Bandwidth, Edgesum and Profile of Graphs

Yung-Ling Lai
Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/dissertations

Part of the Computer Sciences Commons

Recommended Citation
http://scholarworks.wmich.edu/dissertations/1667

This Dissertation-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Dissertations by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
BANDWIDTH, EDGESUM AND PROFILE OF GRAPHS

by

Yung-Ling Lai

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
June 1997
BANDWIDTH, EDGESUM AND PROFILE OF GRAPHS

Yung-Ling Lai, Ph.D.
Western Michigan University, 1997

For a graph \( G = (V, E) \), each 1-1 mapping \( f : V \rightarrow \{1, 2, \ldots, |V|\} \) is called a proper numbering of \( G \). The bandwidth of graph \( G \) is \( \min \max |f(u) - f(v)| \), where the maximum is taken over each edge \( uv \in E(G) \), and the minimum is over all proper numberings \( f \). For graphs in general it is well known that the decision problem associated with finding bandwidth is NP-complete.

The edgesum of \( G \) is the number \( \min \sum_{uv \in E} |f(u) - f(v)| \), where the minimum is taken over all proper numberings \( f \). Determination of the edgesum for arbitrary graphs is known to be NP-complete.

For a proper numbering \( f \), the profile width \( w_f(v) \) of a vertex \( v \) in a graph \( G \) is the number \( w_f(v) = \max_{x \in N[v]} (f(v) - f(x)) \) where \( N[v] = \{ x \in V : x = v \text{ or } xv \in E \} \) is the closed neighborhood of \( v \). The profile of graph \( G \) is \( \min \sum_{v \in V(G)} w_f \) where the minimum is taken over all proper numberings \( f \). It is known that determination of the profile for arbitrary graphs is NP-complete.

The graph parameters bandwidth, edgesum and profile are examined in detail. The results of an extensive survey as to the solved bandwidth, edgesum and profile problems for classes of graphs are presented.

Graphs are appropriate models for many computer applications. Several
application areas are discussed.

The exact values of the profile of the composition of a path with other graphs, a cycle with other graphs, a complete graph with other graphs and a complete bipartite graph with other graphs are given. The exact value of the bandwidth of a butterfly is established. A polynomial time approximation algorithm to find the edgesum and profile of a butterfly is presented. An approximation algorithm to find the profile of a hypercube is presented. Several tight bounds on the profile of the corona of two graphs are developed and the exact value of the profile of the tensor product of a path with a complete bipartite graph is provided.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
For Jung-Chih and Alice

whose love, patience and understanding make this possible.
ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Professor Kenneth Williams for his continual guidance and support throughout this research. I am extremely grateful to Professor Gary Chartrand for his patience, guidance and support during the writing of this dissertation. I also wish to thank Professor Alfred Boals for serving on my committee.

I also like to thank Professor Ajay Gupta and Professor Dionysios Kountanis for their encouragement and friendship. I want to acknowledge Jin-Wen College in Taiwan. Without the financial support from there I would not have been able to finish this project. Many thanks also go to the brothers and sisters in KCCF for their direct and indirect help.

I am indebted to my parents. Their patience and love always encourage me throughout my life. I thank for my husband Jung-Chih for his understanding. The special gratitude goes to my daughter Alice for her sacrifice these years without a full time mother. With her smile and love, I can always find the hope for tomorrow.

Finally, I would like to dedicate this dissertation to our heavenly Father for His guidance and help in my life.

Yung-Ling Lai
TABLE OF CONTENTS

ACKNOWLEDGEMENTS .......................................................... ii
LIST OF TABLES ................................................................. vi
LIST OF FIGURES .............................................................. vii

CHAPTER

I. INTRODUCTION ................................................................. 1
   1.1 Definitions and Examples ......................................... 1
   1.2 A Survey of Known Results ...................................... 6

II. APPLICATION AREAS ..................................................... 12
   2.1 Solving Linear Equations ........................................... 12
   2.2 VLSI Layout ............................................................. 15
   2.3 Interconnection Networks ......................................... 17
   2.4 Constraint Satisfaction Problem ............................... 18
   2.5 The Visual Stimuli Application ................................. 20

III. SOME CORRECTIONS OF PREVIOUS RESULTS ............ 22
    3.1 Correction of Previous Result on $P_m \times C_n$ ........ 22
    3.2 Interval Graphs and Profile .................................... 23

IV. PROFILE OF COMPOSITION ........................................... 29
    4.1 Definition and Examples ........................................ 29

iii

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
# Table of Contents - Continued

## CHAPTER

4.2 Paths With Other Graphs .............................................. 30  
4.3 Cycles and Complete Graphs With Other Graphs .................. 33  
4.4 Complete Bipartite Graphs With Other Graphs .................. 35  

### V. NETWORK ARCHITECTURES ..................................................... 38  
5.1 Introduction ...................................................................... 38  
5.2 Butterfly Architecture ...................................................... 39  
5.3 Profiles of Hypercubes ...................................................... 46  

### VI. PROFILE OF CORONA ................................................................. 49  
6.1 Definition and Examples ................................................... 49  
6.2 Tight Bounds for General Cases ...................................... 50  
6.3 Tight Bounds on the Corona of Graphs With $\overline{K_m}$ ...... 54  
6.4 Special Cases ................................................................. 58  

### VII. PROFILE OF TENSOR PRODUCT .............................................. 65  
7.1 Definition and Examples ................................................... 65  
7.2 Profile of $P_n(T_P)K_{m,m}$ .............................................. 67  
7.3 Profile of $P_n(T_P)K_{r,m}$ for Even Values of $n$ .............. 68  
7.4 Profile of $P_n(T_P)K_{r,m}$ for Odd Values of $n$ ............. 71  

### VIII. SUMMARY ................................................................. 76
LIST OF TABLES

1. Corona and Cartesian Product ............................................. 9
2. Sum and Composition ............................................................. 10
3. Tensor Product and Strong Product ......................................... 11
4. Complete Bipartite Graph, Hypercube and Tree .......................... 11
LIST OF FIGURES

1. A Bandwidth Numbering for $P_4, C_5, K_{1,4}$ and $K_{2,3}$........................ 2
2. An Edgesum Numbering for $P_4, C_5, K_{1,4}$ and $K_{2,3}$........................ 3
3. Profile Numberings for $P_4, C_5, K_{1,4}$ and $K_{2,3}$............................. 4
4. Complementary Numberings, Different Profile Results.......................... 5
5. Lin and Yuan's [71] Numbering for Profile of $P_m \times C_n$.................... 23
6. Corrected Numbering for Profile of $P_m \times C_n$ When $2m \geq n$......... 23
7. The Induced Subgraphs Not Contained in Any Interval Graph.................. 24
8. An Interval Graph $G^*$................................................................. 25
9. Subgraph Induced by Removing $v_5$ From $G^*$.................................... 25
10. $H_3$ and $H_4$ ..................................................................................... 26
11. Another Interval Graph $G'$ .............................................................. 27
13. A 2-dimensional Butterfly ..................................................................... 40
14. A Bandwidth Numbering of 3-dimensional Butterfly............................. 42
15. Algorithm 1 Applied to a 3-dimensional Butterfly, $s_f(G) = 202$ ......... 44
16. Algorithm 2 Applied to a 3-dimensional Butterfly, $P_f(G) = 125$......... 46
17. A 4-dimensional Hypercube .................................................................. 46
18. Algorithm 3 Applied to a 4-dimensional Hypercube, $P_f(G) = 75$....... 48
List of Figures - Continued

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.</td>
<td>$P_3 \wedge P_4.$</td>
<td>49</td>
</tr>
<tr>
<td>20.</td>
<td>A Profile Numbering of $G = K_{1,5} \wedge K_1.$</td>
<td>58</td>
</tr>
<tr>
<td>21.</td>
<td>A Profile Numbering of $K_4 \wedge K_3.$</td>
<td>60</td>
</tr>
<tr>
<td>22.</td>
<td>$C_3(T_P)P_4.$</td>
<td>65</td>
</tr>
<tr>
<td>23.</td>
<td>Two Components of $P_6(T_P)K_{2,5}.$</td>
<td>67</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

1.1 Definitions and Examples

For a graph $G$, $V(G)$ denotes the set of vertices of $G$ and $E(G)$ denotes the set of edges of $G$.

Let $G = (V, E)$ be a graph on $n$ vertices. A 1-1 mapping $f : V \rightarrow \{1, 2, \ldots, n\}$ is called a proper numbering of $G$. The bandwidth $B_f(G)$ of a proper numbering $f$ of $G$ is the number

$$B_f(G) = \max\{|f(u) - f(v)| : uv \in E\},$$

and the bandwidth $B(G)$ of $G$ is the number

$$B(G) = \min\{B_f(G) : f \text{ is a proper numbering of } G\}.$$

A proper numbering $f$ is called a bandwidth numbering of $G$ if $B_f(G) = B(G)$.

For example, Figure 1 shows a bandwidth numbering for the graphs $P_4, C_5, K_{1,4}$ and $K_{2,3}$. In general, $B(P_n) = 1$, $B(C_n) = 2$, $B(K_{1,n}) = \lceil n/2 \rceil$ and $B(K_{m,n}) = m + \lceil n/2 \rceil - 1$ for $m \leq n$. 

1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 1. A Bandwidth Numbering for $P_4, C_5, K_{1,4}$ and $K_{2,3}$.

For a proper numbering $f$, the edgesum $s_f(G)$ produced by $f$ is given by

$$s_f(G) = \sum_{uv \in E(G)} |f(u) - f(v)|.$$ 

The edgesum $s(G)$ of a graph $G$ is defined by

$$s(G) = \min\{s_f(G) : f \text{ is a proper numbering of } G\}.$$ 

A proper numbering that achieves the edgesum of a graph is called an edgesum numbering. The term here called edgesum means exactly the same as the terms bandwidth sum or minimum sum which have been used by a number of previous investigators.

For example, Figure 2 shows an edgesum numbering for the graphs $P_4, C_5, K_{1,4}$ and $K_{2,3}$. In general, $s(P_n) = n - 1$ and $s(C_n) = 2(n - 1)$. Yao and Wang[86]
Figure 2. An Edgesum Numbering for $P_4, C_5, K_{1,4}$ and $K_{2,3}$.

showed that

$$s(K_{1,n}) = \begin{cases} \frac{n(n+2)}{4} & \text{if } n \equiv 0 \pmod{2} \\ (\frac{n+1}{2})^2 & \text{if } n \equiv 1 \pmod{2} \end{cases}$$

and for $m \leq n$, Williams[82] verified that

$$s(K_{m,n}) = \begin{cases} \frac{3n^2m - m^3 + 6m^2n + 4m}{12} & \text{if } n - m \equiv 0 \pmod{2} \\ \frac{3n^2m - m^3 + 6m^2n + m}{12} & \text{if } n - m \equiv 1 \pmod{2}. \end{cases}$$

Also, for a proper numbering $f$, the profile width $w_f(v)$ of a vertex $v$ in a graph $G$ is the number

$$w_f(v) = \max_{x \in N[v]} (f(v) - f(x))$$

where $N[v] = \{x \in V : x = v \text{ or } xv \in E\}$ is the closed neighborhood of $v$. The
A profile numbering $f$ of a graph $G$ is defined by

$$P_f(G) = \sum_{v \in V} w_f(v)$$

and the profile $P(G)$ of $G$ is the number

$$P(G) = \min\{P_f(G) : f \text{ is a proper numbering of } G\}.$$

A proper numbering $f$ is called a profile numbering of $G$ if $P_f(G) = P(G)$.

For example, Figure 3 shows a profile numbering for the graphs $P_4$, $C_5$, $K_{1,4}$ and $K_{2,3}$. In general, $P(P_n) = n - 1$, $P(C_n) = 2n - 3$, $P(K_{1,n}) = n$ and for $m \leq n$, $P(K_{m,n}) = mn + m(m - 1)/2$ (see Lin and Yuan[53]).

For a proper numbering $f$ on a graph $G$ of order $n$, the complementary numbering $f'$ on $G$ is defined by $f'(v) = n + 1 - f(v)$ for each vertex $v$ in $G$. Then $B_{f'}(G) = B_f(G)$ and $s_{f'}(G) = s_f(G)$. Thus the complementary numbering of any bandwidth (or edgesum) numbering is also a bandwidth (or edgesum) numbering. However this relation does not hold for profile numberings, as is...
The decision problem corresponding to finding the bandwidth of an arbitrary graph was shown to be NP-complete by Papadimitriou[68]. Garey, Graham, Johnson and Knuth[24] showed that the problem is NP-complete even for trees of maximum degree 3. The decision problem associated with determining the edge-sum for an arbitrary graph (sometimes call linear layout or the linear arrangement problem) was shown to be NP-complete by Garey, Johnson and Stockmeyer[26]; but the edgesum for trees is polynomial. In fact, Chung[14] provided the most efficient algorithm known to date to achieve an edgesum numbering for arbitrary trees. Also, Chung[15] found the edgesum for the complete binary trees. Lin and Yuan[53] indicated that the profile minimization problem of an arbitrary graph is equivalent to the interval graph completion problem, which was shown to be NP-complete by Garey and Johnson[25]. Kuo and Chang[42] provides a polynomial algorithm to achieve a profile numbering for an arbitrary tree of order $n$.  

![Figure 4. Complementary Numberings, Different Profile Results.](image)
1.2 A Survey of Known Results

A large number of approximation algorithms for the bandwidths of a variety of graphs have been given. Approximations have been developed for general graphs (see Cheng[9], Cuthill and McKee[18], Gibbs, Poole, Jr. and Stockmeyer[27], GowriSankaran and Opatrny[30], Jeff[39], King[41], Luo[60], Quoc and O'Leary[69], Smyth[74], Wiegers and Monien[81]) and also for trees or caterpillars (see GowriSankaran, Miller and Opatrny[29], Odlyzko and Wilf[67], Haralambides, Makedon and Monien[31], and Smithline[73]).

For the profile of general graphs, Everstine[21] gives a comparison of three profile approximation algorithms given by Cuthill and McKee[18], Gibbs, Poole, Jr. and Stockmeyer[27] (their algorithms are called GPS) and Levy[51]. Among those three, GPS is exceptionally fast and the best able to reduce profile. Koo and Lee[43], Luo[60], Quoc and O'LearySnay[69], Snay[75], and Wiegers and Monien[81] also give approximation algorithms to reduce the profile of general graphs.

A number of upper and lower bounds are known which relate bandwidth to various graph invariants (see Bascuñán and Ruiz[2], Bascuñán, Ruiz and Slater[3], Brualdi and McDougal[5], Chinn, Chvátalová, Dewdney and Gibbs[10], de la Véga[19], Erdös, Hell and Winkler[22], Jeffs[39], Leung, Vornberger and Witthoff[50], Lin[52], and Miller[64]). Some upper and lower bounds are known for edge-sum problems (see Yao and Wang[86]); however, the exact values of the band-
width, edgesum and profile have only been discovered for a few classes of graphs. These classes include paths, cycles, complete graphs, complements of complete graphs, and stars. The exact value of the bandwidth of some planar graphs (see Hochberg[36]), bandwidth of triangulated cycles (see Hochberg, McDiarmid and Saks[37]), bandwidth of the $k$-th power of paths (see Lee, Saba and Sun[49]) and certain graphs built from other graphs have also been determined. Surveys by Chung[13] and Chinn, Chvátalová, Dewdney and Gibbs[10] contain a number of results pertaining to solved bandwidth problems.

Tables 1 through 4 summarize the known exact results for bandwidth, edgesum, and profile on graphs built from other graphs.

The corona of graphs $G_1$ and $G_2$, on $n_1$ and $n_2$ vertices respectively, is denoted by $G_1 \circ G_2$ and contains one copy of $G_1$ and $n_1$ copies of $G_2$. Each distinct vertex of $G_1$ is joined to every vertex of the corresponding copy of $G_2$.

The Cartesian product of two graphs $G$ and $H$, denoted $G \times H$, is the graph with vertex set $V(G) \times V(H)$ and $(u_1, v_1)$ is adjacent to $(u_2, v_2)$ if either $u_1$ is adjacent to $u_2$ in $G$ and $v_1 = v_2$ or $u_1 = u_2$ and $v_1$ is adjacent to $v_2$ in $H$.

The sum $G_1 + G_2 + \ldots + G_k$ (also known as join) of $k$ pairwise disjoint graphs for some $k \geq 2$ is the graph with vertex set $V(G) = V(G_1) \cup V(G_2) \cup \ldots \cup V(G_k)$ and edge set $E(G) = \bigcup_{i=1}^{k} E(G_i) \cup \{(u, v) : u \in V(G_i), v \in V(G_j) \text{ and } i \neq j\}$.

The composition of two graphs $G$ and $H$, denoted $G[H]$, is the graph with vertex set $V(G) \times V(H)$ and $(u_1, v_1)$ is adjacent to $(u_2, v_2)$ if either $u_1$ is adjacent
to \( u_2 \) in \( G \) or \( u_1 = u_2 \) and \( v_1 \) is adjacent to \( v_2 \) in \( H \).

The tensor product of graphs \( G_1 \) and \( G_2 \), denoted \( G_1 \times G_2 \), is the graph with vertex set \( V(G_1) \times V(G_2) \) and \((u_1, v_1)\) is adjacent to \((u_2, v_2)\) if \((u_1, u_2) \in E(G_1)\) and \((v_1, v_2) \in E(G_2)\).

The strong product of graphs \( G_1 \) and \( G_2 \), denoted \( G_1 \circ G_2 \), is the graph with vertex set \( V(G_1) \times V(G_2) \) and \((u_1, v_1)\) is adjacent to \((u_2, v_2)\) if one of the following holds: (a) \((x_1, x_2) \in E(G_1)\) and \((y_1, y_2) \in E(G_2)\), (b) \(x_1 = x_2\) and \((y_1, y_2) \in E(G_2)\), or (c) \(y_1 = y_2\) and \((x_1, x_2) \in E(G_1)\).

A graph \( G \) is bipartite if it is possible to partition \( V(G) \) into two subsets \( V_1 \) and \( V_2 \) such that every element of \( E(G) \) has one endvertex in \( V_1 \) and another endvertex in \( V_2 \). A complete bipartite graph \( G = (V, E) = K_{m,n} \) is a bipartite graph with partite sets \( V_1, V_2 \) where \(|V_1| = m, |V_2| = n, V_1 \cup V_2 = V\) and \( E = \{(uv) : u \in V_1 \text{ and } v \in V_2\} \).

The \( d \)-dimensional hypercube has \( n = 2^d \) vertices and \( d2^{d-1} \) edges. Each vertex corresponds to a \( d \)-bit binary number, and two vertices are adjacent if and only if their binary number differs in only one bit.

A tree is an acyclic connected graph.
Table 1
Corona and Cartesian Product

<table>
<thead>
<tr>
<th></th>
<th>Corona</th>
<th>Cartesian Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>Give bounds for two graphs, solved for complete graph</td>
<td>Solved for path with path and path with cycle [16], [17], [35]</td>
</tr>
<tr>
<td></td>
<td>with complete graph, cycle and path with $K_1$ and cycle with $i$ copies of $K_1$</td>
<td>Solved for cycle with cycle [35], [48]</td>
</tr>
<tr>
<td></td>
<td>[11]</td>
<td>Solved for complete graph with path, cycle and complete graph [35]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Give bounds for two graphs and $k$ graphs [16], [17]</td>
</tr>
<tr>
<td>Edgesum</td>
<td>Give bounds for two graphs, solved for path with path</td>
<td>Solved for path with path and cycle with cycle [65]</td>
</tr>
<tr>
<td></td>
<td>and complete graphs with $K_1$ [85]</td>
<td>Solved for $m$ complete graphs [55]</td>
</tr>
<tr>
<td>Profile</td>
<td></td>
<td>Solved for path with path and path with cycle [54]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Solved for path with complete graph, cycle with complete graph and cycle with cycle [61]</td>
</tr>
</tbody>
</table>
Table 2

Sum and Composition

<table>
<thead>
<tr>
<th></th>
<th>Sum</th>
<th>Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>Solved for two graphs [57], [87]</td>
<td>Solved for complete graph with other graph [35], [58]</td>
</tr>
<tr>
<td></td>
<td>Solved for k graphs [45]</td>
<td>Solved for certain graph powers [12]</td>
</tr>
<tr>
<td></td>
<td>Solved for k graphs [47]</td>
<td>Solved for star with other graph [58]</td>
</tr>
<tr>
<td></td>
<td>Solved for certain graph sums [82]</td>
<td>Solved for path and cycle with other graph [35]</td>
</tr>
<tr>
<td>Edgesum</td>
<td>Solved for k graphs [47]</td>
<td>Solved for complete graph with path and complete graph with cycle [58]</td>
</tr>
<tr>
<td></td>
<td>Solved for certain graph sums [82]</td>
<td>Solved for path with path and path with cycle [56]</td>
</tr>
<tr>
<td>Profile</td>
<td>Solved for two graphs [53]</td>
<td></td>
</tr>
</tbody>
</table>
Table 3

Tensor Product and Strong Product

<table>
<thead>
<tr>
<th></th>
<th>Tensor Product</th>
<th>Strong Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td>Solved for paths and cycles with complete graphs [83]</td>
<td>Solved for path with path</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[35], [48], [88]</td>
</tr>
<tr>
<td></td>
<td>Solved for paths with complete bipartite graphs [84]</td>
<td>Solved for complete graph with path and complete graph with cycle [35]</td>
</tr>
<tr>
<td></td>
<td>Solved for path with path, path with cycle and cycle with cycle [46]</td>
<td>Solved for cycle with cycle and path with cycle [48], [88]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Give upper bound for two graphs [88]</td>
</tr>
<tr>
<td>Edgesum</td>
<td>Solved for paths with complete bipartite graphs [84]</td>
<td></td>
</tr>
</tbody>
</table>

Table 4

Complete Bipartite Graph, Hypercube and Tree

<table>
<thead>
<tr>
<th></th>
<th>Complete Bipartite Graph</th>
<th>Hypercube</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edgesum</td>
<td>Solved [82], [86]</td>
<td>Solved [34]</td>
<td>Solved [14], [28], [71]</td>
</tr>
<tr>
<td>Profile</td>
<td>Solved [53]</td>
<td></td>
<td>Solved [42]</td>
</tr>
</tbody>
</table>
CHAPTER II

APPLICATION AREAS

Bandwidth, edgesum and profile are useful parameters for many applications. In this chapter, we discuss some of the better known application areas.

2.1 Solving Linear Equations

The analysis of network systems with digital computers requires the solution of a large number of linear equations. The use of the finite element method also involves solving a large set of linear algebraic equations of the form

\[ [A][X] = [B] \]

where \([A]\) is a large symmetric sparse matrix. The matrix \([A]\) is said to be sparse if the number of nonzero entries is small in comparison with its total number of entries. Such matrices also appear very often as coefficient matrices of systems of differential equations in numerical analysis and physics.

Although storage availability and internal speed has been greatly increased in recent years, the solution of equations still takes a huge amount of space and time. To store such an \(n \times n\) matrix would require storage of all its \(n^2\) entries, and most of them are 0s. When performing operations on these matrices, a large number of the computations only involve multiplying or adding 0s. If we can somehow focus on the nonzero entries, discarding the zero entries, that will save
a large portion of the storage space and it may also save some computation time.

The focus on the nonzero entries of a matrix $[A]$ is easier if all these entries are grouped close to the main diagonals of $[A]$ and are said to be banded. A small band of $[A]$ allows the use of relatively little memory for storing $[A]$ since one can keep track of each nonzero entry by simply recording the row to which it belongs and the distance from the diagonal in the row. Among the solution methods of linear equations, the standard Gauss algorithm is well known and most other methods are only modifications of this basic algorithm to reduce the number of calculations. The small band of $[A]$ makes it possible to carry out only a small fraction of all the computations involved and still get the desired result since the remaining computations involve all zero entries and thus have predictable results.

Of course, the small band does not always occur in the matrices we deal with. When the matrix $[A]$ does not have a small band, we may permute the rows and columns of $[A]$ in order to obtain a matrix that has a smaller band. In other words, we may rearrange the columns and rows of $[A]$ to translate the original system of equations $[A][X] = [B]$ into an equivalent system $[A'][X'] = [B']$ where $[A']$ is a symmetric matrix with a smaller band than $[A]$. Then we may work with $[A']$ rather than with $[A]$.

We know that there is a direct one-to-one relationship between symmetric matrices and graphs. The position of the nonzero entries of an $n \times n$ symmetric matrix can define an adjacency matrix of a graph $G$ on $n$ vertices (See Jeffs[39]).
That is, each row or column of $[A]$ can be represented by a vertex of $G$ and the edge $uv \in E(G)$ if and only if the entry in column $u$, row $v$ of the matrix $[A]$ is nonzero. For an $n \times n$ matrix $[A]$, define the column height $P_j$ of column $j$ ($1 \leq j \leq n$) as

$$P_j = \begin{cases} 
0 & \text{if } a_{ij} = 0 \text{ for } 1 \leq i \leq j \\
 j - i & \text{i is the smallest integer such that } a_{ij} \neq 0
\end{cases}$$

Then the bandwidth of the matrix $A$ denoted $B(A)$ is $\max_j P_j$ and the profile of the matrix denoted $P(A)$ is $\sum_{j=1}^n P_j$. This definition is equivalent to the bandwidth and profile definition on the corresponding graph.

The smallest possible maximum column height achievable (over all row and column permutations) for a given matrix $[A]$ is equivalent to the bandwidth of the corresponding graph $G$. And the execution time arising from solving the system of linear equations is proportional to the sum of the squares of the column heights. See King[41]. For storage, the bandwidth represents the maximum length of a column which must be stored, and the profile represents the total amount of storage needed. A number of papers address this application area, including Cheng[9], Everstine[21] and Jennings[40], King[41], Koo and Lee[43], Lin and Yuan[54], Luo[60], Miller[63], Quoc and O'Leary[69], Snay[75] and Veldhorst[79].
2.2 VLSI Layout

Given a set of modules, the VLSI layout problem consists of placing the modules on a two-dimensional grid in a non-overlapping manner and then wiring together the terminals on the different modules according to a given wiring specification in such a way that the wires do not interfere with each other. Thus, there are two stages in VLSI layout: placing the modules on a board, called the placement problem, and then after the modules are situated, wiring together the terminals of different modules that should be connected. This is called the routing problem.

The first formal model for VLSI layout was developed by Thompson [76], [77]. The model is consistent with the VLSI design rules established by Mead and Conway [62] and is also similar to the widely used Manhattan wiring model. There is a natural one-to-one correspondence between VLSI circuits and graphs. Assume that the graphs are of bounded degrees and that vertices require only a constant area of silicon. We can then model a VLSI circuit in a graph, with the vertices representing the modules and the edges of the graph representing the wires. The graph provides a simplified model of the circuit which can help us to obtain better solutions for the real-world model. No deterministic exact algorithms are known for placement and routing problems in the real world and the techniques used are based on more or less efficient heuristic algorithms.

The goal of the placement problem is to place the modules in such a way
that the total wire-length is minimized. The wiring on a VLSI chip is restricted
to following along grid tracks and is not allowed to overlap on the same track
although vertical path segment may cross a horizontal path segment. In other
words, routing is done by a grid of vertical and horizontal tracks or channels (see
Nanan and Kurtzberg[66], Shing and Hu[72]). Let \( d_{ij} = |x_i - x_j| + |y_i - y_j| \)
be the distance function that measures the real wire-length between two pins
\((x_i, y_i)\) and \((x_j, y_j)\), and let \( c_{ij} \) be the number of wires between the corresponding
modules. Then we want to find the placement of the modules in such a way
that it minimizes the function \( \sum_{i \neq j} c_{ij} d_{ij} \). This problem is equivalent to finding
a numbering on the corresponding graph \( G \) which minimizes \( \sum_{uv \in E} |f(u) - f(v)| \)
which is the edgesum of the graph \( G \). This problem is also known as \textit{Optimal
Linear Ordering} (see Adolphson and Hu[1]).

The bandwidth measures the maximum distance between modules. Signals
do not propagate instantaneously across wires, and the longer the wire, the longer
the propagation delay. In pipelined or systolic systems, the effect of propagation
delays is even more dramatic. The maximum delay determines the clockperiod,
and hence the throughput, of the system. Therefore, minimizing the bandwidth is
equivalent to minimizing the delay communication between modules which is an
important parameter when solving the routing problem of VLSI layout. Several
papers discuss this application areas including Adolphson and Hu[1], Bhatt and
Leighton[4], Diaz[20], Miller[63], Shing and Hu[72] and Ullman[78].
We can use a graph $G = (V, E)$ to represent an interconnection network (sometimes called a parallel computation network). Each vertex of $G$ represents a different computer and $uv \in E$ if and only if there is a direct link between computers $u$ and $v$. In the network, each computer receives part or all of the original input data and the master program will control all the computers and specify what computation need to be performed on each computer. At every time unit each computer can pass the results of the computation to one of its neighboring computers (the ones joined to it by an edge) and these neighbors will use those results as inputs for their own computations later.

If we have a problem $P$ which needs to be solved on an interconnection network $G$ when $G$ is not available, but there is another interconnection network $H$ available, we might want to "simulate" the program for $G$ by a program for $H$ which solves the same problem. The simulation of $G$ by $H$ is a way of describing how to use the program $P$ as a guide so that $H$ can accomplish the same task as $G$ by assigning its computers the tasks assigned to those of $G$.

We seek a one-to-one mapping (embedding) $f$ from $G$ to $H$. Each computation of the program $P$ at a computer $x$ in $G$ will now be replaced by the same computation at the corresponding computer $f(x)$ in $H$. Also, each communication of $P$ between connected computers $x_i$ and $x_j$ will be replaced by $f(x_i)$ and $f(x_j)$ in $H$. The efficiency of the map $f$ is measured by the time delay factor (dilation).
d since whereas \( x_i \) and \( x_j \) could communicate in some unit time \( t \) because they are neighbors, the vertices \( f(x_i) \) and \( f(x_j) \) require communication time \( dt \) where \( d \) is the distance between \( f(x_i) \) and \( f(x_j) \) in \( H \). The bandwidth of \( G \) represents the worst possible delay (dilation) of the embedding from an interconnection network \( G \) to a linear array (path).

It is also natural to consider the average time delay caused by this embedding. Although the dilation might be large, the embedding might still be a good map if it has a small time delay on a large fraction of all the edges of \( G \). We would calculate the average by summing the individual delays and dividing by the total number of edges in \( G \). The edgesum of \( G \) represents the sum of each of the individual delays. Taking the ratio of the edgesum to the size of \( G \), one might ask for the smallest possible average time delay over all possible mappings. This would be provided by an edgesum numbering. (See Miller[63].)

2.4 Constraint Satisfaction Problem

Graph bandwidth also has some applications to a class of search problems known as constraint satisfaction problems (CSPs) from the field of artificial intelligence. The discussion of CSPs in this section closely follows the problem description given in Zabih[89]. CSPs have an associated constraint graph. In the graph the vertices represent the variables of the search problem, and there is an edge between two vertices if there is a (nontrivial) constraint between those
variables. Graph bandwidth provides a link between the syntactic structure of a constraint satisfaction problem and the complexity of the underlying search task.

A CSP has a set of variables and a domain of values. Every variable must be assigned a value. A CSP also consists of some constraints describing which assignments are compatible. Most interesting problems are binary CSPs where the constraints involve pairs of variables. Such a constraint consists of the simultaneously permitted assignments. The constraint graph associated with the binary CSP has edge set \( E = \{(u_i, u_j) : \text{there is a constraint between } u_i \text{ and } u_j\} \). The constraint graph hides a great deal of the information about the search problem, particularly the tightness of the constraints.

There are several reasons to believe that the bandwidth of the constraint graph of a CSP reflects the locality of the search problem. If the CSP has limited bandwidth, each vertex in the constraint graph can have no more than a fixed number of neighbors. This suggests that graphs with small bandwidth should be solvable by only dealing with a small subset of the variables at any instant. A stronger argument can be made on the basis of the claim that nearly decomposable search problems should be easy to solve. In particular, it should be possible to solve them efficiently by solving their subparts more or less independently and then by using divide and conquer (or dynamic programming) to put them together into a solution. The bandwidth of the constraint graph of a CSP serves as a measure of its decomposability. It turns out that any CSP of limited bandwidth can be
solved in polynomial time by dynamic programming. The basic strategy is to find an ordering with minimal bandwidth, and then to use this ordering to solve the CSP.

When using backtrack search, the decision actually responsible for a failure can occur significantly before the failure itself is detected. The greater the number of intervening decisions, the worse the backtracking performs. Our goal is to prune the search tree as much as possible (see [70], [6]). The search trees resulting from small bandwidth orderings are significantly smaller than those resulting from orderings with greater bandwidth. If a CSP is searched in a fixed order using intelligent backtracking, the bandwidth of that ordering provides a bound on the amount of the search tree that intelligent backtracking will prune. It is possible to obtain restrictions on the nodes in the search tree, where these exceptions can occur in terms of the k-consistency (see [23]) of the original CSP. So the bandwidth of a search ordering can provide a measure of its quality. If small bandwidth orderings are really useful, then the large body of heuristics that has been developed by numerical analysts for finding such orderings may prove to be useful for solving CSPs. For more detail about this application, see Zabih[89].

2.5 The Visual Stimuli Application

There are some other applications of edgesum problems such as representing two-dimensional arrays on a sequential file in a computer. If one wishes to
perform local calculations around a point in the array, as in the case of evaluating a differential operator, then \(|f(u) - f(v)|\) measures the distance that must be traversed between \(u\) and \(v\) which is a local operation in the file. The edgesum measures the total cost of the operation. The following description of the problem is from Mitchison and Durbin[65], page 571.

An analogous problem in computer hardware occurs when placing components of a multi-dimensional array processor on a lower dimensional chassis. Originally interest in this problem was raised by a biological question. The cortex of the brain of higher mammals can be regarded as a sheet of nerve cells. In the part of vortex devoted to vision, cells respond to certain visual stimuli, such as oriented bars of light against a dark background. A major discovery in recent years is that variables used to describe these stimuli, such as the location of an edge in space or its orientation, are mapped in a systematic manner on the cortex (see [38]). This mapping of more than two variables onto a two-dimensional sheet is accomplished by cycling through the values of variables to give striped patterns. This suggests that the nervous system may be trying to achieve as much continuity as possible in mapping these variables onto the cortex. The numbering of an array represents the most simplified mathematical model of this problem.
CHAPTER III

SOME CORRECTIONS OF PREVIOUS RESULTS

3.1 Correction of Previous Result on $P_m \times C_n$

**Definition 1** Given two graphs $G$ and $H$, the *cartesian product* of $G$ and $H$ denoted by $G \times H$ is the graph with vertex set $V(G) \times V(H)$ and $(u_1, v_1)$ is adjacent to $(u_2, v_2)$ if (a) $u_1$ is adjacent to $u_2$ in $G$ and $v_1 = v_2$ or (b) $u_1 = u_2$ and $v_1$ is adjacent to $v_2$ in $H$.

In [54], Lin and Yuan described the profile of $P_m \times P_n$ of $P_{m,n}$ (defined in [54]) and of $P_m \times C_n$. The result they provide for $P_m \times C_n$ with $2m \geq n$ is

$$P(P_m \times C_n) = \begin{cases} 
mn^2 - \frac{n}{12}(2n^2 - 3n + 16), & n \equiv 0 \pmod{2} \\
\frac{1}{12}(2n^3 - 3n^2 + 16n - 3), & n \equiv 1 \pmod{2}.
\end{cases}$$

The numbering pattern they provide to achieve this result is given in Figure 5. However, this numbering pattern is in error and is not actually optimal. A corrected pattern is shown in Figure 6. The proper numbering $f$ of Figure 5 as given in [54] is $P_f(P_4 \times C_6) = 111$. However, the correct profile numbering as illustrated in Figure 6 gives the correct profile, which is $P(P_4 \times C_6) = 109$.  

22
3.2 Interval Graphs and Profile

**Definition 2** Let $J_1, J_2, \ldots, J_n$ be intervals on a line. An interval graph $G = (V, E)$ is a graph with vertex set $V = \{J_1, J_2, \ldots, J_n\}$ and $(J_i, J_k) \in E$ if and only if intervals $J_i$ and $J_k$ have a point in common.

**Proposition 1** (From Lovasz[59]) A graph $G$ is an interval graph if and only if $G$ does not contain any of the graphs shown in Figure 7 as an induced subgraph.

The following three results are all from Lin and Yuan[53]. The numbers identifying these results are taken directly from [53] and do not represent identifying numbers in this dissertation.

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Lemma 1.2 A graph $G$ is an interval graph if and only if there exists a numbering $f$ such that if $f(x) < f(y) < f(z)$ and $xz \in E(G)$ then $yz \in E(G)$.

Theorem 1.3 For any graph $G$, $P(G) \geq |E(G)|$; and $P(G) = |E(G)|$ if and only if $G$ is an interval graph.

Theorem 1.4 For any graph $G$, $P(G)$ is the minimum number of edges of an interval supergraph of $G$.

Each of the results stated from [53] will be shown to be in error in this section.

Lemma 1 Let $G^* = (V, E)$ be as shown in Figure 8. Then $G^*$ does not contain any of the graphs shown in Figure 7 as an induced subgraph.

Proof: Let $H_1$ be the graph shown in Figure 7 (A). Note that $|V(H_1)| = 7$ and $|E(H_1)| = 6$. But $|V(G^*)| = 8$, $|E(G^*)| = 13$ and there is no vertex in $G^*$ of degree 7, so $G^*$ does not contain $H_1$ as an induced subgraph.
Let $H_2$ be the graph shown in Figure 7 (B). There are three vertices in $H_2$ of degree 4 but only two vertices in $G^*$ of degree greater than 3. So $G^*$ does not contain $H_2$ as an induced subgraph.

Assume there is an induced $C_k$ for $k > 4$ in $G$. Suppose $v_5$ is not on $C_k$. Then the subgraph induced by removing $v_5$ from $G^*$ is $(V', E')$ where $V' = V - \{v_5\}$ and $E' = \{(v_1, v_2), (v_2, v_6), (v_3, v_4), (v_4, v_6), (v_6, v_7), (v_6, v_8)\}$ (as shown in Figure 9) which does not contain an induced $C_k$ for $k > 4$. Suppose $v_5$ is on $C_k$. Note that $v_5$ has degree 6 and there are only two vertices adjacent to $v_5$ that can be on $C_k$. Since $k > 4$, $v_8$ must be on $C_k$. But $v_8$ is adjacent to $v_6$ and $v_7$, and $(v_6, v_7) \in E$. So $G^*$ does not contain $C_k$ for $k > 4$ as an induced subgraph.

In group (D) of Figure 7, we only need to show that $G^*$ does not contain either $H_3$ or $H_4$ (shown in Figure 10) as an induced subgraph. For $H_3$, note that
$|V(H_3)| = 7$ and $|E(H_3)| = 8$, and $v_5$ is the only vertex in $G$ with degree 6. But $(G - v_5) \not\cong H_3$. So $G^*$ does not contain $H_3$ as an induced subgraph. For $H_4$, note that $|V(H_4)| = 6$ and $|E(H_4)| = 6$. The maximum degree of the vertices in $H_4$ is 3. $v_5$ has degree 6 in $G^*$, there is no way to remove two vertices from $G^*$ and cause $v_5$ to have degree less than 4. So $v_5$ must not be on the induced subgraph of $H_4$. There is only one vertex with degree greater than 2 in $(G^* - v_5)$. So $G$ does not contain $H_4$ as an induced subgraph. Hence, $G^*$ does not contain any of the graphs shown in Figure 7 as an induced subgraph. □

By Proposition 1 and Lemma 1 we have the following theorem.

**Theorem 1** The graph $G^*$ of Figure 8 is an interval graph.

Through exhaustive computer testing, we know that there does not exists a numbering $f$ such that if $f(x) < f(y) < f(z)$ and $xz \in E(G)$ then $yx \in E(G)$. So Lemma 1.2 in [53] has been disproved. Also by exhaustive computer testing, we have $P(G^*) = 14$. In fact, in Figure 8, if we let $f(v_i) = i$ for $1 \leq i \leq 8$, then $f$ is a profile numbering which achieves $P_f(G^*) = P(G^*) = 14$. Since $|E(G^*)| = 13$, we have the following theorem.
2

Figure 11. Another Interval Graph $G'$.

Theorem 2 There exists an interval graph $G = (V, E)$ such that $P(G)$ exceeds $|E(G)|$.

It can be shown in a similar way that the graph $G'$ in Figure 11 is an interval graph and $|E(G')| = 17$. Through exhaustive computer testing, we know that $P(G') = 19$. The two examples $G^*$ and $G'$ not only disprove Theorem 1.3 in [53] but also tell us that the difference between the profile and the size of an interval graph need not be a fixed number.

Also by exhaustive computer testing, we obtain the following lemma.

Lemma 2 If $G = K_{1,3} \times P_2$, then $P(G) = 14$.

By Lemmas 1 and 2 we know that $G^*$ is an interval supergraph of $G$ and $P(G) = 14 > 13 = |E(G^*)|$. So Theorem 1.4 in [53] has been disproved.

Thus each of Lemma 1.2, Theorem 1.3 and Theorem 1.4 is now shown to be false. However, the first part of Theorem 1.3 is true and we state and prove this as our next theorem.
Theorem 3 For any graph G, \( P(G) \geq |E(G)| \).

Proof: Assume there is a graph G with \( P(G) < |E(G)| \). Let \( f \) be a profile numbering of G. Define \( e_f(v) = |\{ u : uv \in E(G) \text{ and } f(u) < f(v) \}| \). It is clear that for all \( v \in V(G) \), \( w_f(v) \geq e_f(v) \). Then we have \( P(G) = P_f(G) = \sum_{v \in V(G)} w_f(v) \geq \sum_{v \in V(G)} e_f(v) = |E(G)| \) contradicting our assumption. \( \square \)

The following theorem corrects the result in Lemma 1.3.

Theorem 4 Given a graph G = (V,E). Then \( P(G) = |E(G)| \) if and only if there exist a numbering f such that if \( f(x) < f(y) < f(z) \) and \( xz \in E(G) \) then \( yz \in E(G) \).

Proof: Define \( e_f(v) = |\{ u : uv \in E(G) \text{ and } f(u) < f(v) \}| \). Note that \( \sum_{v \in V} e_f(v) = |E(G)| \).

First, let \( f \) be a numbering such that if \( f(x) < f(y) < f(z) \) and \( xz \in E(G) \) then \( yz \in E(G) \). We have \( w_f(v) = e_f(v) \) for all \( v \in V \). So \( P_f(G) = \sum_{v \in V} w_f(v) = \sum_{v \in V} e_f(v) = |E(G)| \). Then we have \( P(G) \leq P_f(G) = |E(G)| \). By Theorem 2 we know that \( P(G) \geq |E(G)| \). Therefore \( P(G) = |E(G)| \).

Now suppose \( P(G) = |E(G)| \). Let \( f \) be a profile numbering. For all \( v \in V(G) \), it is clear that \( e_f(v) \leq w_f(v) \). Assume that there exists \( v \) such that \( e_f(v) < w_f(v) \). Then \( P_f(G) = \sum_{v \in V} w_f(v) > \sum_{v \in V} e_f(v) = |E(G)| \) which is a contradiction. Hence \( w_f(v) = e_f(v) \) for all \( v \in V \). Thus we have if \( f(x) < f(y) < f(z) \) and \( xz \in E(G) \) then \( yz \in E(G) \). \( \square \)
CHAPTER IV

PROFILE OF COMPOSITION

4.1 Definition and Examples

Definition 3 The composition $G[H]$ of a graph $G$ with a graph $H$ is the graph with vertex set $V(G) \times V(H)$ such that $(u_1, v_1)$ is adjacent to $(u_2, v_2)$ if either $u_1$ is adjacent to $u_2$ in $G$ or if $u_1 = v_2$ and $v_1$ is adjacent to $v_2$ in $H$.

Figure 12 shows $P_3[P_4]$. The composition problems for a complete graph with a path, a complete graph with a cycle, a path with a path, and a path with a cycle have been solved for bandwidth and edgesum. In this chapter, we investigate the profile of the composition of paths, cycles, complete graphs and complete bipartite graphs with other graphs.

For a composition graph $G[H] = (V, E)$ with graphs $G$ of order $m$ and $H$ of order $n$, we represent the vertex set as $V = \{v_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\}$ where
column $j$ is denoted by $Q_j$ ($1 \leq j \leq n$), which represents a copy of $G$, and row $i$ ($1 \leq i \leq m$) is denoted by $R_i$, which represents a copy of $H$.

The following result from [54] is essential for the work done in this chapter.

**Proposition 2** (From Lin and Yuan[54]) Let $G$ be a graph of order $n$. For any proper numbering $f$ of $G$,

$$P_f(G) = \sum_{i=1}^{n} |N(S_i)| \text{ where } S_t = \{v : v \in V(G) \text{ and } f(v) \leq t\}.$$  

4.2 Paths With Other Graphs

In this section, we establish the profile of the composition of paths with other graphs.

**Theorem 5** Let $G = P_m[H]$ where $H$ is a graph with $n$ vertices. Then

$$P(G) = \begin{cases} 
2P(H) + \frac{mn(3n-1)}{2} - 2n^2 + n & \text{for } m \geq 3 \text{ and } n > 1 \\
\frac{3n^2 - n}{2} & \text{for } m = 2 \text{ and } n > 1 \\
P(H) & \text{for } m = 1 \\
m - 1 & \text{for } n = 1
\end{cases}$$

Proof: For $n = 1$, $G = P_m$; so $P(G) = m - 1$. For $m = 1$, $G = P_1[H] = H$; so $P(G) = P(H)$. For $m = 2$, notice that every vertex in $R_2$ is adjacent to every vertex in $R_1$, once we number a vertex in both rows, all of the unnumbered
vertices are in $N_f(S)$. By Proposition 2 we know that the profile numbering of $G$ must completely number one row before numbering any vertex in the other row. This will minimize $\sum_{k=1}^{mn} |N(S_k)|$ which in turn minimizes $P_f(G)$. Without loss of generality, assume we completely number $R_1$ and then $R_2$. We want to number the vertices in $R_1$ in the order of a profile numbering (say $f$) of $H$. Since every vertex in $R_2$ is adjacent to the vertex $f^{-1}(1)$, it does not matter how we number the vertices in $R_2$. Hence, $P(G) = P(H) + \sum_{i=0}^{n-1} (n + i) = P(H) + (3n^2 - n)/2$.

For $m \geq 3$, we first show that $P(G) \leq 2P(H) + mn(3n - 1)/2 - 2n^2 + n$. Assume that $g$ is a profile numbering of $H$. Consider a numbering $f$ such that

$$f(v_{ij}) = \begin{cases} g(v_j) + (i - 1)n & \text{for } 1 \leq i \leq m - 2, 1 \leq j \leq n \\ g(v_j) + (m - 1)n & \text{for } i = m - 1, 1 \leq j \leq n \\ g(v_j) + (m - 2)n & \text{for } i = m, 1 \leq j \leq n. \end{cases}$$

Then

$$P_f(G) = P(H) + (m - 3) \sum_{i=0}^{n-1} (n + i) + P(H) + \sum_{i=0}^{n-1} (2n + i)$$

$$= 2P(H) + \frac{mn(3n - 1)}{2} - 2n^2 + n.$$ 

Let $h$ be a profile numbering of $G$. Now, assume that $P_h(G) < 2P(H) + mn(3n - 1)/2 - 2n^2 + n$. For the same reason as in the case with $m = 2$, $h$ must completely number a row before starting another row. Furthermore, $h$ must begin with one
of the end rows (say \( R_1 \)); otherwise \( P_h(G) \geq P_f(G) + n \). We claim that \( h \) must number the rows in the order \( R_1, R_2, \ldots, R_{m-2}, R_m, R_{m-1} \).

We prove the claim by contradiction. Assume that this pattern is not followed. So the first violation by \( h \) is at \( R_p \) to \( R_q \), where \( q = p + r \) and \( r > 1 \) and \( p < m - 2 \). Then if \( q < m \),

\[
\sum_{i=p+1}^{q+1} \sum_{j=1}^{n} w_h(v_{i,j}) \geq P(H) + (5n^2 - n) + \frac{(r - 2)(3n^2 - n)}{2}
\]

\[
> \frac{(r + 1)(3n^2 - n)}{2}
\]

\[
= \sum_{i=p+1}^{q+1} \sum_{j=1}^{n} w_f(v_{i,j}).
\]

If \( q = m \), then

\[
\sum_{i=p+1}^{q} \sum_{j=1}^{n} w_h(v_{i,j}) \geq P(H) + (5n^2 - n) + \frac{(r - 3)(3n^2 - n)}{2}
\]

\[
> P(H) + \frac{5n^2 - n}{2} + \frac{(r - 2)(3n^2 - n)}{2}
\]

\[
= \sum_{i=p+1}^{q} \sum_{j=1}^{n} w_f(v_{i,j}).
\]

So, the claim is proved and \( h \) must number the rows in the same order as \( f \).

Since within each row, \( f \) numbers the same way as \( g \), which is a profile numbering of \( H \), it follows that
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} w_f(v_{i,j}) = 2P(H) + \frac{mn(3n-1)}{2} - 2n^2 + n
\]

\[
< \sum_{j=1}^{n} w_h(v_{1,j}) + \sum_{j=1}^{n} w_h(v_{m,j}) + \frac{mn(3n-1)}{2} - 2n^2 + n
\]

\[
= P_h(G),
\]

which implies that \( P_f(G) < P_h(G) \), which is a contradiction. \( \square \)

**Corollary 1** For \( m \geq 3 \),

\[
P(G) = \begin{cases} 
\frac{mn(3n-1)}{2} - 2n^2 + 3n - 2 & \text{for } G = P_m[P_n] \\
\frac{mn(3n-1)}{2} - 2n^2 + 5n - 6 & \text{for } G = P_m[C_n] \\
\frac{mn(3n-1)}{2} - n^2 & \text{for } G = P_m[K_n].
\end{cases}
\]

### 4.3 Cycles and Complete Graphs With Other Graphs

In this section, we establish the profile of the composition of cycles with other graphs and the profile of the composition of complete graphs with other graphs.
**Theorem 6** Let $G = C_m[H]$ where $H$ is a graph with $n$ vertices. Then

$$P(G) = P(H) + \frac{n(5mn - 7n - m + 1)}{2}.$$  

Proof: Let $g$ be a profile numbering of $H$. Define a numbering $f(v_{i,j}) = (i-1)n + g(v_j)$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Then using an argument similar to that in the proof of the previous theorem, we see that a profile numbering is produced, namely,

$$P_f(G) = P(H) + \frac{(m-2)n(3n-1)}{2} + \frac{n(2mn - n - 1)}{2}$$

$$= P(H) + \frac{n(5mn - 7n - m + 1)}{2}.$$

\[\square\]

**Corollary 2** For $m \geq 3$ and $n \geq 3$,

$$P(G) = \begin{cases} 
\frac{n(5mn - 7n - m + 3)}{2} - 1 & \text{for } G = C_m[P_n] \\
\frac{n(5mn - 7n - m + 5)}{2} - 3 & \text{for } G = C_m[C_n] \\
\frac{n(5mn - 6n - m)}{2} & \text{for } G = C_m[K_n]. 
\end{cases}$$
Theorem 7 For $m \geq 1$, let $G = K_m[H]$ where $H$ is a graph with $n$ vertices. Then

$$P(G) = P(H) + \frac{(mn + n - 1)(mn - n)}{2}.$$ 

Proof: In the composition of a complete graph with another graph, every vertex in $R_i$ is adjacent to all other vertices which are in rows other than $R_i$. Hence once we number a vertex 1, we know

$$\sum_{i=n+1}^{mn} w_f(v_i) = \frac{(mn + n - 1)(mn - n)}{2}.$$ 

Since $\min \sum_{i=1}^{n} w_f(v_i) = P(H)$, we then have $P(G) \geq P(H) + (mn + n - 1)(mn - n)/2$. 

And $P(H) + (mn + n - 1)(mn - n)/2$ is achievable by numbering $R_1$ with 1, \ldots, $n$; $R_2$ with $n + 1, \ldots, 2n$ etc. within each row follow a profile numbering of $H$. So, 

$$P(G) = P(H) + (mn + n - 1)(mn - n)/2.$$ 

Corollary 3 For $m \geq 1$ and $n \geq 1$,

$$P(K_m[K_n]) = \frac{mn(mn - 1)}{2}.$$ 

4.4 Complete Bipartite Graphs With Other Graphs

In this section, we present the profile of the composition of a complete bipartite graph with any arbitrary graph of order $l$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Theorem 8 Let $G = K_{m,n}[H]$ where $m \leq n$ and $H$ is a graph of order $l$. Then

$$P(G) = mnl^2 + \frac{ml(ml - 1)}{2} + nP(H).$$

Proof: Assume the two partite sets of vertices in $K_{m,n}$ are $\{v_1, v_2, \ldots, v_n\}$ and $\{v_{n+1}, v_{n+2}, \ldots, v_{n+m}\}$. Also assume that $g$ is a profile numbering of $H$. Consider a numbering $f$ such that $f(v_{ij}) = g(v_j) + (i - 1)l$ for $1 \leq i \leq n + m$. Then

$$P_f(G) = nP(H) + \sum_{i=nl+1}^{(m+n)l} (i - 1) = mnl^2 + \frac{ml(ml - 1)}{2} + nP(H).$$

Now assume that $h$ is a profile numbering of $G$. For $1 \leq i \leq n$ and $n + 1 \leq j \leq n + m$, every vertex in $\bigcup_{i=1}^{n} R_i$ is adjacent to every vertex in $\bigcup_{j=n+1}^{n+m} R_j$ and for $m \leq n$, the vertex $v = h^{-1}(1)$ should be in $\bigcup_{i=1}^{n} R_i$. So

$$\sum_{i=n+1}^{m+n} \sum_{j=1}^{l} w_h(v_{ij}) = mnl^2 + \frac{ml(ml - 1)}{2}.$$

Since $\min \sum_{i=1}^{n} \sum_{j=1}^{l} w_h(v_{ij}) = nP(H)$, we have $P_h(G) = mnl^2 + ml(ml - 1)/2 + nP(H)$. □

A direct application of Theorem 8 leads to Corollary 4.
Corollary 4

\[
P(G) = \begin{cases} 
  mnl^2 + nl - n + \frac{ml(ml - 1)}{2} & \text{for } G = K_{m,n}[P_l] \\
  mnl^2 + 2nl - 3n + \frac{ml(ml - 1)}{2} & \text{for } G = K_{m,n}[C_l] \\
  mnl^2 + \frac{nl(l - 1)}{2} + \frac{ml(ml - 1)}{2} & \text{for } G = K_{m,n}[K_l].
\end{cases}
\]

The profile of the composition of a star with any other graph also follows directly from Theorem 8.

Corollary 5 For \( n \geq 1 \), let \( H \) be a graph with \( l \) vertices. Then

\[
P(K_{1,n}[H]) = nl^2 + \frac{l(l - 1)}{2} + nP(H).
\]

Corollary 6 For \( n \geq 1 \) and \( l \geq 1 \),

\[
P(G) = \begin{cases} 
  nl^2 + nl - n + \frac{l(l - 1)}{2} & \text{for } G = K_{1,n}[P_l] \\
  nl^2 + 2nl - 3n + \frac{l(l - 1)}{2} & \text{for } G = K_{1,n}[C_l] \\
  nl^2 + \frac{l(l - 1)(n + 1)}{2} & \text{for } G = K_{1,n}[K_l].
\end{cases}
\]
CHAPTER V

NETWORK ARCHITECTURES

5.1 Introduction

Linear arrays (paths), rings (cycles), completely connected ($K_n$), binary trees, stars ($K_{1,n}$), 2-dimensional meshes ($P_n \times P_m$), 2-dimensional tori ($C_m \times C_n$), hypercubes and butterflies are some of the most commonly used network architectures. It is known that for the linear arrays $B(P_n) = 1$, $s(P_n) = n - 1$ and $P(P_n) = n - 1$. For rings $B(C_n) = 2$, $s(C_n) = 2n - 2$ and $P(C_n) = 2n - 3$. For completely connected graphs $B(K_n) = n - 1$, $s(K_n) = n(n - 1)/2$ and $P(K_n) = n(n - 1)/2$. For binary trees the bandwidth problem is NP-complete. Chung[14] gives a solution for the edgesum of a complete binary tree. There are several polynomial algorithms to find the edgesum for arbitrary trees (see [13], [28] and [71]). In fact, Chung[13] provided an $O(n^{1.6})$ algorithm, which is the most efficient known, to achieve optimal numberings for arbitrary trees. Also, an $O(n^{1.72})$ algorithm which gives the profile for arbitrary trees was provided by Chang[42]. For stars, $B(K_{1,n}) = \lfloor n/2 \rfloor$, $s(K_{1,n}) = (\lfloor n/2 \rfloor (\lfloor n/2 \rfloor + 1) + \lfloor n/2 \rfloor (\lfloor n/2 \rfloor + 1))/2$ and $P(K_{1,n}) = n$. For 2-dimensional meshes, with $m \leq n$, $B(P_m \times P_n) = m$. Mitchison and Durbin[65] provides a polynomial algorithm to solve the edgesum
problem of a 2-dimensional mesh and Lin and Yuan[54] gives \( P(P_m \times P_n) = m^2n - m(2m^2 - 3m + 7)/6 \). For the 2-dimensional torus,

\[
B(C_m \times C_n) = \begin{cases} 
2m & \text{for } 3 \leq m \leq n \\
2m - 1 & \text{for } 3 \leq m = n.
\end{cases}
\]

The same numbering given by Mitchison and Durbin[65] for the 2-dimensional mesh also achieves the edgesum of the 2-dimensional torus. Furthermore, Mai[61] provides the profile \( P(C_m \times C_n) = [2m - 2n/3 + 1/2)n^2 - 16n/3 + 3 - \min\{1, m - n\} \) with \( m \geq n \geq 3 \). There is a polynomial algorithm for the bandwidth of a hypercube given by Harper[33] and there is a polynomial algorithm for the edgesum of a hypercube (see Harper[34]), but there is no work done for the profile of a hypercube. Also, there is no work done on butterfly architectures.

In this chapter, we investigate the bandwidth, edgesum and profile of butterflies and the profile of hypercubes in an attempt to complete the work on the most commonly used network architectures.

5.2 Butterfly Architecture

A \( d \)-dimensional butterfly has \((d + 1)2^d\) vertices and \(d2^{d+1}\) edges. The vertices correspond to pairs \((l, w)\) where \(l\) is the level or dimension of the vertex \((0 \leq l \leq d)\) and \(w\) is an \(d\)-bit binary number that denotes the position (column) of the vertex. Two vertices \((l_1, w_1)\) and \((l_2, w_2)\) are adjacent if and only if \(l_1 = l_2 + 1\)
and either (1) \( w_1 = w_2 \) or (2) \( w_1 \) and \( w_2 \) differ in only the 1st bit. If \( w_1 = w_2 \), the edge is said to be a straight edge. Otherwise, the edge is called a cross edge.

For example, Figure 13 shows a 2-dimensional butterfly.

In this section, we discuss the bandwidth, edgesum and profile of the butterfly architectures.

For \( S \subseteq V(G) \), \( \partial S \) denotes the set of vertices in \( S \) adjacent to those in \( V(G) - S \) and \( N(S) \) denotes the set of vertices in \( V(G) - S \) adjacent to those in \( S \). Let \( p(G) \) denote \( |V(G)| \) and \( \Delta(G) \) be the diameter of \( G \). For \( S \subseteq V \), \( S' \) denotes \( V(G) - S \). For a given numbering \( f \), let \( S_t = \{v \in V(G) : f(v) \leq t \} \). The following propositions are used in the proofs that follow.

**Proposition 3** (From Harper[33]) If \( G \) is a connected graph, then

\[
B(G) \geq \max_k \min_{|S|=k} |\partial S|.
\]

**Proposition 4** (From Lai and Williams[46]) Let \( f \) be a bandwidth numbering of

![Figure 13. A 2-dimensional Butterfly.](image)
G. Then for \( t \in \{1, 2, \ldots, p(G)\} \), \( |\partial f S_t| \leq B(G) \).

**Proposition 5** (From Lai and Williams[46]) Let \( f \) be a bandwidth numbering of \( G \). Then for \( t \in \{1, 2, \ldots, p(G)\} \), \( |N_f(S_t)| = |\partial f S_t| \leq B(G) \).

Given a graph \( G = (V, E) \), define a cut of \( G \) as a subset of \( E \) which when removed from \( E \) disconnects \( G \) into two connected subgraphs \( H_1 \) and \( H_2 \). Given a cut \( R \) of \( G \), we define \( t(R) = \{ v \in V : v \) is an endvertex of an edge in \( R \} \).

**Lemma 3** Let \( G = (V, E) \) be a graph with \( n \) vertices. If every cut \( R \) of \( G \) that disconnects \( G \) into two subgraphs \( H_1 \) and \( H_2 \) for which \( |V(H_1)| = \lfloor n/2 \rfloor \) and \( |V(H_2)| = \lfloor n/2 \rfloor \) has the property that \( |t(R)| \geq m \), then \( B(G) \geq m/2 \).

Proof: Let \( f \) be a bandwidth numbering of \( G \) and let \( V(H_1) = S_{\lfloor n/2 \rfloor} \). For every cut \( R \) of \( G \), we have \( |t(R)| \geq m \), and \( |t(R)| = |N_f(S_{\lfloor n/2 \rfloor})| + |\partial f(S_{\lfloor n/2 \rfloor})| \geq m \). So either \( |N_f(S_{\lfloor n/2 \rfloor})| \geq m/2 \) or \( |\partial f(S_{\lfloor n/2 \rfloor})| \geq m/2 \). By Propositions 4 and 5 we have \( B(G) = B_f(G) \geq m/2 \).

**Theorem 9** If \( G \) is a \( d \)-dimensional butterfly, then \( B(G) = 2^d \).

Proof: By [44] we know that the bisection width of a \( d \)-dimensional butterfly is \( 2^d \). In fact, every cut \( R \) of a \( d \)-dimensional butterfly which disconnects \( G \) into two subgraphs of equal order has the property that \( |t(R)| \geq 2^{d+1} \). Then by Lemma 3, we have \( B(G) \geq 2^d \).

We number the left-most half of the graph sequentially row by row left-to-right until the last row where we number the whole row. Then we reverse the
order in the second right-most half of the graph. This numbering will achieve the bandwidth $2^d$ (see Figure 14). So $B(G) = 2^d$. □

**Proposition 6** *(From Harper [34]*) If $f$ is a proper numbering of a graph $G$ on $n$ vertices, then

$$
\sum_{u \in G} |f(u) - f(v)| = \sum_{t=1}^{n} e(S_t),
$$

where $e(S_t)$ is the number of edges with one endvertex in $S_t$ and the other endvertex in $V(G) - S_t$.

The following algorithm appears to provide an edgesum numbering for butterflies of dimension $d$ and order $n$.

**Algorithm 1** *Edgesum of butterflies.*

Input: A $d$-dimensional butterfly of $n = (d + 1)2^d$ vertices named 1, 2, ..., $n$ in a consecutive order from level 0, row by row.

![Figure 14. A Bandwidth Numbering of 3-dimensional Butterfly.](image)
Output: A believed edgesum numbering of the given graph.

Variables:

\( V \) is an array which holds the numbering. The index in \( V \) indicates the name of the vertex.

\( NbList \) is an array which holds \( N(S_t) \) where \( S_t \) is the set of vertices that have been numbered \( 1, 2, \ldots, t \).

\( cur \) is the lowest number which is not yet assigned.

Methods:

1. Initialize \( V[i] \) to zero for \( 1 \leq i \leq n \).
3. Put \( N(S_{cur-1}) \) into \( NbList \).
4. In \( NbList \), find a vertex \( j \) such that \( V[j] = cur \) will give a minimum value of \( e(S_{cur}) \).
5. Assign \( cur \) to \( V[j] \) and increase \( cur \) by one.
6. Loop from step 3 to step 5 until all the vertices have been numbered.

End of Algorithm 1

To analyze the time complexity of Algorithm 1 we note that step 1 takes \( O(n) \), step 2 takes \( O(1) \) and step 3 takes no more than \( O(n^2) \), step 4 takes no more than \( O(n^3) \), step 5 takes \( O(1) \) and Step 3,4,5 loop \( O(n) \) times, so the total time complexity for Algorithm 1 is no more than \( O(n^4) \) which, of course, is polynomial.

A version of the implementation of Algorithm 1 in C++ code is given in Appendix.
Figure 15. Algorithm 1 Applied to a 3-dimensional Butterfly, $s_f(G) = 202$.

A. Figure 15 shows the numbering of Algorithm 1 applied to a 3-dimensional butterfly.

Through exhaustive computer testing it has been established that Algorithm 1 provides an edgesum numbering for all butterflies with $n \leq 12$. We have tried, by computer testing, a number of reasonable techniques for $32 \leq n \leq 80$, and this algorithm continues to provide the best numbering. Therefore it is believed that this algorithm will provide an edgesum numbering for all butterflies.

The following algorithm appears to provide a profile numbering for butterflies of dimension $d$ and order $n$.

**Algorithm 2** Profile of butterflies.

Input: A $d$-dimensional butterfly of $n = (d + 1)2^d$ vertices named $1, 2, \ldots, n$ in a consecutive order from level 0, row by row.

Output: A believed profile numbering of the given graph.
Variables:

$V$ is an array which holds the numbering. The index in $V$ indicates the name of the vertex.

cur is the lowest number which is not yet assigned.

Methods:

1. Initialize $V[i]$ to zero for $1 \leq i \leq n$.


3. Among all vertices which have not been numbered, find a vertex $j$ such that setting $V[j] = cur$ will result in a minimum value of $|N(S_{cur})|$.

4. Assign cur to $V[j]$ and increase cur by 1.

5. Loop from step 3 to step 4 until all the vertices have been numbered.

End of Algorithm 2

The time complexity analysis of Algorithm 2 shows that step 1 takes $O(n)$ time, step 2 takes $O(1)$, step 3 takes no more than $O(n^3)$, and step 4 takes $O(1)$. Steps 3 and 4 loop $O(n)$ times, so the total time complexity for Algorithm 1 is no more than $O(n^4)$ which is polynomial. A version of the implementation of Algorithm 2 in C++ code is in Appendix B. Figure 16 shows the numbering of Algorithm 2 applied to a 3-dimensional butterfly.

Through exhaustive computer testing it has been established that Algorithm 2 provides a profile numbering for all butterflies with $n \leq 12$. Again, we have tested, by computer, a number of reasonable techniques for $32 \leq n \leq 80$, and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 16. Algorithm 2 Applied to a 3-dimensional Butterfly, $P_f(G) = 125$.

Figure 17. A 4-dimensional Hypercube.

This algorithm still provides the best numbering. It is believed that this algorithm will provide a profile numbering for all butterflies.

5.3 Profiles of Hypercubes

Recall that a $d$-dimensional hypercube has $n = 2^d$ vertices and $d2^{d-1}$ edges. Each vertex corresponds to a $d$-bit binary number, and two vertices are adjacent if and only if their binary number differs in only one bit. Figure 17 shows a 4-dimensional hypercube.
Polynomial numbering algorithms to provide the bandwidth and edgesum of the hypercube are known. For the profile of a hypercube, the following algorithm provides a numbering for a hypercube of order $n$ which has dimension $d = \log n$. Through exhaustive computer testing, we have established that this algorithm provides a profile numbering for all hypercubes with $d \leq 4$. It is believed that it will provide a profile numbering for all hypercubes.

**Algorithm 3** *Profile of hypercubes.*

Input: A hypercube with $n$ vertices named $0, 1, 2, \ldots, n - 1$ according to the natural numbering which uses the decimal equivalent of the binary number.

Output: A numbering which provides a low profile.

Variables:

- $V$ is an array which holds the numbering. The index in $V$ indicates the name of the vertex.
- $cur$ is the lowest number which is not yet assigned.
- $pre$ is the lowest number of a vertex whose neighbors are not all assigned a number yet.

Methods:

1. Initialize $V[i]$ to zero for $1 \leq i \leq n$.
2. Set $V[0] = 1$ and $cur = 2$.
3. Set $pre = 0$.
4. Number all unnumbered vertices which are adjacent to $pre$ as $cur$ and
Figure 18. Algorithm 3 Applied to a 4-dimensional Hypercube, $P_f(G) = 75$.

increase cur by 1 after numbering each vertex.

5. Increase pre by 1.

6. Repeat step 4 and 5 until all the vertices are numbered.

End of Algorithm 3

The time complexity analysis of Algorithm 3 shows that step 1 takes $O(n)$, step 2 to step 3 takes $O(1)$ time, step 4 takes $O(n)$ and step 5 takes $O(1)$ and steps 4 and 5 loop $O(n)$ times, so the total time complexity for Algorithm 3 is $O(n^2)$ which is polynomial. Figure 18 shows the numbering of Algorithm 3 applied to a 4-dimensional hypercube.

Since none of Algorithms 1, 2 and 3 have been mathematically proved optimal except for graphs of small order, they must be regarded as providing approximation numberings. Also, their result provides an upper bound on the exact edgesum and profile.
CHAPTER VI

PROFILE OF CORONA

6.1 Definition and Examples

The corona of two graphs was first defined by Harary [32].

**Definition 4** Given graphs $G_1$ and $G_2$ on $n_1$ and $n_2$ vertices respectively, the corona $G = G_1 \ast G_2$ of $G_1$ with respect to $G_2$ has $V(G) = V(G_1) \cup \{n_1 \text{ distinct copies of } V(G_2)\}_1, V(G_2)_2, \ldots, V(G_2)_n\}$, and $E(G) = E(G_1) \cup \{n_1 \text{ distinct copies of } E(G_2)\} \cup \{(u_i, v) : u_i \in V(G_1), v \in V(G_2)_i\}$.

Figure 19 shows $P_3 \ast P_4$. Chinn, Lin and Yuan[11] determined a tight upper bound for the bandwidth of the corona of two graphs and gave solutions for special cases. Williams[85] established a tight upper bound and a tight lower bound for the edgesum of the corona of two graphs and gave solutions for some special cases. There is no work done for the profile of the corona of two graphs. This chapter determine a tight upper bound and a tight lower bound for the profile of the corona of two graphs. Also, the exact values are determined for the profile

![Figure 19. $P_3 \ast P_4$.](image)
of the corona of several families of graphs.

6.2 Tight Bounds for General Cases

In this section we give an upper bound and a lower bound for the profile of the corona of any two graphs and show some special cases to achieve the upper bound and the lower bound.

Recall by Proposition 2 that for a graph $G$ of order $n$ and proper numbering $f$ of $G$, $P_f(G) = \sum_{i=1}^{n} |N_f(S_i)|$, where $S_i = \{u : f(u) \leq i\}$.

Lemma 4 Given a graph $G = (V(G), E(G))$, let $H = (V(H), E(H))$ such that $V(H) \cap V(G) = \emptyset$ and $u \in V(G)$. Let $G' = (V', E')$ where $V' = V(G) \cup V(H)$ and $E' = E(G) \cup E(H) \cup \{uv : v \in V(H)\}$. Then there exists a profile numbering $f$ of $G'$ such that for each $v \in V(H)$, $f(v) < f(u)$.

Proof: Let $g$ be a profile numbering of $G'$. Suppose that there is $v \in V(H)$ such that $g(v) > g(u)$. Let $a = g(u)$ and $b = \max\{g(v) : v \in H\}$. Define a proper numbering $f$ of $G$ as follows:

$$f(x) = \begin{cases} b & x = u \\ g(x) - 1 & a < g(x) \leq b \\ g(x) & g(x) < a \text{ or } g(x) > b. \end{cases}$$

Then for $t < a$ or $t > b$, $|N_f(S_t)| = |N_g(S_t)|$, $\sum_{t=a}^{b} |N_f(S_t) \cap V(G)| \leq \sum_{t=a}^{b} |N_g(S_t) \cap V(G)| + c$ where $c = |\{x : x \in V(G), g(x) < b \text{ and } xv \notin E(G)\}|$ and $\sum_{t=a}^{b} |N_f(S_t) \cap V(G)|$.
\[ V(H) < \sum_{t=1}^{k} |N_{g}(S_t) \cap V(H)| \cdot c' \text{ where } c' = |\{v : v \in V(H) \text{ and } v \notin N_{g}(S_{k-1})\}|. \] Since \( g \) is a profile numbering, if \( c > 0 \) then \( c' > 0 \). So \( \sum_{t=1}^{n(m+1)} |N_{f}(S_t)| \leq \sum_{t=1}^{n(m+1)} |N_{g}(S_t)|. \) Thus, \( f \) must be a profile numbering of \( G' \) and for each \( v \in V(H), f(v) < f(u). \)

**Lemma 5** Given a graph \( G = (V(G), E(G)) \), let \( H = (V(H), E(H)) \) such that \( V(H) \cap V(G) = \emptyset \) and \( u \in V(G) \). Let \( G' = (V', E') \) where \( V' = V(G) \cup V(H) \) and \( E' = E(G) \cup E(H) \cup \{uv : v \in V(H)\} \). Then \( P(G') \geq P(G) + P(H) + |V(H)| \).

**Proof:** Let \( f \) be a profile numbering of \( G' \) which satisfies Lemma 4. Define a proper numbering \( h \) of \( G \) as follows:

\[
h(x) = \begin{cases} f(x) & f(x) < \min \{f(v) : v \in V(H)\} \\ f(x) - k & k = |\{v : f(x) > f(v) \text{ and } v \in V(H)\}|. \end{cases}
\]

Since \( f(u) > f(v) \) for each \( v \in V(H) \), we know that \( \sum_{x \in V(H)} w_{f}(u) \geq P(H) \) and \( w_{f}(u) \geq w_{h}(u) + |V(H)|. \) So, \( \sum_{x \in V(G)} w_{f}(x) = \sum_{x \in V(G)} w_{f}(x) + \sum_{x \in V(H)} w_{f}(x) \geq \sum_{x \in V(G)} w_{h}(x) + |V(H)| + P(H) \geq P(G) + P(H) + |V(H)| \)

which implies that \( P(G') \geq P(G) + P(H) + |V(H)|. \)

Applying Lemma 5 \( n_{1} \) times, we have Theorem 10.

**Theorem 10** Let \( G_{1} \) and \( G_{2} \) be two graphs of orders \( n_{1} \) and \( n_{2} \) respectively. Then

\[
P(G_{1} \land G_{2}) \geq P(G_{1}) + n_{1}P(G_{2}) + n_{1}n_{2}.
\]
The bound in Theorem 10 is tight as illustrated later in Theorem 11.

Lemma 6 is immediate since $G$ is connected.

**Lemma 6** If $G$ is a connected graph of order $n$ and $f$ is a proper numbering on $G$, then $|N_f(S_t)| \geq 1$ for $1 \leq t < n$.

**Theorem 11** If $G$ is a graph of order $m$, then $P(P_n \Delta G) = nP(G) + mn + n - 1$.

Proof: Let the vertices in $P_n$ be named sequentially, from one end to the other, $v_i$ for $1 \leq i \leq n$. Let $f$ be a profile numbering of $G$. Define a proper numbering $g$ of $P_n \Delta G$ as follows:

$$g(x) = \begin{cases} (m+1)i & x = v_i \text{ and } 1 \leq i \leq n \\ (i-1)(m+1) + f(x) & x \in V(G_i) \text{ and } 1 \leq i \leq n. \end{cases}$$

Then $P_g(P_n \Delta G) = nP(G) + (m+1)(n-1) + m = nP(G) + mn + n - 1$. So,

$P(P_n \Delta G) \leq nP(G) + mn + n - 1$.

Suppose that $P(P_n \Delta G) < nP(G) + mn + n - 1$. Let $h$ be a profile numbering of $P_n \Delta G$. We note that

$$|N_g(S_t)| = \begin{cases} 0 & t = (m+1)n \\ 1 & t \equiv 0 \pmod{m+1} \text{ and } t \neq (m+1)n \\ |N_f(S_t)| + 1 & \text{otherwise} \end{cases}$$

By Proposition 2, there exists an integer $t$ with $1 \leq t < (m+1)n$ such that $|N_h(S_t)| < |N_g(S_t)|$. By Lemma 6, $t \not\equiv 0 \pmod{m+1}$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
For either \( h^{-1}(t) \in V(G_i) \) or \( h^{-1}(t) \in V(P_n) \), we have \(|N_h(S_t)| \geq |N_f(S_t)| + 1 = |N_g(S_t)|\) which produces a contradiction. Thus, \( P(P_n \wedge G) = nP(G) + mn + n - 1 \).

**Theorem 12** If \( G_1 \) and \( G_2 \) are two graphs of orders \( n_1 \) and \( n_2 \) respectively, then

\[
P(G_1 \wedge G_2) \leq n_1P(G_2) + (n_2 + 1)P(G_1) + n_1n_2.
\]

Proof: Let \( f_1 \) be a profile numbering of \( G_1 \) and let \( v_i \) denote the vertex in \( G_1 \) with \( f_1(v_i) = i \) for \( 1 \leq i \leq n_1 \). Let \( f_2 \) be a profile numbering of \( G_2 \) and let \( u_j \) denote the vertex in \( G_2 \) with \( f_2(u_j) = j \) for \( 1 \leq j \leq n_2 \). Define a proper numbering \( g \) of \( H = G_1 \wedge G_2 \) as follows:

\[
g(x) = \begin{cases} 
(n_2 + 1)f_1(x) & x \in V(G_1) \\
(n_2 + 1)(f_1(v_i) - 1) + f_2(x) & x \in V(G_2) \text{ and } (x, v_i) \in E(H)
\end{cases}
\]

Then \( P_g(G_1 \wedge G_2) = n_1P(G_2) + (n_2 + 1)P(G_1) + n_1n_2 \). Thus, \( P(G_1 \wedge G_2) \leq n_1P(G_2) + (n_2 + 1)P(G_1) + n_1n_2 \). \( \square \)

The bound in Theorem 12 is tight as illustrated later in Theorem 13.

**Definition 5** The complement of a graph \( G = (V, E) \) is denoted as \( \overline{G} = (V, \overline{E}) \) where \( \overline{E} = \{uv : uv \notin E\} \).

**Theorem 13** Let \( G = \overline{K_n} \wedge K_m \). Then \( P(G) = nm(m + 1)/2 \).
Proof: Note that \( P(K_n) = 0 \) and \( P(K_m) = m(m - 1)/2 \). By Theorem 12, 
\[ P(G) \leq nP(K_m) + mP(K_n) + mn = nm(m - 1)/2 + mn = nm(m + 1)/2. \]
Since \( K_n \land K_m \) is \( n \) separate copies of \( K_{m+1} \), \( P(G) = nm(m + 1)/2. \)

\[ \square \]

6.3 Tight Bounds on the Corona of Graphs With \( K_m \)

A tight upper bound and a tight lower bound is given in this section for the corona of a connected graph with \( K_m \).

Given \( H = K_m \) and applying Lemma 5 \( n \) times, we have Theorem 14.

**Theorem 14** If \( G \) is a graph of order \( n \), then \( P(G \land K_m) \geq P(G) + mn. \)

Theorem 15 shows that the bound in Theorem 14 is tight.

**Theorem 15** If \( G = P_n \land K_m \), then \( P(G) = mn + n - 1. \)

Proof: Let the vertices in \( P_n \) be named sequentially, from one end to the other, \( v_i \) for \( 1 \leq i \leq n \). Also, \( u_{ij} \) for \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \) will be a vertex in \( K_m \) which is joined to \( v_i \). Let \( f(v_i) = (m + 1)i \) and \( f(u_{ij}) = (m + 1)i - j. \)
Then \( P_f(G) = (m + 1)(n - 1) + m = mn + n - 1. \) So, \( P(G) \leq mn + n - 1. \) By Theorem 10, we have \( P(G) \geq P(P_n) + mn = (n - 1) + mn = mn + n - 1. \) Hence, \( P(G) = mn + n - 1. \)

\[ \square \]

**Corollary 7** If \( G = P_n \land K_1 \), then \( P(G) = 2n - 1. \)

**Lemma 7** For any connected graph \( G \) of order \( n \geq 3 \), there exists a profile numbering \( f \) of \( G \) such that at least two vertices of \( G \), say \( u \) and \( v \), have \( w_f(u) > 0 \)
and $w_f(v) > 0$.

Proof: Let $g$ be a profile numbering of $G$. Let the vertices of $G$ be named $v_1, v_2, \ldots, v_n$ where $g(v_i) = i$. Then we know that $w_g(v_n) > 0$. If there exists a vertex $v_i$ for $1 \leq i < n$ such that $w_g(v_i) > 0$, then the lemma follows. So we assume that $w_g(v_i) = 0$ for $1 \leq i < n$. Then both of $v_1$ and $v_{n-1}$ must only be adjacent to $v_n$. Define $f$ as follows:

$$f(v_i) = \begin{cases} i & 1 \leq i \leq n-2 \\ n & i = n-1 \\ n-1 & i = n \end{cases}$$

Then $\sum_{i=1}^n w_f(v_i) = \sum_{i=1}^n w_g(v_i)$. So $f$ is a profile numbering of $G$ and $w_f(v_{n-1}) = w_g(v_n) - 1 > 0$ and $w_f(v_n) = 1 > 0$. $\square$

**Theorem 16** If $G = (V, E)$ is a connected graph of order $n$ for $n \geq 3$, then

$$P(G \land \overline{K_m}) \leq (m+1)P(G) + m(n-2).$$

Proof: For each profile numbering $h$ of $G$, define $W_h = \{v : w_h(v) > 0\}$. Let $f$ be the profile numbering such that $|W_f| = \max |W_h|$ where the maximum is taken over all profile numberings of $G$. Let the vertices of $G$ be named $v_1, v_2, \ldots, v_n$ where $f(v_i) = i$. Let the vertices of the $i$th copy of $\overline{K_m}$ in $G \land \overline{K_m}$ be denoted by $u_{ij}$ for $1 \leq j \leq m$. Then define a proper numbering $g$ of $H = G \land \overline{K_m}$ as $g(v_i) = (m+1)i$ and $g(u_{ij}) = (m+1)(i-1) + j$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Then \( P_g(G \wedge \overline{K_m}) = (m+1)P(G) + m(n - |W_f|) \). By Lemma 7, we know that \(|W_f| \geq 2\) so the theorem follows. \(\square\)

**Corollary 8** If \( G \) is a connected graph of order \( n \) for \( n \geq 3 \), then \( P(G \wedge K_1) \leq 2P(G) + n - 2 \).

The bound of Theorem 16 is tight as is illustrated in Theorem 17.

**Theorem 17** If \( G = K_{1,n} \wedge \overline{K_m} \), then \( P(G) = 2mn - m + n \).

Proof: Note that the order of \( K_{1,n} \) is \( n + 1 \) and \( P(K_{1,n}) = n \). By Theorem 16, 
\[ P(G) \leq (m+1)P(K_{1,n}) + m((n+1) - 2) = (m+1)n + m(n+1-2) = 2mn - m + n. \]

Let the vertices in \( K_{1,n} \) be denoted as \( v_1, v_2, \ldots, v_n \) and \( v_{n+1} \) where \( \text{deg}(v_i) = 1 \) for \( 1 \leq i \leq n \). Name the vertices in \( \overline{K_m} \) joined to \( v_i \) as \( u_{ij} \) for \( 1 \leq i \leq n + 1 \) and \( 1 \leq j \leq m \). Define \( f \) as follows:

\[
f(v_i) = \begin{cases} 
(m+1)i & 1 \leq i \leq n-1 \\
(m+1)(n+1) & i = n \\
(m+1)n & i = n + 1 
\end{cases}
\]

and \( f(u_{ij}) = f(v_i) - j \) for \( 1 \leq j \leq m \) and \( 1 \leq i \leq n + 1 \). Then \( P_f(G) = m(n-1) + m + 1 + (n-1)(m+1) = 2mn - m + n. \)

Now suppose that \( P(G) < 2mn - m + n \). Let \( g \) be a profile numbering of \( G \). By Proposition 2, there exists an integer \( t \) with \( 1 \leq t < 2n + 2 \) such that \( |N_g(S_t)| < |N_f(S_t)| \). Note that
\[ |N(S_t)| = \begin{cases} 
0 & i = mn + n \\
1 & 1 \leq i \leq m \\
1 & i > (n-1)(m+1) - 1 \\
1 & i \equiv 0 \pmod{m+1} \text{ and } i \neq mn + n \\
2 & i \not\equiv 0 \pmod{m+1} \text{ and } m < i < (n-1)(m+1). 
\end{cases} \]

By Lemma 6, \( t \) must satisfy following conditions: \( t \not\equiv 0 \pmod{m+1}, m < t < (n-1)(m+1) \) and \( |N_g(S_t)| = 1 \). However, for any \( |N_g(S_t)| < |N_f(S_t)| \), we must have \( |N_g(S_{t+i})| > |N_f(S_{t+i})| \) for some \( i \) such that \( 1 \leq i \leq m \). Since we have \( t \not\equiv 0 \pmod{m+1} \) and \( m < t < (n-1)(m+1) \), then \( g^{-1}(t) = u_{(n+1)j} \) for some integer \( j \) such that \( 1 \leq j \leq m \) and \( N_g(S_t) = \{v_{n+1}\} \). Consider the set \( X \) of vertices such that for all \( x \in X \), we have \( t < g(x) \leq t + m \). If all such elements of \( X \) are in copies of \( \overline{K_m} \), then there is at least one of them, say \( y \), such that \( |N_g(S_{g(y)})| = 2 \). If there is a vertex \( v_i \) such that \( t+1 \leq g(v_i) \leq t + m \), then \( |N_g(S_{g(v_i)})| \geq 2 \). So \( \sum_{i=1}^{m+1} |N_g(S_t)| \geq \sum_{i=1}^{m+1} |N_f(S_t)| \). Therefore, \( P_g(G) \geq P_f(G) \) producing a contradiction. Thus, \( P(G) = 2mn - m + n \). □
6.4 Special Cases

In this section, we determine the exact values of the profile of $K_n \wedge K_m$ and $C_n \wedge G$.

Corollary 9 is a direct consequence of Theorem 17.

**Corollary 9** If $G = K_{1,n} \wedge K_1$, then $P(G) = 3n - 1$.

A profile numbering of $G = K_{1,5} \wedge K_1$ is shown in Figure 20.

**Lemma 8** Let $G$ be a graph of order $m$ and let $V(K_n) = \{v_1, v_2, \ldots, v_n\}$ denote the vertex set of $K_n$. Then there exists a profile numbering $f$ of $K_n \wedge G$ such that for each $u \in V(G_i)$, $f(u) < f(v_i)$ for $1 \leq i \leq n$.

**Proof:** Let $g$ be a profile numbering of $G$. We denote the vertex set of $G_i$ by $V(G_i)$. Suppose that there is an integer $i$ with $1 \leq i \leq n$ such that $g(v_i) < g(u)$ for some $u \in V(G_i)$. Let $a = g(v_i)$ and $b = \max\{g(u) : u \in G_i\}$. Define $f$ as
follows:

\[
f(v) = \begin{cases} 
  b & v = v_i \\
  g(v) - 1 & a < g(v) \leq b \\
  g(v) & g(v) < a \text{ or } g(v) > b.
\end{cases}
\]

Then for \( t < a \) or \( t > b \), \(|N_f(S_t)| = |N_g(S_t)| \). For \( a \leq t \leq b \), \(|N_f(S_t) \cap V(K_n)| \leq |N_g(S_t) \cap V(K_n)|, \) \(|N_f(S_t) \cap V(G_i)| \leq |N_g(S_t) \cap V(G_i)| \) and \(|N_f(S_t) \cap V(G_j)| = |N_g(S_t) \cap V(G_j)| \) for \( j \neq i \). So \( \sum_{i=1}^{n(m+1)} |N_f(S_t)| \leq \sum_{i=1}^{n(m+1)} |N_g(S_t)| \). Thus, \( f \) must be a profile numbering of \( K_n \land G \) and for each \( u \in V(G_i) \), \( f(u) < f(v_i) \) for \( 1 \leq i \leq n \).

\[ \square \]

**Theorem 18** If \( G = K_n \land K_m \) for positive integers \( n \) and \( m \), then

\[
P(G) = \frac{nm(m-1)}{2} + \frac{m+1}{2}(\left\lfloor \frac{n}{2} \right\rfloor^2 + \left\lfloor \frac{n}{2} \right\rfloor^2) + \frac{m-1}{2}(\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor) + \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n}{2} \right\rfloor.
\]

Proof: Let the vertices in \( K_n \) be denoted as \( v_1, v_2, \ldots, v_n \) and the vertices in \( K_m \) that are joined to \( v_i \) be \( u_{i,j} \) for \( 1 \leq i \leq n, 1 \leq j \leq m \). Let \( V = \{v_i : 1 \leq i \leq n\} \) and \( U_i = \{u_{i,j} : 1 \leq i \leq n \text{ and } 1 \leq j \leq m\} \). Now we define a proper numbering \( f \) of \( G \) as follows:

\[
f(u_{i,j}) = \begin{cases} 
  (i-1)m + j & 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor \text{ and } 1 \leq j \leq m \\
  (i-1)(m+1) + j & \left\lfloor \frac{n}{2} \right\rfloor < i \leq n \text{ and } 1 \leq j \leq m,
\end{cases}
\]

and
Figure 21. A Profile Numbering of $K_4 \wedge K_3$.

$$f(v_i) = \begin{cases} m \left\lfloor \frac{n}{2} \right\rfloor + i & 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor \\ (m + 1)i & \left\lfloor \frac{n}{2} \right\rfloor < i \leq n. \end{cases}$$

Then

$$P_f(G) = \frac{nm(m - 1)}{2} + \sum_{i=0}^{\lfloor n/2 \rfloor - 1} (m(\left\lfloor \frac{n}{2} \right\rfloor - i) + i) + \sum_{i=\lfloor n/2 \rfloor + 1}^{n} (m(i - \left\lfloor \frac{n}{2} \right\rfloor) + i - 1)$$

$$= \frac{nm(m - 1)}{2} + \frac{m + 1}{2} \left( \left\lfloor \frac{n}{2} \right\rfloor^2 + \frac{n}{2} \right) + \frac{m - 1}{2} \left( \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor \right) + \frac{n}{2} \cdot \frac{n}{2}.$$  

Figure 21 shows $f$ applied to $K_4 \wedge K_3$.

Then the following is true.
\[
|N(S_t)| = \begin{cases} 
  k + km - i & (k-1)m + 1 \leq i \leq km \text{ and } 1 \leq k \leq \left\lceil \frac{n}{2} \right\rceil \\
  n - (i - m\left\lceil \frac{n}{2} \right\rceil) & m\left\lceil \frac{n}{2} \right\rceil < i \leq (m+1)\left\lceil \frac{n}{2} \right\rceil \\
  k(m+1) - i + n - k & (k-1)(m+1) + 1 \leq i \leq k(m+1) \text{ and } \left\lceil \frac{n}{2} \right\rceil < k \leq n.
\end{cases}
\]

Suppose, to the contrary, that \( P(G) < nm(m-1)/2 + (m+1)(\left\lceil n/2 \right\rceil)^2 + (n/2)^2)/2 + (m-1)(\left\lceil n/2 \right\rceil + \left\lceil n/2 \right\rceil)/2 + [n/2][n/2] \). Let \( g \) be a profile numbering of \( G \) which satisfies Lemma 8. By Proposition 2 there exists an integer \( t \) with \( 1 \leq t \leq n(m+1) \) such that \( |N_g(S_t)| < |N_f(S_t)| \). The following cases are possible for \( t \).

**Case 1:** \( (k-1)m + 1 \leq t \leq km \) and \( 1 \leq k \leq \left\lceil n/2 \right\rceil \). Then \( S_f(t) \cap V = \emptyset \).

If \( |S_g(t) \cap V| = l \) for \( 0 < l \leq t \), then \( |N_g(S_t)| \geq n - l + km - t + l > k + km - t = |N_f(S_t)| \). Since \( f \) completely numbers one copy of \( K_m \) before numbering another copy of \( K_m \), \( f \) minimizes \( |N_f(S_t) \cap (\cup_{i=1}^{n} U_i)| \). If \( |S_g(t) \cap V| = 0 \), then \( |N_g(S_t)| = |N_g(S_t) \cap (\cup_{i=1}^{n} U_i)| \geq |N_f(S_t) \cap (\cup_{i=1}^{n} U_i)| = |N_f(S_t)| \).

**Case 2:** \( m\left\lceil n/2 \right\rceil \leq t \leq (m+1)\left\lceil n/2 \right\rceil \). In this case \( |N_f(S_t) \cap U_i| = 0 \) for \( 1 \leq i \leq n \). If \( |S_g(t) \cap V| - |S_f(t) \cap V| = l \) for \( 0 < l \leq n \), \( |N_g(S_t) \cap V| = |N_f(S_t) \cap V| - l \) and \( |N_g(S_t) \cap (\cup_{i=1}^{n} U_i)| \geq |N_f(S_t) \cap (\cup_{i=1}^{n} U_i)| + l \). So, \( |N_g(S_t)| \geq |N_f(S_t)| \). If \( |S_g(t) \cap V| = |S_f(t) \cap V| \), \( |N_g(S_t) \cap V| = |N_f(S_t) \cap V| \) and \( |N_g(S_t) \cap (\cup_{i=1}^{n} U_i)| \geq \)
\[|N_f(S_t) \cap (\cup_{i=1}^n U_i)|\). So, \(|N_g(S_t)| \geq |N_f(S_t)|\). If \(|S_f(t) \cap V| - |S_g(t) \cap V| = l\) for \(0 < l \leq \lfloor n/2 \rfloor\), \(|N_g(S_t) \cap V| = |N_f(S_t) \cap V| + l\) and \(|N_g(S_t) \cap (\cup_{i=1}^n U_i)| \geq |N_f(S_t) \cap (\cup_{i=1}^n U_i)|\). So \(|N_g(S_t)| > |N_f(S_t)|\).

**Case 3:** 
\((k - 1)(m + 1) + 1 \leq t \leq k(m + 1)\) and \([n/2] < k \leq n\). In this case, \(|N_f(S_t) \cap U_i| = 0\) for \(1 \leq i \leq k - 1\) and \(|S_g(t) \cap V| \leq |S_f(t) \cap V|\). If \(|S_g(t) \cap V| = |S_f(t) \cap V|\), then \(|N_g(S_t) \cap V| = |N_f(S_t) \cap V|\) and \(|N_g(S_t) \cap (\cup_{i=1}^n U_i)| \geq |N_f(S_t) \cap (\cup_{i=1}^n U_i)|\). So \(|N_g(S_t)| \geq |N_f(S_t)|\). If \(|S_f(t) \cap V| - |S_g(t) \cap V| = l\) for \(0 < l \leq k\), then \(|N_g(S_t) \cap V| = |N_f(S_t) \cap V| + l\) and \(|N_g(S_t) \cap (\cup_{i=1}^n U_i)| \geq |N_f(S_t) \cap (\cup_{i=1}^n U_i)| - l\). So \(|N_g(S_t)| \geq |N_f(S_t)|\).

Thus in all cases \(|N_g(S_t)| \geq |N_f(S_t)|\), producing a contradiction. Therefore, we must have \(P(G) = nm(m - 1)/2 + (m + 1)(\lfloor n/2 \rfloor + \lfloor n/2 \rfloor^2)/2 + (m - 1)(\lfloor n/2 \rfloor + \lfloor n/2 \rfloor^2)/2 + \lfloor n/2 \rfloor/2 + \lfloor n/2 \rfloor/2\).

\[\square\]

**Corollary 10** If \(G = K_n \land K_1\), then

\[P(G) = \left\lfloor \frac{n}{2} \right\rfloor^2 + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor^2.\]

In addition, it is believed the following conjecture to be true.

**Conjecture 1** If \(G\) is a graph of order \(m\), then

\[P(K_n \land G) = nP(G) + \frac{m + 1}{2}(\left\lfloor \frac{n}{2} \right\rfloor^2 + \left\lfloor \frac{n}{2} \right\rfloor) + \frac{m - 1}{2}(\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor) + \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n}{2} \right\rfloor.\]
Theorem 19 If $G$ is a graph of order $m$, then

$$
P(C_n \land G) = nP(G) + (m - 1)(2n - 3) + m \text{ for } n \geq 3.
$$

Proof: Let the vertices in $C_n$ be named sequentially $v_1 \ldots v_n$. Let $f$ be a profile numbering of $G$. Define a proper numbering $g$ of $C_n \land G$ as follows:

$$
g(x) = \begin{cases} 
(m + 1)i & x = v_i \text{ and } 1 \leq i \leq n \\
(i - 1)(m + 1) + f(x) & x \in V(G_i) \text{ and } 1 \leq i \leq n.
\end{cases}
$$

Then $P_g(C_n \land G) = nP(G) + (m + 1)(2n - 3) + m$. So, $P(C_n \land G) \leq nP(G) + (m + 1)(2n - 3) + m$. Suppose that $P(C_n \land G) < nP(G) + (m + 1)(2n - 3) + m$.

Let $h$ be a profile numbering of $C_n \land G$. Then by Proposition 2, there exists an integer $t$ with $1 \leq t < (m + 1)n$ such that $|N_h(S_t)| < |N_g(S_t)|$.

Case 1: $1 \leq t \leq m$. For either $h^{-1}(t) \in V(G_i)$ or $h^{-1}(t) \in V(C_n)$, we have $|N_h(S_t)| \geq |N_f(S_t)| + 1 = |N_g(S_t)|$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Case 2: $m < t < (m + 1)n - 1$ and $t \equiv 0 \pmod{m + 1}$. For either $h^{-1}(t) \in V(G_i)$ or $h^{-1}(t) \in V(C_n)$, we have $|N_h(S_t)| \geq 2 = |N_g(S_t)|$.

Case 3: $m < t < (m + 1)n - 1$ and $t \not\equiv 0 \pmod{m + 1}$. For either $h^{-1}(t) \in V(G_i)$ or $h^{-1}(t) \in V(C_n)$, we have $|N_h(S_t)| \geq |N_f(S_t)| + 2 = |N_g(S_t)|$.

Thus in all cases we have $|N_h(S_t)| \geq |N_g(S_t)|$, producing a contradiction.

Thus, $P(C_n \land G) = nP(G) + (m + 1)(2n - 3) + m$. □
CHAPTER VII

PROFILE OF TENSOR PRODUCT

7.1 Definition and Examples

The tensor product of two graphs was defined in Capobianco and Molluzzo[7].

Definition 6 The tensor product of graphs $G_1$ and $G_2$ is defined to be $G = (V, E)$ where $V = V(G_1) \times V(G_2)$ and $((x_1, y_1), (x_2, y_2)) \in E$ whenever $(x_1, x_2) \in E(G_1)$ and $(y_1, y_2) \in E(G_2)$.

We use $G_1(T_P)G_2$ to denote $G$. Figure 22 shows $C_3(T_P)P_4$.

Proposition 7 (From Weichsel[80]) If $G_1$ and $G_2$ are connected, then $G_1(T_P)G_2$ is connected if and only if $G_1$ or $G_2$ has an odd cycle.

Proposition 8 (From Miller[63]) If $G = G_1(T_P)G_2$ for connected graphs $G_1$ and $G_2$, then $G$ consists of exactly two components if and only if $G_1$ and $G_2$ are both bipartite.

Figure 22. $C_3(T_P)P_4$. 

65
For the tensor product of a path with a complete bipartite graph, Williams [84] provided a numbering to achieve the bandwidth and a linear algorithm to achieve edgesum. In this chapter, we provide linear algorithms to achieve the profile of paths with complete bipartite graphs.

In this chapter, when discussing $G_1(T_p)G_2$ for $G_1$ a path and $G_2$ a complete bipartite graph, we assume the vertices of $G_1$ and $G_2$ are identified in the following natural manner. In the case of a path, the vertices are identified sequentially proceeding from one end vertex of the path to the other. In the case of the complete bipartite, graph vertices are identified sequentially within each partite set in an arbitrary manner. We use $(r, i)$ to denote the $i$th vertex of the $r$-partite and $(m, i)$ to denote the $i$th vertex of the $m$-partite. We then use $(i, (r, j))$ to denote vertex $(v_{i, u_{(r,j)}}) \in V(P_n(T_p)K_{r,m})$.

By Propositions 7 and 8, for $n \geq 2$, $P_n(T_p)K_{r,m}$ always consists of two components. We define row $i$, $R_i$, of each component separately as $R_i = \{(i, j) : 1 \leq j \leq m$ and $(i, j) \in V(G)\}$. For $m \geq r$, let $H_1$ be the component of $G$ containing $(1, (m, 1))$ and $H_2$ be the other component. Figure 23 illustrates $P_6(T_p)K_{2,5}$.

**Proposition 9** (From Lin and Yuan[54]) If $G$ has components $G_1, G_2, \ldots, G_m$, then

$$P(G) = \sum_{i=1}^{m} P(G_i).$$
Figure 23. Two Components of $P_6(T_P)K_{2,5}$.

7.2 Profile of $P_n(T_P)K_{m,m}$

A graph $G_1$ is isomorphic to a graph $G_2$ if there exists a one-to-one mapping $\phi$, called an isomorphism, from $V(G_1)$ onto $V(G_2)$ such that $\phi$ preserves adjacency; that is, $uv \in E(G_1)$ if and only if $\phi u \phi v \in E(G_2)$. If $G_1$ is isomorphic to $G_2$, then we say $G_1$ and $G_2$ are isomorphic and write $G_1 \cong G_2$.

Theorem 20 Let $G = P_n(T_P)K_{m,m}$. Then

$$P(G) = \begin{cases} 
0 & n = 1 \\
m(3m - 1) & n = 2 \\
(3n - 4)m^2 - (n - 2)m & n \geq 3.
\end{cases}$$

Proof: For $n = 1$, $G = K_{2m}$; so $P(G) = 0$. For $n = 2$, $H_1 \cong H_2 \cong K_{m,m}$. By Proposition 9, $P(G) = 2P(H_1) = m(3m - 1)$. For $n \geq 3$, $H_1 \cong H_2 \cong P_n[\overline{K}_m]$. By
Proposition 9 and Theorem 5,

\[ P(G) = 2(2P(K_m) + mn(3m - 1)/2 - 2m^2 + m) = (3n - 4)m^2 - (n - 2)m. \]

\[ \square \]

7.3 Profile of \( P_n(T_P)K_r,m \) for Even Values of \( n \)

**Algorithm 4** Finds a proper numbering \( h \) for component \( H_1 \) of \( G = P_n(T_P)K_r,m \)

with \( r < m \) where \( n \) is even and \( n \geq 4 \).

Begin

1. Number \( R_1 \) with integer 1, 2, \ldots, \( m \).

2. If \( m^2 + m \geq 2r^2 \) then

   for \( i = 1 \) to \( (n - 4)/2 \)
   
   Number \( R_{2i+1} \) then \( R_{2i} \) with next \( m + r \) integers
   
   else for \( i = 1 \) to \( (n - 4)/2 \)
   
   Number \( R_{2i} \) then \( R_{2i+1} \) with next \( r + m \) integers.

3. If \( m^2 - m \geq 3r^2 - r \) then

   Number \( R_{n-1} \) then \( R_{n-2} \) then \( R_n \) sequentially

   else Number \( R_{n-2} \) then \( R_n \) then \( R_{n-1} \) sequentially.

End of Algorithm
**Theorem 21** Let $G = P_n(T_p)K_{r,m}$ with $r < m$ for even values of $n$ with $n \geq 4$. Algorithm 4 produces a profile numbering of component $H_1$ in linear time.

Proof: Let $g$ be a profile numbering of $H_1$. By Proposition 1, we know that $P(H_1) = \sum_{i=1}^{(r+m)n/2} |N_g(S_i)|$. We want to show that the numbering $h$ we get from Algorithm 4 is no worse than $g$ for $H_1$ of $G$. Since every vertex in $R_i$ is adjacent to every vertex in rows $R_{i-1}$ and $R_{i+1}$ for $1 < i < n$, $g$ must completely number a row before starting another row.

Claim 1: For $0 \leq p \leq (n-6)/2$, let $a(p) = m + p(r + m) + 1$ and $b(p) = m + (p + 1)(r + m)$, then $\sum_{t=a(p)}^{b(p)} |N(S_t)| \geq \min\{2rm + r^2 + r(r - 1)/2, 2rm + r(r - 1)/2 + m(m - 1)/2\}$. Furthermore $\sum_{t=a(p)}^{b(p)} |N(S_t)| = \min\{2rm + r^2 + r(r - 1)/2, 2rm + r(r - 1)/2 + m(m - 1)/2\}$ is achievable.

Proof of Claim 1: First suppose $g$ numbers a row $|R_i| = m$ and then numbers $R_j$ with $|R_j| = r$. Then $\sum_{t=a(p)}^{b(p)} |N(S_t)| \geq 2rm + r^2 + r(r - 1)/2$. Next suppose $g$ numbers a row $R_j$ with $|R_j| = r$ and then numbers $R_i$ with $|R_i| = m$. Then $\sum_{t=a(p)}^{b(p)} |N(S_t)| \geq 2rm + r(r - 1)/2 + m(m - 1)/2$.

Next suppose $g$ numbers a row $R_4$ with $|R_4| = m$ and then a row $R_j$ with $|R_j| = m$. Then $\sum_{t=a(p)}^{b(p)} |N(S_t)| \geq 2rm + r^2 + 2r^2 > 2rm + r^2 + r(r - 1)/2$. Next suppose $g$ numbers a row $R_4$ with $|R_4| = r$ and then a row $R_j$ with $|R_j| = r$. Then $\sum_{t=a(p)}^{b(p)} |N(S_t)| > 3rm + r(r + 1)/2 > 2rm + r^2 + r(r - 1)/2$.

Therefore, in all cases, $\sum_{t=a(p)}^{b(p)} |N(S_t)| \geq \min\{2rm + r^2 + r(r - 1)/2, 2rm + r(r - 1)/2 + m(m - 1)/2\}$. But note that $\min\{2rm + r^2 + r(r - 1)/2, 2rm + r(r - 1)/2 + m(m - 1)/2\}$ is achievable.
Claim 2: \( h \) is no worse than \( g \).

Proof of Claim 2: Let \( a = m + (n - 4)(r + m)/2 + 1 \) and \( b = (m + r)n/2 \). We establish Claim 2 by considering the following two cases.

Case 1: \( R_1 \in S_{a-1} \). In this case, \( \sum_{t=a}^{b} |N_g(S_t)| \geq \min\{2rm + r^2 + r(r - 1), 2rm + (r(r - 1) + m(m - 1))/2\} \). This is true since if there are three or more rows of length \( r \) in \( S_{a-1} \), then \( \sum_{t=a}^{b} |N_g(S_t)| \geq 3r(r - 1)/2 + 3rm > 2rm + r^2 + r(r - 1) \).

Suppose there are two rows of length \( r \) in \( S_{a-1} \).

If \( g \) numbers both rows of length \( r \) before the row of length \( m \), then \( \sum_{t=a}^{b} |N_g(S_t)| \geq 2rm + (r(r - 1) + m(m - 1))/2 \). Otherwise, \( \sum_{t=a}^{b} |N_g(S_t)| \geq 2rm + r^2 + r(r - 1) \). If there is only one row of length \( r \) in \( S_{a-1} \), then \( \sum_{t=a}^{b} |N_g(S_t)| \geq 2rm + (r^2 + m^2 - 1)/2 > 2rm + (r(r - 1) + m(m - 1))/2 \).

If there are no rows of length \( n \) in \( S_{a-1} \), then \( \sum_{t=a}^{b} |N_g(S_t)| \geq 2rm + m(m - 1)/2 + 2r(2r - 1)/2 > 2rm + (r(r - 1) + m(m - 1))/2 \). Note that \( \sum_{t=a}^{b} |N_g(S_t)| \) = \( \min\{2rm + r^2 + r(r - 1), 2rm + (r(r - 1) + m(m - 1))/2\} \) is achieved by Algorithm 4.

Case 2: \( R_1 \in S_{a-1} \). In this case \( \sum_{t=a}^{b} |N_g(S_t)| \geq \min\{2rm + r^2 + r(r - 1), 2rm + (r(r - 1) + m(m - 1))/2\} - rm \) and \( \sum_{t=1}^{m} |N_g(S_t)| \geq 2rm \). We know that \( \sum_{t=1}^{m} |N_h(S_t)| = rm \); so by the proof given above, we have \( P_h(H_1) \leq P_g(H_1) \).
Theorem 22 Let $G = P_n(T_p)K_{r,m}$ with $r < m$ and $n$ is even. Then

$$P(G) = \begin{cases} 
2(r + m - 1) & n = 2 \\
2mr(n - 1) + r^2(n - 2) + \frac{nr(r - 1)}{2} & n \geq 4 \text{ and } m^2 - m \geq 3r^2 - r \\
2mr(n - 1) + r^2(n - 4) + \frac{r(r - 1)(n - 2)}{2} & n \geq 4, m^2 + m \geq 2r^2 \\
2mr(n - 1) + \frac{(r^2 - r + m^2 - m)(n - 2)}{2} & n \geq 4 \text{ and } m^2 + m < 2r^2.
\end{cases}$$

Proof: Note that when $n$ is even, $H_1$ and $H_2$ are isomorphic. By Proposition 9, $P(G) = 2P(H_1)$. For $m = 2$, $G$ becomes two disjoint paths of length $r + m$ so $P(G) = 2(r + m - 1)$. For $m \geq 4$, by Theorem 21, when $m^2 - m \geq 3r^2 - r$ then $P(H_1) = mr(n - 1) + r^2(n - 2)/2 + nr(r - 1)/4$. When $m^2 + m < 2r^2$, $P(H_1) = 2rm + (n - 2)(2rm + r(r - 1)/2 + m(m - 1)/2)/2$. Also when $m^2 - m < 3r^2 - r$ and $m^2 + m \geq 2r^2$ then $P(H_1) = 3rm + (n - 4)(2rm + r^2 + r(r - 1)/2)/2 + (r(r - 1) + m(m - 1))/2$. By Proposition 9, the result follows. □

7.4 Profile of $P_n(T_p)K_{r,m}$ for Odd Values of $n$

Algorithm 5 Finds a proper numbering $h$ for $G = P_n(T_p)K_{r,m}$ with $r < m$, $n$ is odd and $n \geq 5$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Begin

1. Number $R_1$ of $H_1$ with integers $1, 2, \ldots, m$.

2. If $m^2 + m \geq 2r^2$ then
   
   for $i = 1$ to $(n - 3)/2$
   
   Number $R_{2i+1}$ then $R_{2i}$ of $H_1$ with the next $m + r$ integers
   
   else for $i = 1$ to $(n - 3)/2$
   
   Number $R_{2i}$ then $R_{2i+1}$ of $H_1$ with the next $r + m$ integers.

3. Number $R_n$ then $R_{n-1}$ of $H_1$ sequentially.

4. If $m^2 - m \geq 3r^2 - r$ then
   
   Number $R_2$ then $R_1$ of $H_2$ with the next $m + r$ integers
   
   else Number $R_1$ then $R_2$ of $H_2$ with the next $r + m$ integers.

5. If $m^2 + m \geq 2r^2$ then
   
   for $i = 2$ to $(n - 3)/2$
   
   Number $R_{2i}$ then $R_{2i-1}$ in $H_2$ with the next $r + m$ integers,
   
   else for $i = 2$ to $(n - 3)/2$
   
   Number $R_{2i-1}$ then $R_{2i}$ in $H_2$ with the next $m + r$ integers.

6. If $m^2 - m \geq 3r^2 - r$ then
   
   Number $R_{n-1}, R_{n-2}$ then $R_n$ of $H_2$ sequentially,
   
   else Number $R_{n-2}, R_n$ then $R_{n-1}$ of $H_2$ sequentially.

End of Algorithm
**Theorem 23** Let $G = P_n(T_p)K_{r,m}$ with $r < m$ where $n$ is odd and $n \geq 5$. Then
Algorithm 5 produces a profile numbering of $G$ in linear time.

Proof: In Algorithm 5, steps 1 to 3 number $H_1$ of $G$ and steps 4 to 6 number $H_2$ of $G$. Let $n_1 = |V(H_1)|$ and $n_2 = |V(H_2)|$. By an argument similar to that in the proof of Theorem 21, we see that steps 1 to 3 in Algorithm 5 give a profile numbering of $H_1$.

It remains to show that steps 4 to 6 of Algorithm 5 give a profile numbering of $H_2$. Define a proper numbering $h'$ of $H_2$ as $h'(v) = h(v) - n_1$, then $P_{h'}(H_2) = P_h(H_2)$. Let $g$ be a profile numbering of $H_2$. By the same reasons as seen in the proof of Theorem 21, $g$ must completely number each row before starting to number another row. The following claims can then be shown in a similar manner.

Claim 1: $\sum_{i=1}^{n+r} |N_g(S_i)| \geq \min\{2mr + m(m - 1)/2, 2mr + r^2 + r(r - 1)/2\}$.

Claim 2: For $1 \leq p \leq (n - 5)/2$, let $a(p) = p(r + m) + 1$ and $b(p) = (p + 1)(r + m)$. Then $\sum_{i=a(p)}^{b(p)} |N_g(S_i)| \geq \min\{2rm + r(r - 1)/2 + m(m - 1)/2, 2rm + r^2 + r(r - 1)/2\}$.

Claim 3: Let $a = (n - 3)(r + m)/2 + 1$ and $b = ((n - 1)m + (n + 1)n)/2$. Then $\sum_{i=a}^{b} |N_g(S_i)| \geq \min\{2rm + r(r - 1)/2 + m(m - 1)/2, 2rm + r^2 + r(r - 1)\}$.

Note that the lower bounds in all the claims are achievable by Algorithm 5. By Proposition 9, Algorithm 5 produces a profile numbering of $G$ in linear time. □
Theorem 24 Let $G = P_n(T_p) K_{r,m}$ with $r < m$ and $n$ is odd. Then

$$P(G) = \begin{cases} 
0 & n = 1 \\
4rm + r^2 + \frac{3r(r-1)}{2} & n = 3 \text{ and } m^2 - m \geq 4r^2 - 2r \\
4rm + \frac{r(r-1) + m(m-1)}{2} & n = 3 \text{ and } m^2 - m < 4r^2 - 2r \\
2mr(n - 1) + r^2(n - 2) + \frac{nr(r-1)}{2} & n \geq 5 \text{ and } m^2 - m \geq 3r^2 - r \\
2mr(n - 1) + r^2(n - 4) + \frac{r(r-1)(n-2)}{2} & n \geq 5, m^2 + m \geq 2r^2 \text{ and } m^2 - m < 3r^2 - r \\
2mr(n - 1) + \frac{(r^2 - r + m^2 - m)(n-2)}{2} & n \geq 5 \text{ and } m^2 + m < 2r^2.
\end{cases}$$

Proof: When $n = 1$ $G$ becomes a collection of isolated vertices so $P(G) = 0$. When $n = 3$, in $H_1$ we must number both end rows (i.e., $R_1$ and $R_3$) before numbering $R_2$ since every vertex in $R_2$ is adjacent to every vertex in $R_1$ and $R_3$. So, $P(H_1) = 2mr + r(r-1)/2$. In $H_2$, since we need to completely number each row before starting another, and since $R_1$ and $R_3$ are equivalent, there are only three ways to proceed. Let $f$ number $H_2$ in the order of $R_1$, then number $R_3$ then number $R_2$. Then we have $P_f(H_2) = 2mr + m(m - 1)/2$. Let $g$ number $H_2$ in the order of $R_1$ then number $R_2$ then number $R_3$. Then we have $P_g(H_2) = \ldots$
\[
rm + rm + m(m - 1)/2 + r(r - 1)/2 = 2rm + m(m - 1)/2 + r(r - 1)/2.
\]
Let \(h\) number \(H_2\) in the order of \(R_2\) then number \(R_1\) then number \(R_3\). Then we have
\[
P_h(H_2) = 2rm + r^2 + r(r - 1)/2 + r(r - 1)/2 = 2rm + r^2 + r(r - 1) = 2rm + 2r^2 - r.
\]
Since \(P_g(H_2) \geq P_f(H_2)\) is always true, then if \(m^2 - m \geq 4r^2 - 2r\), \(P(H_2) = P_h(H_2)\) else \(P(H_2) = P_f(H_2)\). By Proposition 9, when \(m^2 - m \geq 4r^2 - 2r\), \(P(G) = P(H_1) + P(H_2) = 4rm + r^2 + 3r(r - 1)/2\) and when \(m^2 - m < 4r^2 - 2r\), \(P(G) = P(H_1) + P(H_2) = 4rm + (r(r - 1) + m(m - 1))/2\). When \(n \geq 5\), by Theorem 23, for \(m^2 + m \geq 2r^2\), \(P(H_1) = 2mr + (n - 3)(2rm + r^2 + r(r - 1)/2)/2 + r(r - 1)/2\); otherwise \(P(H_1) = 2rm + (n - 3)(2rm + r(r - 1)/2 + m(m - 1)/2)/2 + r(r - 1)/2\). When \(m^2 - m \geq 3r^2 - r\), \(P(H_2) = 4rm + 2r^2 + 3r(r - 1)/2 + (n - 5)(2rm + r^2 + r(r - 1)/2)/2\). When \(m^2 - m < 3r^2 - r\) and \(m^2 + m \geq 2r^2\), \(P(H_2) = 4rm + m(m - 1) + r(r - 1)/2 + (n - 5)(2rm + r(r - 1) + m(m - 1))/2\). When \(m^2 - m < 2r^2\), \(P(H_2) = 4rm + m(m - 1) + r(r - 1)/2 + (n - 5)(2rm + r(r - 1) + m(m - 1))/2\).

By Proposition 9, the result follows. \(\square\)
CHAPTER VIII

SUMMARY

This dissertation has investigated the bandwidth, edgesum and profile of a number of classes of graphs and has provided solutions for several of them.

In Chapter II, various application areas of the graph parameters bandwidth, edgesum and profile are discussed. These include minimizing the storage and computation time for solving linear equations; solving the placement problem and the routing problem in VLSI design layout; embedding one network in another and solving constraint satisfaction problems in AI.

In Chapter III, we indicated the errors of some theorems which were given by previous researchers and gave the corrected results.

In Chapter IV, we solved the profile of the composition of paths, cycles, complete graphs and complete bipartite graphs with other graphs.

In Chapter V, the bandwidth of a d-dimension butterfly was solved. A polynomial time approximation algorithm was presented to number a butterfly in order to minimize the edgesum and the profile. Also, a polynomial time algorithm was presented to minimize the profile of a hypercube.

In Chapter VI, a tight upper bound and a tight lower bound were given for the profile of the corona of any two graphs. A tight upper bound and a tight
lower bound were provided for the profile of the corona of any connected graph with $K_m$. The exact value of the corona of a complete graph with a complete graph and a cycle with any other graph were established.

Finally, in Chapter VII, the exact value of the profile of the tensor product of a path with a complete bipartite graph was established.
Appendix A

Code of Edgesum of Butterflies
```cpp
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>

const int max = 80; // max number of vertices in a graph

class graph{

    int n; // number of vertices in the graph
    int adj[max][max]; // adjacency matrix of the graph
    int num[max]; // numbering of the graph

public:

    graph(){ for(int i = 0; i < max; i++)
            num[i] = 0; }

    void readAdj(ifstream&); // read in adjacency matrix
        // from a file

    void doit(ifstream& fin);

    void printNum(); // print out current numbering

    int edgesum(); // return the Edgesum of a numbering

    int CountNb1(int*); // count e(S_t)

    void mynum4(); // my way to number a graph - select a neighbor,
        // - keep smallest e(S_t)

    int find4(int); // find the vertex to be numbered next

};
```
void graph::doit(ifstream& fin){
    fin >> n;
    readAdj(fin);
}

void graph::readAdj(ifstream& fin){
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            fin >> adj[i][j];
}

void graph::printNum(){
    cout << endl;
    for(int i=0; i<n; i++) {
        if (num[i] < 10) cout << ' ';
        cout << num[i] << " ";
    switch(n) { // for butterfly output
    case 12: if( (!((i+1)%4)) cout << endl; break;
    case 32: if( (!((i+1)%8)) cout << endl; break;
    case 80: if( (!((i+1)%16)) cout << endl; break;
    }
int graph::edgesum(){
    int e=0; //edgesum
    for (int i=0; i<n; i++)
        for (int j=i+1; j<n; j++)
            if (adj[i][j] == 1)
                e = e + abs(num[i]-num[j]);
    return e;
}

// count number of neighbors - edge version
int graph::CountNbl(int* tnum){
    int k=0;
    for(int i=0; i<n; i++)
        if(tnum[i])
            for(int j=0; j < n; j++)
                if (adj[i][j]&&(!tnum[j])) k++;

return k;
}

int graph::find4(int c){

int tnum[max]; // numbering array that I am using now
    for(int i = 0; i < n; i++) // initialize tnum[]
        tnum[i] = num[i];

int j, m,
    k=0, // number of neighbors we have right now
    b, // the vertex that I choose to assign
    nb[max]; // neighbor list
    for(i=0; i<n; i++) // initialize neighbor list
        if(num[i])
            for(j=0; j < n; j++)
                if ((adj[i][j])&&(!tnum[j])){
                    nb[k++] = j;
                    tnum[j] = -1;
                }
    for(i=0; i<n; i++) // change the mark on the tnum[] back
        if (tnum[i] ==-1) tnum[i] = 0;


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
int min = n*(n-1)/2;

for(int l=0; l<k; l++){
    tnum[nb[l]] = c;
    m = CountNbi(tnum);  // count e(S_{cur})
    if (m < min){
        min = m;
        b =nb[l];
    }
    tnum[nb[l]] = 0;
}
return b;

void graph::mynum4(){  // for edgesum
    int i, j;
    num[0] = 1;
    for (i=2; i <= n; i++){
        j = find4(i);  // find the vertex to number
        num[j] = i;
    }
}
int main()
{

    ifstream finN("adj.dat");

    int e;
    while (finN) {
        graph g1;
        g1.doit(finN);
        if (finN.eof()) break;
        g1.mynum4();
        cout << "*** by mynum5 *** The numbering is : ";
        g1.printNum();
        e = g1.edgesum();
        cout << "The edgesum is " << e << endl << endl << endl;
    }
}
Appendix B

Code of Profile of Butterflies
```cpp
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>

const int max = 80; // max number of vertices in a graph

class graph{
    int n; // number of vertices in the graph
    int adj[max][max]; // adjacency matrix of the graph
    int num[max]; // numbering of the graph

public:
    graph(){ for(int i = 0; i < max; i++)
        num[i] = 0; }
    void readAdj(ifstream&); // read in adjacency matrix
        // from a file
    void doit(ifstream& fin);
    void printNumO; // print out current numbering
    int profile(); // return the Profile of a numbering
    int CountNb(int*); // count e(S_t)
    void mynumO; // my way to number a graph - select any vertex,
        // - keep smallest |N(S)|
    int find(int); // find the vertex which should be numbered next
};
```
void graph::doit(ifstream& fin){
    fin >> n;
    readAdj(fin);
}

void graph::readAdj(ifstream& fin){
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            fin >> adj[i][j];
}

void graph::printNum(){
    cout << endl;
    for(int i=0; i<n; i++) {
        if (num[i] < 10) cout << ' ';
        cout << num[i] << " ";
    switch(n){ // for butterfly output
        case 12: if(!((i+1)%4)) cout << endl; break;
        case 32: if(!((i+1)%8)) cout << endl; break;
        case 80: if(!((i+1)%16)) cout << endl; break;
    }
}
cout << endl << endl;

int graph::profile(){
    int p = 0; //profile
    for (int i=0; i<n; i++){
        int min = num[i];
        for (int j=0; j < n; j ++)
            if (adj[i][j] == 1 && num[j] <min)
                min = num[j];
        p = p+num[i]-min;
    }
    return p;
}

int graph::CountNb(int* tnum){
    int k=0;
    for(int i=0; i<n; i++)
if(tnum[i]&&tum[i] != -1))
    for(int j=0; j < n; j++)
        if (adj[i][j]&&(tnum[j])){
            k++;
            tnum[j] = -1;
        }
    for(i=0; i<n; i++)
        if (tum[i] ==-1) tum[i] = 0;
    return k;
}

int graph::find(int c){
    int tnum[max]; // numbering array that I am using now
    for(int i = 0; i < n; i++) // initialize tum[]
        tnum[i] = num[i];
    int m,b;
    int min = n*(n-1)/2;
    for(int l=0; l<n; l++)
        if(!tum[l]) {
            tum[l]= c;
            m = CountNb(tnum);
        }
if (m < min) {
    min = m;
    b = 1;
}

tnum[1] = 0;

return b;
}

void graph::mynum() {
    int i, j;
    num[0] = 1;
    for (i = 2; i <= n; i++) {
        j = find(i); // find the vertex to number
        num[j] = i;
    }
}

int main() {
    ifstream finN("adj.dat");
    int p;
while (finN) {

graph g1;

g1.doit(finN);

if (finN.eof()) break;

g1.mynum();

cout << "*** by mynum *** The numbering is : ";

g1.printNum();

p = g1.profile();

cout << "The profile is " << p << endl << endl << endl;
}
}
BIBLIOGRAPHY


[87] J. Yuan, “The bandwidth of the join of two graphs”, Henan Science, Vol. 8,


problems”, AAAI-90: Proceedings, Eighth National Conference on Artificial