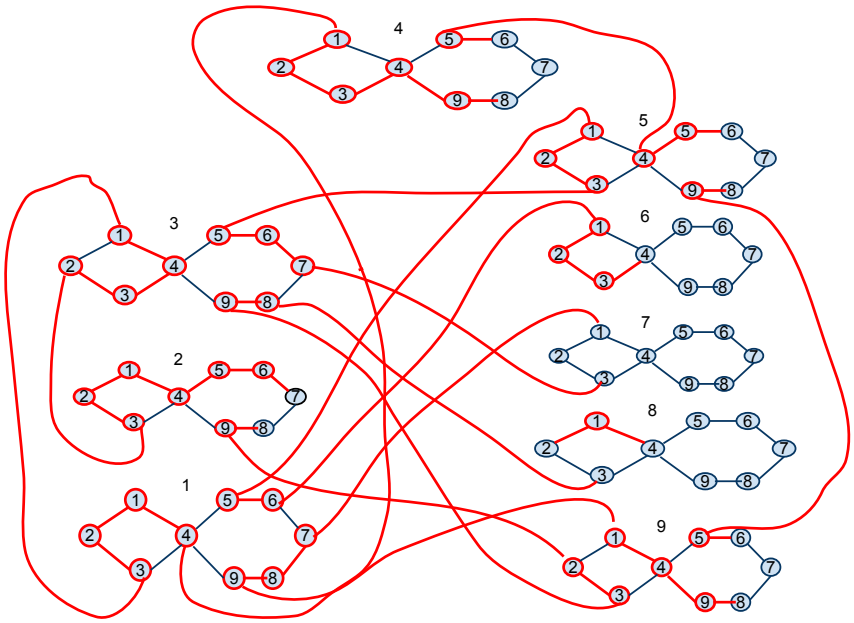

Figure 3.14: $OTIS(BF(4,6))$:The verices and edges shown in red are saturated vertices and forced edges respectively

$\langle 9,8\rangle$ have become saturated, the edges $(\langle 9,6\rangle, \langle 9,7\rangle)$ and $(\langle 9,8\rangle, \langle 9,7\rangle)$ have to be dropped from the Hamiltonian Cycle assuming one exists. This leaves vertex $\langle 9,7\rangle$ with degree 1 and hence a Hamiltonian Cycle is not possible.

Case 2: $\langle 5,7\rangle$ chooses $\langle 5,6\rangle$ and $\langle 7,5\rangle$ as its Hamiltonian neighbours.

In this case, the intercluster edge $(\langle 5,8\rangle, \langle 8,5\rangle)$ become forced. In cluster 6, edges $(4,5)$ and $(5,6)$ gets forced. This saturates vertex $\langle 6,4\rangle$ hence forcing the following edges: $(\langle 4,6\rangle, \langle 4,7\rangle)$, $(\langle 6,9\rangle, \langle 6,8\rangle)$ and $(\langle 6,9\rangle, \langle 9,6\rangle)$. This, in turn, forces edges $(\langle 9,7\rangle, \langle 9,8\rangle)$, $(\langle 7,9\rangle, \langle 9,7\rangle)$ and $(\langle 8,4\rangle, \langle 8,9\rangle)$.Now, vertex $\langle 8,4\rangle$ gets saturated, hence forcing the following edges in cluster 8: $(2,3)$ and $(5,6)$, and the edge $(7,8)$ in cluster 4. Now, note that $(\langle 8,6\rangle, \langle 6,8\rangle)$ cannot be a Hamiltonian edge, as it forces subcycle. Therefore, in cluster 8, $(6,7)$ is

an edge. This in turn forces the edge $(9, 8)$ in cluster 7. Now in cluster 7, $(4, 1)$ and $(4, 3)$ becomes forced. Now, since both the vertices $\langle 7, 1\rangle$ and $\langle 7, 3\rangle$ have become saturated, the edges $(\langle 7, 2\rangle, \langle 7, 1\rangle)$ and $(\langle 7, 2\rangle, \langle 7, 3\rangle)$ have to be dropped from the Hamiltonian Cycle assuming one exists. This leaves vertex $\langle 7, 2\rangle$ with degree 1 and hence a Hamiltonian Cycle is not possible.

Case 3: $\langle 5, 7\rangle$ chooses $\langle 5, 8\rangle$ and $\langle 7, 5\rangle$ as its Hamiltonian neighbours. Following similar argument, as above, we identify the following forced edges (Fig 3.14).

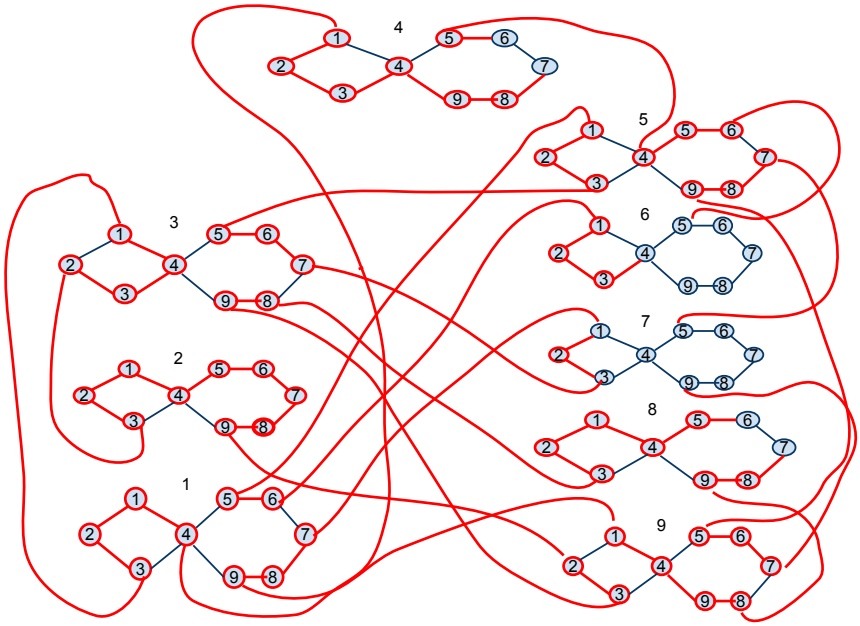

Figure 3.15: $OTIS(BF(4, 6))$:The vertices and edges shown in red are saturated vertices and forced edges respectively

Now, note that vertex $(\langle 7, 4\rangle$ cannot have both $\langle 7, 5\rangle$ and $\langle 7, 9\rangle$ as its Hamiltonian neighbours as this forces subcycle. Without loss of generality, let $\langle 7, 5\rangle$ be its Hamiltonian neighbour, the other Hamiltonian neighbour being $\langle 4, 7\rangle$. This

forces the following edges (Fig 3.15).

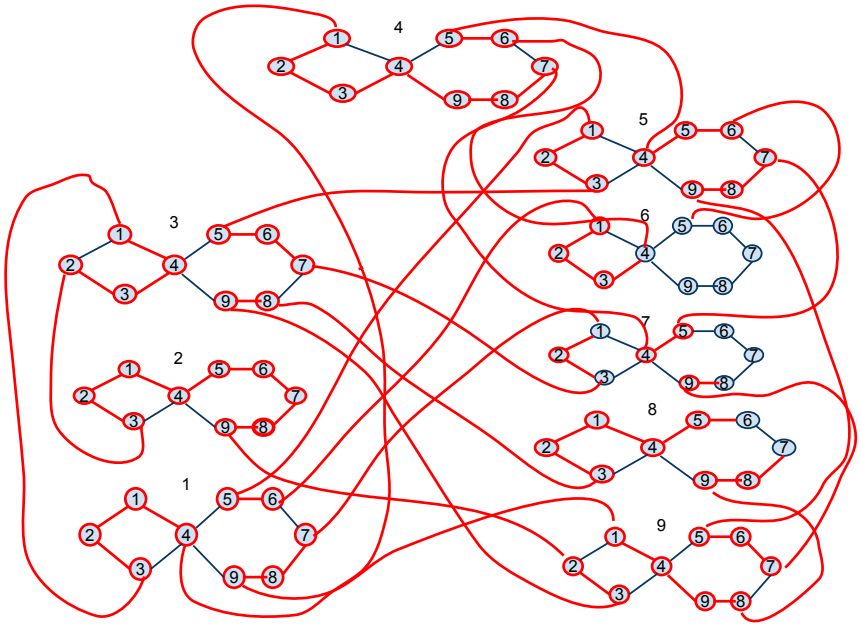

Figure 3.16: $OTIS(BF(4,6))$:The vertices and edges shown in red are saturated vertices and forced edges respectively

Now, since both the vertices $\langle 6,4\rangle$ and $\langle 9,6\rangle$ have become saturated, the edges $(\langle 6,4\rangle,\langle 6,9\rangle)$ and $(\langle 6,9\rangle,\langle 9,6\rangle)$ have to be dropped from the Hamiltonian Cycle assuming one exists. This leaves vertex $\langle 6,9\rangle$ with degree 1 and hence a Hamiltonian Cycle is not possible.

This completes the proof.

3.6 Independent Spanning Trees Construction

Definition 3.1 *[Khuller and Schieber, 1992] Two spanning trees of a graph $G = (V, E)$ are called vertex (respectively edge) independent if they are rooted at the same*

vertex r, and for each vertex $u \in V$, the two paths from u to r, one path in each tree, are internally vertex (respectively edge) disjoint.

We explain with an example in Fig 3.17. Here he two spanning trees T_1 and T_2, rooted at vertex 1, are independent.

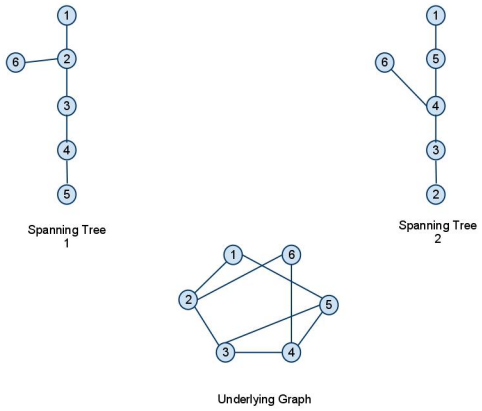

Figure 3.17: T_1 and T_2 are two independent spanning trees of the underlying graph

We know that construction of Hamiltonian Cycle is linear in number of vertics of the OTIS-network on $OTIS(BF(2m+1, 2n+1))$ and $OTIS(BF(2m+1, 2k))$ (From observation 4.1).

Once the Hamiltonian Cycle is constructed, we can construct two rooted Independent Spanning Trees in $O(1)$ time as follows: (Fig 3.18)

- Pick any vertex as root and denote it as r
- Delete one edge incident to r. This gives a spanning Tree T_1
- Now, retain the edge previously deleted, and delete the other edge incident to r. This gives another spanning tree T_2

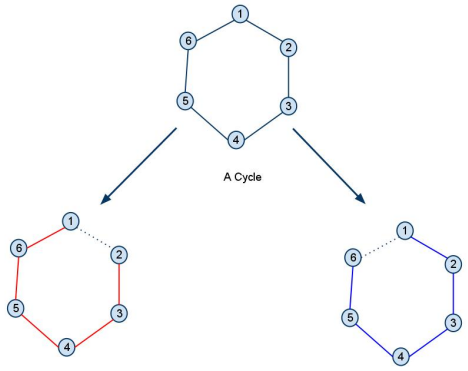

Figure 3.18: Independent spanning tree construction from Hamiltonian cycle

Clearly, for all vertices v in the graph, paths connecting r and v in T_1 and T_2 are edge and vertex disjoint. Even if we make $BF(2m+1, 2n+1)$ or $BF(2m+1, 2k)$ denser by introducing chords inside the cycles C_{2m+1}, C_{2n+1} or C_{2k} such that at least one of the vertices retain degree 2, the two rooted Independent Spanning Trees constructed by this algorithm in will still be valid, which will run in time linear in the number of vertices of the OTIS-graph. But the general algorithm by [Itai and Rodeh, 1988] will show poor performance as it runs in time linear in the number of edges.

CHAPTER 4

Longest Path on Biconvex Graphs

The longest path problem is the problem of finding a simple path of maximum length in a graph. Polynomial solutions for this problem are known only for special classes of graphs, while it is NP-hard on general graphs. We are proposing a $O(n^6)$ time algorithm to find the longest path on biconvex graphs, where n is the number of vertices of the input graph. We have used Dynamic Programming approach. We also present a more efficient absolute approximation algorithm with Constant $C = 1$ to find longest path on biconvex graphs.

4.1 Biconvex Graphs

A bipartite graph G=(S, T, E) is convex on the vertex set S if S can be ordered so that for each element t in the vertex set T the elements of S connected to t form an interval of S; this property is called the adjacency property. G is biconvex if it is convex on both S and T [Abbas and Stewart, 2000*a*].

4.1.1 Related Results

The class of Biconvex Graphs was first defined by Glover in 1967 [Glover, 1967]. Glover showed a practically important application of doubly convex-bipartite graphs in industry. Lipski and Preparata [Lipski and Preparata, 1981] solved the maximum matching problem on both doubly convex-bipartite graphs and convex-bipartite graphs. Dekel and Sahni [Dekel and Sahni, 1984] developed an efficient parallel algorithm that finds maximum matchings in convex-bipartite graphs. Their result may also be used to solve several scheduling problems. Abbas and Stewart showed

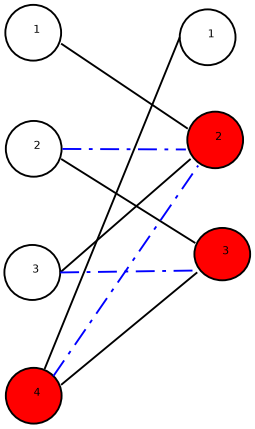

Figure 4.1: he vertices which violate adjacency property, i.e., whose neighborhood does not form an interval in the other partition, are shown in dark. The dotted edges are not present in the graph.

that the vertices of biconvex graphs have an ordering that they called a biconvex straight ordering. This ordering preserves the biconvex property, and it generalizes the strong ordering achievable for bipartite permutation graphs and such an ordering can be achieved efficiently in parallel [Abbas and Stewart, 2000*b*]. They used this ordering to solve the vertex ranking problem on biconvex graphs.

Longest Path Problem is one of the most well known NP-hard problems in Graph Theoretic literature. Only approximate solutions is known to find paths of logarithmic length on Hamiltonian graphs [Karger *et al.*, 1997]. In fact, Karger et al. show that the problem of finding a path of length $n-n^{\epsilon}$ on a n-vertex Hamiltonian graph is NP-Hard and there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP. Hence it is meaningful to investigate the Longest Path problem on special graph classes, for which Hamiltonian Path problem is polynomial time solvable. But surprisingly, there are only a very few known polynomial time solutions for the Longest Path problems. The Hamiltonian Path problem remains NP-complete even when restricted to some small classes of graphs such as split graphs, chordal bipartite graphs, strongly chordal graphs, circle graphs, planar graphs, and grid graphs [Golumbic, 2004] [Ioannidou *et al.*, 2009]. However it becomes polynomial time solvable on some special graph classes, namely, inter-

val graphs, co-comparability graphs, circular-arc graphs and convex bipartite graphs [Ioannidou *et al.*, 2009]. But surprisingly, there are a very few known polynomial time solutions for the Longest Path problems. The small graph classes on which Longest path problem has a polynomial time solution includes weighted trees, block graphs, cacti [Uehara and Uno, 2004], bipartite permutation graphs [Uehara and Valiente, 2007] and Ptolemaic graphs. Recently polynomial time solutions have been proposed by Ioannidou et al. on interval graphs [Ioannidou *et al.*, 2009] and co-comparability graphs [Ioannidou and Nikolopoulos, 2010]. However, status of the Longest path problem was left open for convex and biconvex graphs in [Ioannidou *et al.*, 2009].

4.1.2 Organization

We will discuss our results under the following sections. In section 4.2, we give some preliminaries, in section 4.3 we discuss about the biconvex ordering and monotonicity of a path. This ordering in the most crucial part, exploiting which we design the algorithm. In section 4.4 we discuss about the algorithm and argue its correctness. In section 4.5 we analyse the time complexity of the algorithm. In section 4.6 we discuss about the time-efficient additive approximation algorithm.

4.2 Preliminaries

We will use standard graph theoretic terminology. Let $G = (V, E)$ be a finite undirected simple graph with vertex set $V(G)$ and edge set $E(G)$. A simple path of a graph G is a sequence of distinct vertices $v_1, v_2, \cdots, v_k$ such that $v_i v_{i+1} \in E(G)$ $\forall 1 \leq i \leq k-1$ and is denoted by $(v_1, v_2, \cdots, v_k)$. We define the length of a simple path P to be the number of edges in P and $V(P)$ to be the set of vertices in the path P. v_k is referred to as the right endpoint of the path $P = (v_1, v_2, \cdots, v_k)$. Moreover, let $P = (v_1, v_2, \cdots, v_{i-1}, v_i, v_{i+1}, \cdots, v_j, v_{j+1}, \cdots, v_k)$ and $P_0 = (v_i, v_{i+1}, \cdots, v_j)$ be two paths of a graph. Sometimes, we shall denote the path

P by $P = (v_1, v_2, \cdots, v_{i-1}, P_0, v_{j+1}, \cdots, v_k)$. We define the sub path of P to be a path P' such that $V(P') \subseteq V(P)$ and $E(P')$ is the same as $E(P)$.
Let π be some ordering given on the indices of the vertices. We use the notation $v_i \prec_\pi v_j$ to denote that index i occurs before index j in the given ordering π.

4.3 Biconvex Orderings and Monotone Paths

We introduce a new ordering on Biconvex graphs. In this section, we first define two orderings σ_S and σ_T on the indices of the vertices of a biconvex graph and then define *Monotonicity* of a path. We assume that π_1 and π_2 are the labellings of the vertices of S and T partition respectively (in increasing order of indices), preserving the adjacency property. Using these two orderings, π_1 and π_2, we create σ_S and σ_T as follows.

4.3.1 Ordering of Vertices

Let $\pi_1 = (s_1, s_2, ..., s_p)$ be the vertices of partition S, and similarly, $\pi_2 = (t_1, t_2, ..., t_q)$ be the vertices of partition T, where $|S| = p$ and $|T| = q$ and total number of vertices $n = p + q$. We introduce an ordering σ_S as follows:
Initially set $\sigma_S = \pi_1$.

1. Traverse the vertices of π_2 from left to right.
2. Find the leftmost endpoint and rightmost endpoint of the neighborhood interval of every t_i. Let s_{li} and s_{ri} be the leftmost and rightmost endpoints of the neighborhood interval of some t_i respectively.
3. Update σ_S as follows: Insert t_i immediately after its rightmost neighbor, i.e after s_{ri} $\forall 1 \leq i \leq q$.

Hereafter we will denote by $\sigma_S = (u_1, u_2, ..., u_n)$ the above ordering of the vertices of graph G.

We denote by $u_{f(u_i)}$ and $u_{h(u_i)}$ the leftmost and rightmost neighbor of u_i in σ_S respectively, which appear before u_i in σ_S.

In a similar fashion, we can start from the ordering π_2 of T and relabel S and obtain another ordering denoted by σ_T. (Fig 4.2)

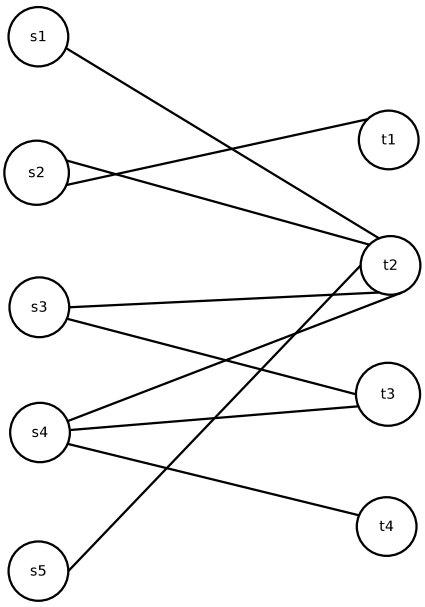

Figure 4.2: Given $\pi_1 = (s_1, s_2, s_3, s_4, s_5)$ and $\pi_2 = (t_1, t_2, t_3, t_4)$
The ordering $\sigma_S = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$

4.3.2 Monotonic Path

We define *S-Monotone* and *S-S* path in the section. *T-Monotone* and *T-T* paths fare defined symmetrically.

Definition 4.1 *A S-Monotone path of a Biconvex graph $G = (S \cup T, E)$ is a simple path $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, \ldots, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ such that $s_{\alpha_k} \prec_{\sigma_S} s_{\alpha_{k+1}} \forall\ k \ni 1 \leq k \leq j$.*

Symmetrically, we define T-Monotone path.

Definition 4.2 *A path which starts from a vertex on S-partition and end on a vertex in S-partition is called a S-S path.*

Symmetrically, we define T-T path. We state and prove the lemma for S-S paths. The same argument will hold good for T-T paths just by replacing the S-vertices with T-vertices and vice versa.
First, we informally introduce the lemma. Informally, we claim that given any simple path $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2} \ldots, s_{\alpha_{j-1}}, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ of a biconvex graph, for the longest S-S sub path of P we can construct an equivalent path P' such that, $P' = \{s_{\gamma_1}, t_*, s_{\gamma_2}, t_*, \cdots, s_{\gamma_j}\}$ where $\gamma_1 \prec_{\sigma_S} \gamma_2 \prec_{\sigma_S} \gamma_3 \ldots \prec_{\sigma_S} \gamma_j$, where $\gamma_i \in \{\alpha_1, \alpha_2, \ldots, \alpha_j\}$ and t_* denotes T-vertex of some index belonging to $\{\beta_1, \beta_2, \ldots, \beta_j\}$.

Lemma 4.1 *Let $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2} \ldots s_{\alpha_{j-1}}, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ be a simple path of a biconvex graph $G = (S, T, E)$. Let P_{max} denote the longest S-S sub path of P. Then the vertices in $V(P_{max})$ can be reordered to get a path $P_{max}{}'$, which is S-Monotone.*

Proof: We prove the lemma by induction on $|S|$.
Basis: To prove the basis, let P=$\{s_\alpha, t_a, s_\beta, t_b, s_\gamma\}$ be a simple path which violates the monotonicity property. We construct P' on the set of vertices of $V(P)$ such that P' satisfies S-Monotonicity property. There are three possible cases.

- $\alpha \succ \beta \succ \gamma$. Then P'=P^R.

- $\alpha \succ \beta$, $\beta \prec \gamma$ and $\alpha \prec \gamma$. It follows that $\beta \prec \alpha \prec \gamma$. Now t_b is adjacent to s_β and s_γ. Owing to the adjacency property, t_b is also adjacent to s_α. Hence we have P'=$\{s_\beta, t_a, s_\alpha, t_b, s_\gamma\}$. The other case $\beta \succ \alpha \succ \gamma$ can be dealt with similarity to (1). (Fig 4.3) [1]

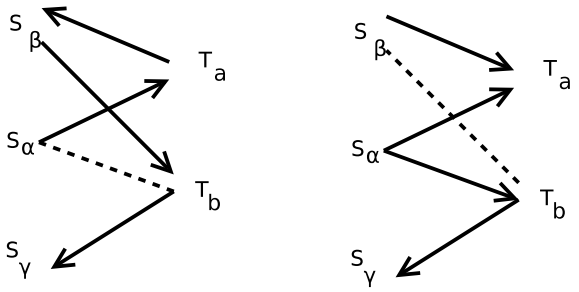

Figure 4.3: (a) S-S path violating monotonicity property (b) S-Monotone path

- $\alpha \succ \beta$, $\beta \prec \gamma$ and $\alpha \succ \gamma$. It follows that $\beta \prec \gamma \prec \alpha$. Now t_a is adjacent to s_α and s_β. Owing to the adjacency property, s_a is also adjacent to s_γ. Hence we have P'=$\{s_\beta, t_b, s_\gamma, t_a, s_\alpha\}$. The other case $\beta \succ \gamma \succ \alpha$ can be dealt in a similar way. Thus the lemma works when the path contains at least three S vertices. (Fig 4.4)

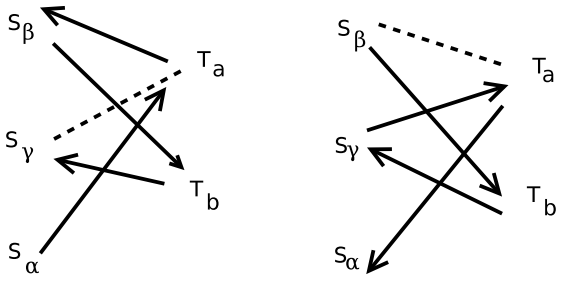

Figure 4.4: (a) S-S path violating monotonicity property (b) S-Monotone path

Inductive hypothesis: For the longest S-S sub path of a simple path P=$\{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, \ldots, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$, there exists a path $P' = \{s_{\gamma_1}, t_*, s_{\gamma_2}, \ldots t_*, s_{\gamma_j}\}$ satisfying the

[1] Graphs considered here are undirected. In the diagram, the arrowheads simply represent the vertex-traversal sequence in the path. The dotted lines represent the edges that have to present owing to adjacency property.

S-Monotonicity property.

Inductive Step: Let P_1=$P \cdot s_{\alpha_{j+1}}$.

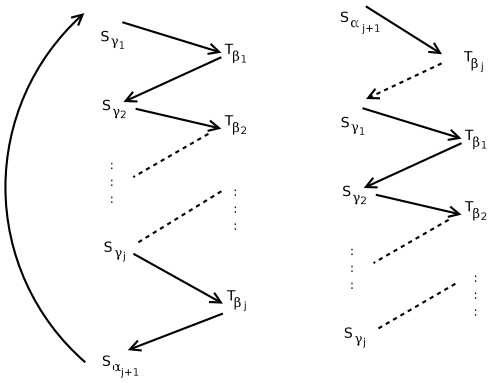

Figure 4.5: (a) Non S-Monotone path P_1=$P \cdot S_{\alpha_{j+1}}$ (b) S-Monotone path P

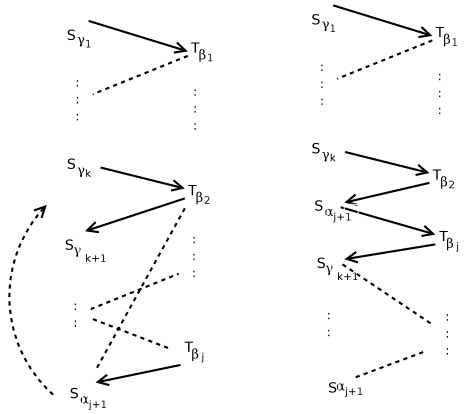

Figure 4.6: (a)Non S-Monotone path P (b) S-Monotone path P_1=$P \cdot S_{\alpha_{j+1}}$

- If $\alpha_{j+1} \succ \gamma_j$, then it follows that the longest S-S sub path of P_1 is already S-Monotone. No further reordering is required.
- If $\alpha_{j+1} \prec \gamma_1$. This implies that the last T vertex t_{β_j} is also adjacent to s_{γ_1}. Then it follows that
  P_1'=$s_{\alpha_{j+1}}, t_{\beta_j} \cdot \{s_{\gamma_1}, t_{\beta_1}, \cdots, s_{\gamma_j}\}$. (Fig 4.5)

- If there exist some $k \ni \gamma_k \prec \alpha_{j+1} \prec \gamma_{k+1}$, then in the path $P = \{s_{\gamma_1}, t_{\beta_1}, \ldots s_{\gamma_k}, t_{\beta_k}, s_{\gamma_{k+1}}, \ldots s_{\gamma_j} t_{\beta_j}\}$. We see that t_{β_k} is adjacent to $s_{\alpha_{j+1}}$ -(1). Now, $\alpha_{j+1} \prec \gamma_{k+1}$. Also from Path P, we have $\gamma_{k+1} \prec \gamma_j$. Hence t_{β_j} is also adjacent to $s_{\gamma_{k+1}}$-(2). From (1) and (2) we have P_1'=$\{s_{\gamma_1}, t_{\beta_1}, \ldots, s_{\gamma_k}, t_{\beta_k}, s_{\alpha_{j+1}}, t_{\beta_j}, s_{\gamma_{k+1}}, \ldots, s_{\gamma_j}\}$. (Fig 4.6)

4.4 The Algorithm and Correctness

4.4.1 Some constructs and notations used in the algorithm

Definition 4.3 *For every pair of indices i, j such that $1 \leq i \leq j \leq n$ we define the graph $G(i, j)$ to be the subgraph $G[A]$ of G induced by the set $A = \{u_i, u_{i+1}, ..., u_j\} \backslash \{u_k \in T(G) : u_{f(u_k)} <_{\sigma_S} u_i\}$.*

Definition 4.4 *Let P be a path of $G(i, j)$, $1 \leq i \leq j \leq n$.The path P is called $S-$ bimonotone if P is a $S-$ Monotone path of $G(i, j)$ and both endpoints of P belong to $S-$ partition.*

Similarly we define $T-$ *bimonotone* path symmetrically.

Notation

Let $\sigma_S = (u_1, u_2, ..., u_n)$ be the ordering on G [or $\sigma_T = (u_1, u_2, ..., u_n)$]. $\forall u_k \in S(G)$ [or $\forall u_k \in T(G)$] we denote by $P(u_k; i, j)$ the longest S-bimonotone [or T-bimonotone] path of $G(i, j)$ with u_k as its right endpoint and by $l(u_k; i, j)$ the number of vertices of $P(u_k; i, j)$.

4.4.2 Algorithm

In this section, we present our algorithm for solving the longest path problem on biconvex graphs; it consists of three phases. Let S denote the partition with higher

cardinality, i.e, $|S| \geq |T|$. Therefore the length of the longest path is less than or equal to $2|T|$.

- **Phase1:**

  Given the biconvex graph $G = (S, T, E)$, generate the ordering $\sigma_S = (u_1, u_2, \cdots, u_n)$. Now, for all s_i, s_j, where, $1 \leq i < j \leq |S|$, do the following:

  1. Choose the subsequence $\sigma_{sij} = (u_k, u_{k+1}, \cdots, u_m)$ such that u_k is the vertex s_i and u_m is either s_j or, $u_m \in T(G)$ and it lies between s_j and s_{j+1} in the ordering σ_S. If there are multiple T vertices lying between s_j and s_{j+1} in σ_S, then u_m is the rightmost of them. (Fig 4.7)
  2. Run the "Longest path" routine for all σ_{sij} as the input ordering. There i and j will replace indices 1 and n respectively.
  3. Remember the maximum path length obtained over these iterations and all the paths of that maximum length.

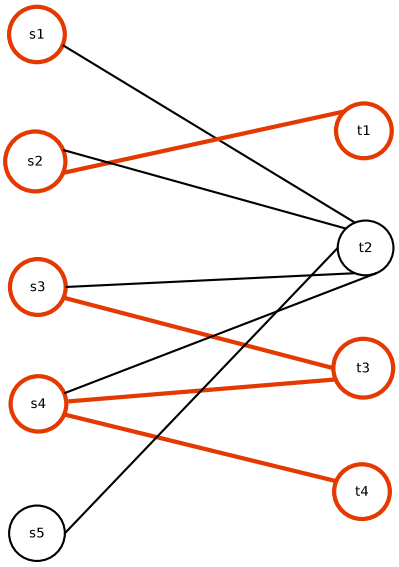

Figure 4.7: $\sigma_S = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$ and $\sigma_{S14} = (s_1, s_2, t_1, s_3, s_4, t_3, t_4)$
$\sigma_{S13} = (s_1, s_2, t_1, s_3)$

- **Phase 2:**

  Symmetric to *Phase 1*, this phase is executed for vertices of T-partition with the initial ordering $\sigma_T = (u_1, u_2, \cdots, u_n)$

- **Phase 3:**

  1. Let the path lengths obtained as output from *Phase 1* and *Phase 2* be x and z respectively. Compute $\max\{x, z\}$. Without loss of generality, let z be the maximum.
  2. Consider all the T-bimonotone paths of length z obtained from Phase2. check if the end vertices of the paths have any unvisited neighbor, i.e., neighbor which does not occur on that path.
  3. if such a neighbor exists, extend the path till that neighbor. Let P' denote this extended path.
  4. Output $z + 1$ as the maximum path length and P' as the longest path.
  5. Else, output z as the maximum path length and the corresponding path as the longest path.

Algorithm 1 Longest Path (*Phase 1*)

Input: The biconvex graph G and input ordering $\sigma_{s1n} = (u_1, u_2, ..., u_n)$.
Output: A longest S-bimonotone path of G and the longest path length.

for $j = 1$ to n **do**
  for $i = j$ down to 1 **do**
    if $i = j$ and $u_i \in S(G)$ **then**
      $l(u_i; i, i) \leftarrow 1$;$P(u_i; i, i) \leftarrow (u_i)$
    end if
    if $i \neq j$ **then**
      for all $u_k \in S(G)$, $i \leq k \leq j-1$ **do**
        $l(u_k; i, j) \leftarrow l(u_k; i, j-1)$;
        $P(u_k; i, j) \leftarrow P(u_k; i, j-1)$;
      end for
      if $u_j \in S(G)$ **then**
        $l(u_j; i, j) \leftarrow 1$;$P(u_j; i, j) \leftarrow (u_j)$
      end if
      if u_j is a T vertex i.e $u_j \in T(G)$ and $i \leq f(u_j)$ **then**
        execute process *G(i,j)*
      end if
    end if
  end for
end for
compute the $\max\{(u_k; 1; n) : u_k \in S(G)\}$ and the corresponding path $P(u_k; 1; n)$.Return (max $\{(u_k; 1; n) : u_k \in S(G)\}$ -1) as the maximum path length and $P(u_k; 1; n)$ as a longest path.
We carry out the second phase by re-running the algorithm with $\sigma_{t1n} = (u_1, u_2, ..., u_n)$. By replacing vertex set T with S and vice versa. The output of second phase is a longest T-bimonotone path of G and the longest path length.

Algorithm 2 The subroutine **process(G(i,j))**

for $y = f(u_j) + 1$ to $j - 1$ **do**
  for $x = f(u_j)$ to $y - 1$ **do**
    if $u_x, u_y \in S(G)$ **then**
      $w_1 \leftarrow l(u_x; i, j-1)$; $P'_1 \leftarrow P(u_x; i, j-1)$
      $w_2 \leftarrow l(u_y; x+1, j-1)$; $P'_2 \leftarrow P(u_y; x+1, j-1)$
      if $w1 + w2 + 1 > l(u_y; i, j)$ **then**
        $l(u_y; i, j) \leftarrow w1 + w2 + 1$;
        $P(u_y; i, j) = (P'_1, u_j, P'_2)$;
      end if
    end if
  end for
end for
return the value $\{l(u_k; i, j)\}$ and the path $\{P(u_k; i, j), \forall u_k \in S(G(f(u_j)+1, j-1))\}$

4.4.3 Illustration with Example

Let us consider the biconvex graph shown in Fig 4.8. We illustrate the first phase of the algorithm with σ_{S1n} in Fig 4.9 and 4.10.

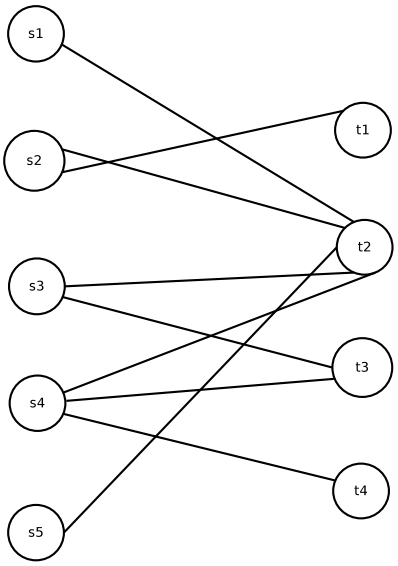

Figure 4.8: σ_{S1n}
$=(s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$

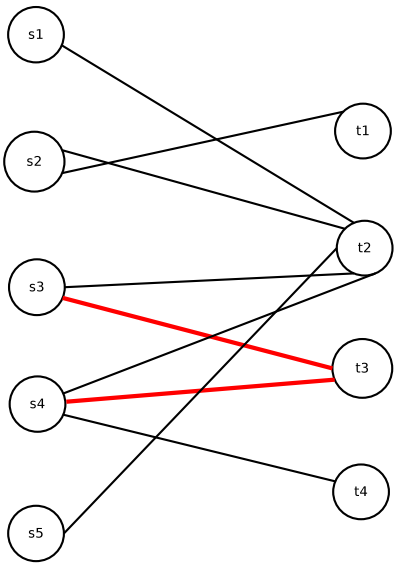

Figure 4.9: Path getting updated at call $process(G(4,6))$ and $x = 4, y = 5$

4.4.4 Proof of Correctness

Candidates for Longest Path

There are four candidates for a longest path P, which starts in a vertex of S or T and ends in a vertex of S or T. We denote these candidates as $P^{SS}, P^{ST}, P^{TS}, P^{TT}$, where P^{XY} denotes a longest path among the set of paths starting from a vertex in X and ending in a vertex in Y. Now the outline of our algorithm is as follows:

1. Compute the length of P^{SS} and P^{TT} from *Phase 1* and *Phase 2* of the algorithm respectively.

2. Let $x = |P^{SS}|$ and $z = |P^{TT}|$ and let $y = \max(x, z)$.

3. As is evident from the *Phase 3*, the length of a longest path possible on the graph is either y (If P^{SS} or P^{TT} is the candidate) OR $y + 1$ (If P^{ST} or P^{TS} is the candidate).

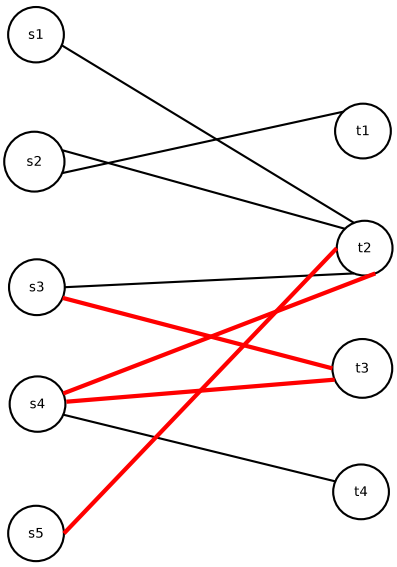

Figure 4.10: Path getting updated at call $process(G(1,9))$ with $x = 5, y = 8$

4.4.5 Correctness Argument

We shall prove two claims in order to prove the correctness.

Claim 4.1 *The Algorithm (Phase 1) correctly computes a longest S-bimonotone path(i.e, P^{SS} path) of the graph G and Phase 2 correctly computes a longest T-bimonotone path(i.e, P^{TT} path) of the graph G.*

The following observations hold for every induced subgraph $G(i,j), 1 \leq i \leq j \leq n$ and is used for proving the correctness of *Phase 1* of the algorithm. Similarly we can state the observations by replacing σ_S with σ_T, partition S with T and vice versa and prove the correctness of *Phase 2* of the algorithm. he following properties hold trivially:

Observation 4.4.1 *Let G(i, j) be the induced subgraph of S-bimonotone graph G and σ_S be the input ordering of G. Let $P = (P_1, u_j, P_2)$ be a S-bimonotone path of G(i, j), and let u_j be a S vertex of G(i, j). Then, P_1 and P_2 are S-bimonotone paths of G(i, j).*

Observation 4.4.2 *Let G(i, j) be the induced subgraph of biconvex graph G and* σ_S *be the input ordering of G. Let* P_1 *be a S-bimonotone path of G(i, j) with* u_x *as its right endpoint, and let* P_2 *be a S-bimonotone path of G(x + 1;j-1) with* u_y *as its right endpoint, such that V(*P_1*)∩V(*P_2*) =∅. Suppose that* u_j *is a T vertex of G and that* $u_i \prec_{\sigma_S} u_{f(uj)} \prec_{\sigma_S} u_x$ *Then,* $P = (P_1, u_j, P_2)$ *is a S-bimonotone path of G(i, j) with* u_y *as its right endpoint.*

Now we shall prove our claim.

Proof: Let P be the longest S-bimonotone path of the subgraph $G(i, j)$ with $u_y \in S(G(i, j))$ as its rightmost endpoint. Consider the case when $T(G(i, j)) = \phi$. Hence the algorithm sets path $P(u_y; i, j) = u_y$ and $l(u_y; i, j) = 1$. Therefore, the lemma holds for every induced subgraph $G(i, j)$ for which $T(G(i, j)) = \phi$.

Now consider the case where $T(G(i, j)) \neq \phi$.

We will prove the correctness by induction on number of T-vertices.

Basis: Consider $\sigma_S = (u_1, u_2, u_3)$ where $u_1, u_2 \in S(G(i, j))$ and $u_3 \in (G(i, j)$ and u_1 and u_2 are the leftmost and rightmost neighbors of u_3 respectively. $process(G(i, j))$ will be called and according to the algorithm, $P(u_2; i, j) = (u_1, u_3, u_2)$ and $S(u_2; i, j) = 3$ will be given as output which is indeed a longest S-bimonotone path.

Inductive Hypothesis: Let the algorithm generate the longest S-bimonotone path when $G(i, j)$ contain k number of T-vertices.

Induction Step: Now we consider the scenario where we $G(i, j)$ contains $k + 1$ T-vertices. Now according to the algorithm, all the neighbors of the newly added T-vertex are considered and for all possible combinations of the neighbors say u_x and u_y the length of path $P(u_y; i, j)$ is computed by considering the sub paths P_1, a longest S-bimonotone path in $G(i, j - 1)$ with u_x as the rightmost vertex and P_2, a longest S-bimonotone path in $G(x + 1, j - 1)$ with u_y as the rightmost vertex. (By the principle of strong induction, P_1 and P_2 are longest S-Bimonotone paths, given as output by the algorithm as in both the subgraphs $G(i, j - 1)$ and $G(x + 1, j - 1)$

the number of T-vertices $\leq k$.)
Now P is updated as follows: $P(u_y; i, j) = (P_1, uj, P_2)$ if $l(P_1) + l(P_2) + 1 > l(P)$, where $V(P_1) \cap V(P_2) \neq \emptyset$. This is evident from observation 2.

Hence the path computed by *Phase 1* of the Algorithm is indeed a longest $S-$ *bimonotone* path on G with u_y as its right endpoint.

Claim 4.2 *At least one of the paths, obtained as output from Phase 1 and Phase 2 is extendible if and only if a longer path exists.*

Proof: *Forward Implication:*
This follows from the correctness of the Algorithm (*Phase 1*),proved above.
Backward Implication
Here we need to prove the following implication: A longer path exists $\implies$ It is the extension of one of the paths obtained in *Phase 1* and *Phase 2*.
As we have already discussed, there are 4 candidates for longest path, P^{SS}, P^{TT},P^{ST} and P^{TS}. In *Phase 1* and *Phase 2* of the algorithm, all the longest S-bimonotone and T-bimonotone paths (respectively) are generated.

So the only case, where a longer path might exist, is the case where the candidate longest path is either a ST or a TS path.

Without Loss of Generality, let the longest path be a ST path. Now, from lemma 1, we know, for the longest S-S sub path of P, we can reorder the vertices, so that the S-vertices follow the S-Monotonicity property. Let length of this ST path be x. Then the length of its longest S-S sub path has to be $x - 1$.

Now, let the longest path length, given as output of *Phase 1* (Generating longest S-bimonotone path) be m.

If possible, let $m \neq x - 1$. Then there are two possibilities:

- If $m < x-1$, this contradicts the correctness of Algorithm (*Phase 1*). Therefore, $m \not< y$.
- If $m > x - 1$, this implies ST is not the longest path. Hence a contradiction again.

Hence $m = x - 1$. Since the *Phase 1* of the algorithm has generated all possible S-bimonotone paths of length m, hence the longest ST path has to be an extension of any one of them.

Similar argument will follow if the longest path is a TS path.

4.5 Time Complexity

- The ordering σ_S (and similarly σ_T) will take $O(|E|)$ time, where $|E|$ denotes the number of edges in the graph.
- The subroutine process() takes O($|S|^2$) due to $|S|^2$ pairs of neighbors u_x and u_y of the T vertex in the graph $G(i, j)$.
  Additionally, the subroutine process() is executed at most once for each subgraph G(i, j) of G, $1 \leq i \leq j \leq n$, i.e, it is executed O(n^2) times. Thus, time complexity is $O(|S|^2|n|^2)$ time for *Phase 1* of the algorithm.Similarly for *Phase 2* of the algorithm, time complexity is $O(|T|^2|n|^2)$. Hence total complexity is $\max\{O(|S|^2|n|^2), O(|T|^2|n|^2)\}$, which is $O(n^4)$ where n is the total number of vertices of the biconvex graph.
- Generating σ_{sij} [σ_{tij}]from σ_S [and similarly σ_T] will take linear time.
- *Phase 1* of the algorithm will be executed for each ordered pair s_i, s_j. There can be $O(|S|^2)$ such ordered pairs. Similarly, *Phase 2* can be executed for $O(|T|^2)$ such ordered pairs. So the total time complexity is
  $\max\{O(|S|^2|n|^4), O(|T|^2|n|^4)\}$, which is $O(n^6)$ where n is the total number of vertices of the biconvex graph.

4.6 An $O(n^4)$-time Approximation Algorithm with Constant Additive Error

A simple observation from our algorithm leads to an absolute approximation to find longest paths in biconvex graphs in time $O(n^4)$. Since the running time reduces substantially, (by a factor of n^2), and the quality of approximation is considerably good, this modified algorithm can be practically more useful. We discuss this modified algorithm in this section.

The Algorithm:

1. Given the biconvex graph $G = (S, T, E)$, generate the orderings σ_S and σ_T.
2. Run the Longest Path routine with σ_S and remember the longest path length and the corresponding path; let us denote this length as max_S.
3. Run the Longest Path routine with σ_T and remember the longest path length and the corresponding path; let us denote this length as max_T.
4. Declare $\max\{max_S, max_T\}$ as the longest path length and the corresponding path as the longest path.

Claim 4.3 *Steps 2 and 3 correctly generate longest S-Bimonotone and T-Bimonotone paths respectively.*

Proof: The proof of this lemma trivially follows from Claim 4.1.

Claim 4.4 *Let L_{max} be the longest path length declared by the algorithm and L'_{max} be the correct longest path length on the given graph. Then either $L_{max} = L'_{max}$ or $L_{max} = L'_{max} - 1$*

Proof: As we have already discussed, there are 4 candidates for longest path, P^{SS}, P^{TT},P^{ST} and P^{TS}. In Step 2 and Step 3 of the algorithm, all the longest S-bimonotone and T-bimonotone paths (respectively) are generated.
So the only case, where a longer path might exist, is the case where the candidate longest path is either a ST or a TS path. Now from the backward Implication proof of Claim 4.2, it is follows, that if a ST or a TS path is the longest path in the graph, then its length is exactly one greater than the length of the longest S-Bimonotone or T-Bimonotone paths. Hence, in that case, the path length given as output from the modified algorithm, is exactly one less than the actual longest path length.

4.6.1 Time Complexity

- The ordering σ_S (and similarly σ_T) will take $O(|E|)$ time, where $|E|$ denotes the number of edges in the graph.
- The subroutine process() takes O($|S|^2$) due to $|S|^2$ pairs of neighbors u_x and u_y of the T vertex in the graph $G(i, j)$.
  Additionally, the subroutine process() is executed at most once for each subgraph G(i, j) of G, $1 \leq i \leq j \leq n$, i.e, it is executed O(n^2) times. Thus, time complexity is $O(|S|^2|n|^2)$ time for *Phase 1* of the algorithm.Similarly for *Phase 2* of the algorithm, time complexity is $O(|T|^2|n|^2)$. Hence total complexity is $\max\{O(|S|^2|n|^2), O(|T|^2|n|^2)\}$, which is $O(n^4)$ where n is the total number of vertices of the biconvex graph.

CHAPTER 5

Conclusions and Future Work

We have considered two well known problems on important special classes of graphs in the thesis and have answered a few important open questions. We have studied the Hamiltonian Cycle problem on Optical Transpose Interconnection Networks and have shown that existence of Hamiltonian cycle on the base graph is not a sufficient condition for the OTIS network to be Hamiltonian. We have shown two infinite families of graph where, the OTIS network is Hamiltonian, but the base graph is not. Further, we have relaxed the condition of Hamiltonian cycle to Hamiltonian path and investigated if Hamiltonian path on the base graph is a sufficient condition for the OTIS network to be Hamiltonian. We have answered the is negative. We have also proposed an alternate algorithm for Independent Spanning Tree construction in time linear in the number of vertices of the input graph.

It would be interesting to come up with a necessary condition of Hamiltonicity on OTIS graphs. A necessary and sufficient condition would be even more useful. Another important open direction is to see if the Independent Spanning Tree conjecture holds on any k-connected OTIS network for arbitrary values of k, as this is a most important aspect of fault tolerance is any distributed network.

We have also presented a polynomial time algorithm to find the longest path on a biconvex graph. The class of biconvex graphs is theoretically interesting because, the longest path problem is known to be NP-Hard on chordal bipartite graphs, which is a superclass of biconvex graphs, and polynomial time on class of bipartite permutation graphs, which is a subclass of biconvex graphs. The status of the problem was unresolved on biconvex graphs. We have also proposed a more time efficient absolute approximation algorithm for finding longest path on biconvex graphs.

We believe this algorithm can be extended to find Longest path on Convex graphs and a subclass of Split Graphs as well. It would be interesting to explore further in this direction. The status of the problem is still open on Convex graphs and it is NP-Hard for general Split Graphs.

REFERENCES

1. **Abbas, N.** and **L. K. Stewart** (2000*a*). Biconvex graphs: ordering and algorithms. *Discrete Applied Mathematics*, **103**(1-3), 1–19.

2. **Abbas, N.** and **L. K. Stewart** (2000*b*). Biconvex graphs: ordering and algorithms. *Discrete Appl. Math.*, **103**, 1–19. ISSN 0166-218X.

3. **Barth, D.** and **A. Raspaud** (1994). Two edge-disjoint hamiltonian cycles in the butterfly graph. *Information Processing Letters*, **51**(4), 175–179.

4. **Bjöandrklund, A.**, Determinant sums for undirected hamiltonicity. *In Foundations of Computer Science (FOCS), 2010*. 2010. ISSN 0272-5428.

5. **Brandstädt, A.**, **V. B. Le**, and **J. P. Spinrad**, *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. ISBN 0-89871-432-X.

6. **Broersma, H. J.** (2002). On some intriguing problems in hamiltonian graph theory–a survey. *Discrete Mathematics*, **251**(1-3), 47–69. ISSN 0012-365X.

7. **Carpenter, G. F.** (1990). The synthesis of deadlock-free interprocess communications. *Microprocessing and Microprogramming*, **30**(1-5), 695–701. ISSN 0165-6074. Proceedings Euromicro 90: Hardware and Software in System Engineering.

8. **Chen, W.**, **W. Xiao**, and **B. Parhami** (2009). Swapped (OTIS) networks built of connected basis networks are maximally fault tolerant. *Parallel and Distributed Systems, IEEE Transactions*, **20**(3), 361–366. ISSN 1045-9219.

9. **Dekel, E.** and **S. Sahni** (1984). A parallel matching algorithm for convex bipartite graphs and applications to scheduling. *Journal of Parallel and Distributed Computing*, **1**(2), 185–205. ISSN 0743-7315.

10. **Garey, M. R.** and **D. S. Johnson**, *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979. ISBN 0716710455.

11. **Glover, F.** (1967). Maximum matching in convex bipartite graph. *Naval Res. Logist. Quart.*, **14**, 313–316.

12. **Golumbic, M. C.**, *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004. ISBN 0444515305.

13. **Gould, R. J.** (1991). Updating the hamiltonian problema survey. *Journal of Graph Theory*, **15**(2), 121–157. ISSN 1097-0118.

14. **Hardgrave, W. W.** and **G. L. Nemhauser** (1962). On the relation between the traveling-salesman and the longest-path problems. *Operations Research*, **10**(5), pp. 647–657. ISSN 0030364X.

15. **Hoseinyfarahabady, M. R.** and **H. Sarbazi-Azad**, On pancyclicity properties of OTIS networks. *In HPCC'07*. 2007.

16. **Ioannidou, K.**, **G. B. Mertzios**, and **S. D. Nikolopoulos**, The longest path problem is polynomial on interval graphs. *In MFCS '09*. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-03815-0.

17. **Ioannidou, K.** and **S. D. Nikolopoulos**, The longest path problem is polynomial on cocomparability graphs. *In WG'10*. 2010.

18. **Itai, A.** and **M. Rodeh** (1988). The multi-tree approach to reliability in distributed networks. *Information and Computation*, **79**(1), 43–59. ISSN 0890-5401.

19. **Karger, D.**, **R. Motwani**, and **G. Ramkumar** (1997). On approximating the longest path in a graph. *Algorithmica*, **18**, 82–98. ISSN 0178-4617. 10.1007/BF02523689.

20. **Khuller, S.** and **B. Schieber** (1992). On independent spanning trees. *Information Processing Letters*, **42**(6), 321–323. ISSN 0020-0190.

21. **Krishnamoorthy, A. V.**, **P. J. Marchand**, **F. E. Kiamilev**, and **S. C. Esener** (1992). Grain-size considerations for optoelectronic multistage interconnection networks. *Applied Optics*, **31**(26), 5480–5507.

22. **Lipski, W.** and **F. P. Preparata** (1981). Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. *Acta Informatica*, **15**, 329–346. ISSN 0001-5903. 10.1007/BF00264533.

23. **Marsden, G. C.**, **P. J. Marchand**, **P. Harvey**, and **S. C. Esener** (1993). Optical transpose interconnection system architectures. *Optics Letters*, **18**(13), 1083–1085.

24. **Papadimitriou, C. M.**, *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994. ISBN 0201530821.

25. **Parhami, B.** (2005). The hamiltonicity of swapped OTIS networks built of hamiltonian component networks. *Information Processing Letters*, **95**(4), 441–445.

26. **Parhami, B.** (2006). A class of odd-radix chordal ring networks. *The CSI Journal on Computer Science and Engineering*, **4**(2&4), 1–9.

27. **Sahani, S.**, **E. Horowitz**, and **S. Rajasekaran**, *Fundamentals of Computer Algorithms*. Universities Press, 2007, 2nd edition.

28. **Sedgewick, R.** and **K. Wayne**, *Algorithms*. Amazon, 2002, 4th edition.

29. **Spinrad, J.**, **A. Brandstädt**, and **L. Stewart** (1987). Bipartite permutation graphs. *Discrete Applied Mathematics*, **18**(3), 279–292. ISSN 0166-218X.

30. **Uehara, R.** and **Y. Uno**, Efficient algorithms for the longest path problem. *In ISAAC*. 2004.

31. **Uehara, R.** and **G. Valiente** (2007). Linear structure of bipartite permutation graphs and the longest path problem. *Information Processing Letters*, **103**(2), 71–77. ISSN 0020-0190.

List of Papers from this Thesis

- **Esha Ghosh N. S. Narayanaswamy,** and **C. Pandu Rangan**. A Polynomial Time Algorithm for Longest Paths in Biconvex Graphs. *Proceedings of Workshop on Algorithms and Computation(WALCOM) 2011, pp.191-201, LNCS 6552, Springer-Verlag 2011.*

- **Esha Ghosh**, **Subhas K. Ghosh,** and **C. Pandu Rangan**. On the Fault Tolerance and Hamiltonicity of the Optical Transpose Interconnection System of Non-Hamiltonian Base Graphs. *under review*