Lower Bounds and Algorithms for Searching Networks

A Thesis
Submitted to The Faculty of Graduate Studies and Research
in Partial Fulfillment of the Requirements
For the Degree of

Doctor of Philosophy
in
Computer Science
University of Regina

By
Yuan Xue

Regina, Saskatchewan
October, 2019

© Copyright 2019: Yuan Xue
UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

SUPERVISORY AND EXAMINING COMMITTEE

Yuan Xue, candidate for the degree of Doctor of Philosophy in Computer Science, has presented a thesis titled, *Lower Bounds and Algorithms for Searching Networks*, in an oral examination held on October 15, 2019. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner: Dr. Gary MacGillivray, University of Victoria

Co-Supervisor: Dr. Boting Yang, Department of Computer Science

Co-Supervisor: Dr. Sandra Zilles, Department of Computer Science

Committee Member: *Dr. Xue Dong Yang, Department of Computer Science

Committee Member: Dr. Malek Mouhoub, Department of Computer Science

Committee Member: Dr. Chun-Hua Guo, Department of Mathematics & Statistics

Chair of Defense: Dr. Fanhua Zeng, Faculty of Graduate Studies & Research

*Not present at defense*
Abstract

Research on graph searching has recently gained interest in computer science, mathematics, and physics. This thesis provides new results on two graph search models, namely fast searching and the zero-visibility cops and robber model.

Given a graph that contains an invisible fugitive, the fast searching problem is to find the fast search number, i.e., the minimum number of searchers to capture the fugitive in the fast search model. This model was first introduced by Dyer, Yang and Yaşar in 2008. Although the literature provides a number of results on fast searching, many properties of the fast search number have not yet been revealed. In this thesis, we give new lower bounds on the fast search number. Using the new lower bounds, we prove an explicit formula for the fast search number of the cartesian product of an Eulerian graph and a path. We also give formulas for the fast search number of variants of the cartesian product. We present an upper bound on the fast search number of hypercubes, and extend the results to a broader class of graphs including toroidal grids. In addition, we examine the complete $k$-partite graphs and provide lower bounds and upper bounds on their fast search number. We also investigate some special classes of complete $k$-partite graphs, such as complete bipartite graphs and complete split graphs. We solve the open problem of determining the fast search number of complete bipartite graphs, and present
upper and lower bounds on the fast search number of complete split graphs. We also introduce the notion of $k$-combinable graphs, and propose an efficient method for computing the fast search number of such graphs.

The zero-visibility cops and robber game is a variant of Cops and Robbers subject to the constraint that the cops have no information at any time about the location of the robber. We first study a partition problem in which for a given graph and an integer $k$, we want to find a partition of the vertex set such that the size of the boundary of the smaller subset in the partition is at most $k$ while the size of this subset is as large as possible under some conditions. Then we apply such partitions to prove lower bounds on the zero-visibility cop number of graph products. We also investigate the monotonic zero-visibility cop number of graph products. In addition, we prove lower bounds on the zero-visibility cop number for various classes of graphs. In particular, we give lower bounds on the zero-visibility cop number for graph joins and lexicographic products of graphs.
Acknowledgements

I would like to express my sincere gratitude to Professor Boting Yang and Professor Sandra Zilles, my supervisors, for leading me into the splendid world of network searching. I deeply appreciate their kind guidance, insightful advice and generous support throughout all stages of this research. They are great mentors to me in both academic research and life. I am grateful for their brilliant suggestions and comments during my thesis preparation.

I would like to express my heartfelt gratitude to all my committee members, Professor Gary MacGillivray, Professor Malek Mouhoub, Professor Xue Dong Yang, and Professor Chun-Hua Guo, for their time and effort to review this thesis and providing valuable feedback, which have greatly contributed to the improvement of the thesis.

I would like to thank all my labmates and colleagues at the University of Regina for providing an enjoyable working environment throughout my Ph.D. study.

I also gratefully acknowledge the financial support from the Faculty of Graduate Studies and Research and the Department of Computer Science through scholarships and teaching assistantships.

I would like to thank my beloved parents, for their constant support, understanding and encouragement.
Contents

Abstract ................................................................. iii

Acknowledgements ................................................... v

List of Figures ......................................................... x

List of Tables .......................................................... xiv

1 Introduction ................................................................ 1

1.1 Fast Searching ....................................................... 5

1.2 Zero-Visibility Cops and Robber Game ....................... 6

1.3 Contributions and Organization of the Thesis .............. 9

2 Preliminaries ........................................................... 14

2.1 Fast Searching Model ............................................. 16

2.2 Zero-Visibility Cops and Robber Game .................... 20
3 Fast Searching on Cartesian Products

3.1 Lower Bounds and Cartesian Products ........................................... 22
  3.1.1 Lower Bounds ................................................................. 23
  3.1.2 Cartesian Products of Eulerian Graphs and Paths ... 28
  3.1.3 Variants of $B_m \square P_n$ ................................................. 42

3.2 Hypercubes and Toroidal Grids .................................................. 44

4 Fast Searching on Complete $k$-partite Graphs ................................ 53

4.1 Lower Bounds and Algorithms for Complete $k$-partite Graphs .......... 53

4.2 Complete Bipartite Graphs $K_{m,n}$ .......................................... 70
  4.2.1 Both $m$ and $n$ are Odd ............................................... 76
  4.2.2 $m$ is Odd and $n$ is Even .............................................. 78

4.3 Complete Split Graphs $S_{m,n}$ ................................................. 82
  4.3.1 $m$ is Odd and $n \geq 2$ .................................................. 83
  4.3.2 $m$ is Even and $n \geq 2$ ................................................ 84

5 Fast Searching on $k$-Combinable Graphs ........................................ 88

5.1 Align Operation on Pendant Edges .............................................. 89

5.2 Cactus Graphs ............................................................................. 94
  5.2.1 Algorithm ........................................................................... 100
5.3 Cartesian Product of a Tree and an Edge 114

6 A Partition Method for the Zero-Visibility Cops and Robber Game 128

6.1 Partitions and Boundaries of Vertex Sets 128

6.2 Lower Bounds 135

6.3 Cartesian Products 142

6.4 Strong Products 144

6.5 Hypercubes 148

7 Lower Bounds on the Cop Number 151

7.1 Lower Bounds 151

7.1.1 Graph Joins 153

7.1.2 Lexicographic Products of Graphs 159

7.1.3 Complete Multipartite Graphs 162

7.1.4 Split Graphs 166

7.2 Matching Upper Bounds 168

7.2.1 Graph Joins 168

7.2.2 Lexicographic Products of Graphs 171
7.2.3 Complete Multipartite Graphs .................................. 172
7.2.4 Split Graphs .......................................................... 174

8 Conclusions and Open Problems ........................................ 177

8.1 Summary ................................................................. 177
8.2 Open Problems ......................................................... 178

References ................................................................. 180
## List of Figures

1.1 Two examples .......................................................... 3

2.1 $\text{fs}(G_1) = 4$, $\text{fs}(G_2) = 2$, $\text{fs}(G_3) = 7$ and $\text{fs}(G_4) = 3$. ............ 18

3.1 Cartesian product of a cycle $C_3$ and a path $P_3$. .................. 30

3.2 An instance of $G'_6 \times 3$. ........................................... 31

3.3 An instance of $G'_6 \times 1$. ........................................... 31

3.4 An instance of $G''_6 \times 2$. ........................................... 31

3.5 Decomposition of $G_6 \times 5$ into $G_6 \times 2$ and $G_6 \times 3$. .................. 41

3.6 Decomposition of $G_6 \times 6$ into three copies of $G_6 \times 2$. ............ 41

3.7 An instance of $Z_6 \times 2$. ........................................... 43

3.8 $C_4 \square C_n$. ......................................................... 46

3.9 $Q_k = Q_{k-4} \square Q_4$: the vertex $Q(i,j)$, $1 \leq i, j \leq 4$, represents $Q_{k-4}^{(i,j)}$. 50
4.1 (1) After searcher $\lambda$ slides from $v_1$ to $u_1$, $v_1$ becomes the first cleared vertex of $K_{3,3}$. Let this moment be denoted by $t_1$, and we have $w_1 = v_1$. (2) Searcher $\lambda$ will slide from $u_3$ to $v_3$ in the next step. After that, $u_3$ becomes the first cleared vertex in $V_1$. Let $t_2$ denote this moment, and we have $w_2 = u_3$.

4.2 At the moment $t_1$, each vertex in $A_1$ contains a searcher. Further, each vertex in $A_1$ has cleared incident edges at $t_2$ (see Figure 4.3). In this case, $A_1 = \{v_2, v_3\}$.

4.3 At the moment $t_2$, each vertex in $A_2$ contains a searcher, and all vertices in $A_1$ and $A_2$ have cleared incident edges. In this case, $A_2 = \{v_4\}$.

4.4 In this case, $w_1 = u_1$ and $w_3 = u_2$. At this moment, $u_2$ contains exactly two searchers, each vertex in $V_2$ is occupied by exactly one searcher, and no searchers are located on other vertices.

5.1 The graph above has two pendant edges $v_3v_4$ and $v_5v_6$. Each of the pendant edges can be cleared by sliding a searcher from either of the endpoints to the other. The profile of the graph is $(1, 2, 3, 3, 1, 2, 3, 3)$, and Table 5.1 lists each search number that is associated with a specific permutation and orientation.
5.2 The graph above can be split into two vertex-disjoint trees. Note that there are exactly four edges connecting the two trees, and the fast search number of a tree can be computed in polynomial time. Upon knowing the profiles of these two trees, we then can apply the new method presented in Theorem 5.1.2 to compute the fast search number of the entire graph.

5.3 An example to illustrate that $G_i$ may not be a subgraph of $G$.

5.4 $\pi_{I,I}(G) = \pi_{I,O}(G) = \pi_{O,I}(G) = 2$, and $\pi_{O,O}(G) = 4$.

5.5 $\pi_{I,I}(G) = \pi_{I,O}(G) = 2$, $\pi_{O,I}(G) = 3$, and $\pi_{O,O}(G) = 4$.

5.6 $\pi_{I,I}(G) = \pi_{I,O}(G) = 0$, $\pi_{O,I}(G) = \pi_{O,O}(G) = 2$.

5.7 $\pi_{I,I}(G) = 1$, $\pi_{I,O}(G) = \pi_{O,I}(G) = 2$, $\pi_{O,O}(G) = 3$.

5.8 $\pi_{I,I}(G) = 0$, $\pi_{I,O}(G) = 1$, $\pi_{O,I}(G) = \pi_{O,O}(G) = 2$.

5.9 $\pi_{I,I}(G) = 1$, $\pi_{I,O}(G) = \pi_{O,I}(G) = \pi_{O,O}(G) = 3$.

5.10 An example graph $G_2$ in category 2. $s_1(G_2) = s_2(G_2) = 4$, $s_3(G_2) = 5$, $s_4(G_2) = 6$.

5.11 An example graph $G_3$ in category 3. $s_1(G_3) = s_2(G_3) = 1$, $s_3(G_3) = s_4(G_3) = 3$.

5.12 An example graph $G_4$ in category 4. $s_1(G_4) = 4$, $s_2(G_4) = 5$, $s_3(G_4) = 6$.

5.13 An example graph $G_5$ in category 5. $s_1(G_5) = 1$, $s_2(G_5) = 2$, $s_3(G_5) = s_4(G_5) = 3$. 
5.14 The above are three connected subgraphs of $T \square P_2$. Let $G'_1$,  
$G'_c$ and $G'_2$ denote the three connected subgraphs from left to  
right. Let $G' = G'_1 \cup G'_c \cup G'_2$. It is easy to see that $s_1(G'_1) =  
s_1(G'_2) = 1$, $s_2(G'_1) = s_2(G'_2) = 2$, $s_3(G'_1) = s_3(G'_2) = 3$, and  
$s_4(G'_1) = s_4(G'_2) = 3$. By using our strategies given above, we  
have $s_1(G') = 4$, $s_2(G') = 4$, $s_3(G') = 5$ and $s_4(G') = 6$. . . . . .  

6.1 Let $\ell = 3$, $i_1 = 2$ and $i_2 = 4$. $C^4_m$ contains three vertices $v_{4,2}$,  
v_{4,3} and $v_{4,4}$ in $V_1$. Each of the paths $v_{2,2}v_{3,2}v_{4,2}$, $v_{2,3}v_{3,3}v_{4,3}$ and  
$v_{2,4}v_{3,4}v_{4,4}$ contains a vertex in $\partial V_1$. . . . . . . . . . . . . . . . .  

6.2 Let $\ell' = 1$, $j_1 = 3$, $j_2 = 1$ and $j_3 = 5$. $P^1_n$, $P^5_n$ and $P^6_n$ contain two  
vertices in $V_1$ respectively. $P^1_n$ and $P^5_n$ partition all the copies  
of $P_n$ into two sets $\mathcal{P}_1$ and $\mathcal{P}_2$: $\mathcal{P}_1$ contains $P^2_n$, $P^3_n$ and $P^4_n$, and  
$\mathcal{P}_2$ contains $P^1_n$, $P^5_n$ and $P^6_n$. Note that $P^1_n$, $P^2_n$ and $P^5_n$ contain  
five vertices in $V_1$ and four of them are in $\partial V_1$. . . . . . . . . . . . . . . . .  

7.1 An example of graph join for which 3 cops are sufficient. . . .  

7.2 An example of graph join for which 4 cops are sufficient. . . .  

7.3 Four cops are sufficient for capturing the robber on $K_{3,4,5}$. . . .  

xiii
List of Tables

1.1 Zero-visibility cop number of graph products. . . . . . . . . . . 11

2.1 A comparison of the search numbers of the graphs in Figure 2.1. 19

5.1 Number of searchers for specific permutations and orientations. 90
Chapter 1

Introduction

Suppose that a fugitive invaded a complex network and is hiding at some node, thus threatening the security of the whole network. To protect the network, a set of searchers are launched into the network. The purpose of the searchers is to capture and eliminate the fugitive while the fugitive tries to escape from being captured. Then, a basic optimization question arises: what is the smallest number of searchers that can guarantee the capture of the fugitive?

Consider the tree in Figure 1.1, which is invaded by a fugitive. A set of searchers are launched into the tree for capturing the fugitive. Searchers and fugitive can move from one vertex to another if the vertices are connected by a path. In addition, we assume that the fugitive (1) stays either on a vertex or along an edge, (2) can move at a great speed along any path only when it contains no searchers, and (3) is always invisible to all searchers. The fugitive
is captured if it meets a searcher along an edge or on a vertex. The following describes how to use two searchers to capture the fugitive:

1. place searcher $\lambda_1$ on vertex $u_6$, and place searcher $\lambda_2$ on vertex $u_7$;
2. slide $\lambda_2$ from $u_7$ to $u_3$;
3. slide $\lambda_1$ from $u_6$ to $u_2$ along the path $u_6u_3u_1u_2$;
4. slide $\lambda_2$ from $u_3$ to $u_4$ along the path $u_3u_1u_2u_4$;
5. slide $\lambda_1$ from $u_2$ to $u_5$ along the edge $u_2u_5$.

Clearly, the above sliding rules for searchers guarantee the capture of the fugitive. To make the problem a bit more challenging, we assume that each edge can be traversed only once by a searcher. Consider the Petersen graph in Figure 1.1, which contains a hidden fugitive. The following describes how to use seven searchers to capture the fugitive.

1. place searcher $\lambda_i$ on $u_i$, for $1 \leq i \leq 5$, place searcher $\lambda_6$ on $u_1$ and searcher $\lambda_7$ on $v_1$;
2. slide searcher $\lambda_6$ along the path $u_1u_2u_3u_4u_5u_1$;
3. slide searcher $\lambda_i$ from $u_i$ to $v_i$, for $1 \leq i \leq 5$;
4. slide searcher $\lambda_7$ along the path $v_1v_3v_5v_2v_4v_1$.

Inspired by an article of Breisch [16] who considered the problem of finding a lost explorer in dark complex caves, Parsons [46] first introduced the
Figure 1.1: Two examples of graphs for illustrating how to capture the fugitive.

discrete version of Parsons’ model, called
edge search problem. In the edge search model, there are three actions for searchers: placing a searcher on a vertex, removing a searcher from a vertex and sliding a searcher along an edge from one endpoint to the other. An edge is cleared by a sliding action. Kirousis and Papadimitriou [35, 36] introduced the node search problem, in which there are two actions for searchers: placing and removing. An edge becomes cleared if both endpoints are occupied by searchers. Bienstock and Seymour [8] introduced the mixed search problem, which combines edge searching and node searching. In the mixed search problem, actions for searchers are the same as those in the edge search model, and an edge is cleared if both of its endpoints are occupied by searchers or cleared by a sliding action. It has also been proved that the differences of the three search numbers are bounded by a rather small constant [8, 36]. In addition, the problem of determining the edge search number (node search number or mixed search number, respectively) is NP-complete [8, 36, 39, 43].

Seymour and Thomas [50] studied a variant of node searching called helicopter cops-and-robber game, where the fugitive becomes visible. Dendris et al. [18] studied the case of a lazy robber, where the invisible fugitive can only move when a searcher is about to be placed on the vertex it currently occupies. Further, the problem of determining whether $k$ searchers suffice to capture the fugitive when it is visible or lazy is NP-complete [5]. Another variant where multiple robbers are involved was considered by Richerby and Thilikos in [49]. More variants of searching games, such as time constrained searching, can be found in [4, 6, 67, 31, 53, 55, 54].
1.1 Fast Searching

Dyer et al. [22] introduced the fast search problem, and proposed a linear time algorithm for computing the fast search number of trees. Let \( G \) denote an undirected graph. In the fast search model, a fugitive hides either on vertices or on edges of \( G \). The fugitive can move at a great speed at any time from one vertex to another along a path that contains no searchers. We call an edge *contaminated* if it may contain the fugitive, and we call an edge *cleared* if we are certain that it does not contain the fugitive. In order to capture the fugitive, one launches a set of searchers on some vertices of the graph; these searchers then clear the graph edge by edge while at the same time guarding the already cleared parts of the graph. There are two actions for searchers: placing and sliding; an edge is cleared by a sliding action and every edge is traversed exactly once. This idea is modelled by rules that describe the searchers’ allowed moves, as explained in Chapter 2. A fast search strategy of a graph is a sequence of actions of searchers that clear all contaminated edges of the graph. The fast search number of \( G \), denoted by \( \text{fs}(G) \), is the smallest number of searchers needed to capture the fugitive in \( G \).

Stanley and Yang [52] gave a linear time algorithm for computing the fast search number of Halin graphs and their extensions. They also presented a quadratic time algorithm for computing the fast search number of cubic graphs, while the problem of finding the node search number of cubic graphs is NP-complete [41]. Yang [62] proved that the problem of finding the fast search number of a graph is NP-complete; and it remains NP-complete for
Eulerian graphs. He also proved that the problem of determining whether the fast search number of $G$ equals one half of the number of odd vertices in $G$ is NP-complete; and it remains NP-complete for planar graphs with maximum degree 4. Dereniowski et al. [19] characterized graphs for which 2 or 3 searchers are sufficient in the fast search model. They proved that the fast searching problem is NP-hard for multigraphs and for graphs.

The fast search problem has a close relationship with the edge search problem [22]. Alspach et al. [3] presented a formula for the edge search number of complete $k$-partite graphs. Dyer et al. [22] calculated the fast search number of complete bipartite graphs $K_{m,n}$ when $m$ is even. They also presented lower and upper bounds on the fast search number of $K_{m,n}$ when $m$ is odd. However, the gap between the lower and upper bounds can be arbitrarily large, and the problem of closing this gap remains unsolved for eight years.

1.2 Zero-Visibility Cops and Robber Game

Cops and Robbers is a pursuit and evasion game in graph theory, which was introduced independently by Nowakowski and Winkler [44] and Quilliot [48]. These two papers consider the game with one cop and one robber and characterize the cop-win graphs. Aigner and Fromme [1] introduced the cop number, which is the smallest number of cops required for capturing the robber on a graph. More results can be found in [14, 15, 42, 45, 34, 10, 12, 13, 25].

The zero-visibility cops and robber game is a variant of Cops and Robbers,
which was proposed by Tošić [57]. This game has the same setting as Cops and Robbers except that the cops have no information about the location of the robber. As a consequence of limiting the information presented to the cops, more cops are needed to ensure the capture of the robber. The main question is to determine the minimum number of cops that can guarantee to capture the robber. Already for trees, the zero-visibility cop number can exceed the cop number, and in general the difference between these two parameters can be arbitrarily large.

When the zero-visibility cops and robber game is played on a graph $G$ with $k$ cops and one robber, the robber has full information about the locations of all cops, but the cops have no information about the location of the robber at any time, i.e., the robber is invisible to the cops. The game is played in a sequence of rounds. Each round consists of a pair of turns, a cops’ turn to move, followed by a robber’s turn to move. At round 0, each of the $k$ cops selects a vertex of $G$ to occupy, and then the robber selects a vertex of $G$. At round $i$, $i \geq 1$, first each cop takes an action, then the robber takes an action. An action of a player is to move from the vertex currently occupied to one of its neighbors or to stay still. The cops capture the robber if one of them occupies the same vertex as the robber. If this happens in a finite number of moves, then the cops win; otherwise, the robber wins. Note that we will concentrate on the zero-visibility cops and robber game. The following definitions are used only in the context of zero-visibility cops and robber game. The cop number of $G$, denoted by $c_0(G)$, is the minimum number of cops required to capture the robber on $G$. A cop-win strategy for $G$ is optimal if it uses $c_0(G)$ cops to
capture the robber. We call a vertex \textit{cleared} if it is certain that this vertex is not occupied by the robber, and \textit{contaminated} otherwise.

Monotonicity (i.e., the property that each vertex or edge, once cleared, remains cleared forever) is an important issue in graph searching problems. Megiddo et al. [43] showed that the edge search problem is NP-hard. That this problem belongs to the NP class follows from the monotonicity result in [39]. Bienstock and Seymour [8] proposed a method that gives a succinct proof for the monotonicity of the mixed search problem. This method was extended to the monotonicity of digraph search problems [64, 65]. The monotonic cop-win strategy in the zero-visibility cops and robber game is introduced in [21].

Let $R_i$ be the set of vertices that are contaminated just after the cops’ turn in the $i$-th round. We say that a cop-win strategy is monotonic if $R_{i+1} \subseteq R_i$ for any round $i \geq 0$. The \textit{monotonic cop number} of $G$, denoted by $mc_0(G)$, is the minimum number of cops required by a monotonic cop-win strategy for $G$.

Although Cops and Robbers has been widely studied, there are not many results in the study of the zero-visibility cops and robber game. Tošić [57] gave characterizations of graphs for which one cop is sufficient, and computed the cop number of paths, cycles, complete graphs and complete bipartite graphs. Dereniowski et al. [21] proved that the zero-visibility cop number of a graph is bounded above by its pathwidth and the monotonic zero-visibility cop number can be bounded both above and below by multiples of the pathwidth. Tang [56] gave a quadratic time algorithm for computing the zero-visibility cop number for a tree, which was later improved by Dereniowski et al. [20] by presenting a linear-time algorithm. Dereniowski et al. [20] also proved that the problem
of determining the zero-visibility cop number of a graph is NP-complete.

Note that the difference between the cop number in Cops and Robbers and the zero-visibility cop number can be arbitrarily large for product graphs. For example, in Cops and Robbers, we can use two cops to capture the robber on $P_m \Box P_n$ or $P_m \boxtimes P_n$, where $P_m$ is a path with $m$ vertices and $n \geq m \geq 2$. But in the zero-visibility cops and robber game, we will show that $c_0(P_m \Box P_n) \geq c_0(P_m \Box P_n) = \lceil \frac{m+1}{2} \rceil$, see Chapter 6.

Dereniowski et al. [21] showed that the zero-visibility cops and robber game is highly non-monotonic. For any $k > 1$, they constructed a class of graphs $G$ whose pathwidth is at least $k$ but $c_0(G) = 2$. Because of these notorious graphs, it is difficult to apply the lower bound techniques for pathwidth or its related graph searching models to find nontrivial lower bounds for the zero-visibility cop number.

### 1.3 Contributions and Organization of the Thesis

In this thesis, I focus on the fast searching model and zero-visibility cops and robber model. The contributions of the research are summarized as follows [58, 59, 60, 61].

1. Lower bounds on the fast search number are given. Using the lower bounds, we prove an explicit formula for the fast search number of the
cartesian product of an Eulerian graph and a path. We also give formulas for the fast search number of variants of the cartesian product. We present an upper bound and a lower bound on the fast search number of hypercubes, and extend the results to a broader class of graphs including toroidal grids.

2. We provide lower bounds and upper bounds on the fast search number of complete $k$-partite graphs. We also investigate some special classes of complete $k$-partite graphs, such as complete bipartite graphs and complete split graphs. We solve the open problem of determining the fast search number of complete bipartite graphs, and present upper and lower bounds on the fast search number of complete split graphs.

3. We introduce the notion of $k$-combinable graphs and propose an efficient method for computing the fast search number of such graphs. Algorithms for clearing cactus graphs and cartesian product of a tree and an edge, along with rigourous analysis, are then obtained from this method.

4. A new partition method is proposed that can be used to prove lower bounds for the zero-visibility cops and robber game. Note that at any moment of the zero-visibility cops and robber game, the set of cleared vertices and the set of contaminated vertices form a partition of the vertex set. This partition may change dynamically after each turn of either player. Since any subset of cops can move to their neighbors in a turn of the cops, we have many possible partitions to analyze after such a turn. To overcome this difficulty, we first establish general properties
of partitions of vertex sets, independent of the game, in Section 6.1.
We then apply those partition properties to show lower bounds on the zero-visibility cop number of cartesian products and strong products of graphs. The corresponding results are summarized in the following table.

<table>
<thead>
<tr>
<th>$G$</th>
<th>$m,n$</th>
<th>Lower bound on $c_0(G)$</th>
<th>Upper bound on $mc_0(G)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_m \Box P_n$</td>
<td>$n \geq m \geq 1$</td>
<td>$\left\lceil \frac{m+1}{2} \right\rceil$</td>
<td>$\left\lceil \frac{m+1}{2} \right\rceil$</td>
</tr>
<tr>
<td>$C_m \Box P_n$</td>
<td>$m \geq 3, n \geq 2$</td>
<td>$\min{\left\lceil \frac{m+1}{2} \right\rceil, n+1}$</td>
<td>$\min{\left\lceil \frac{m+1}{2} \right\rceil, n+1}$</td>
</tr>
<tr>
<td>$C_m \Box C_n$</td>
<td>$n \geq m \geq 3$</td>
<td>$\left\lceil \frac{m+1}{2} \right\rceil$</td>
<td>$m + 1$</td>
</tr>
<tr>
<td>$P_m \otimes P_n$</td>
<td>$n \geq m \geq 2$, $m$ is even</td>
<td>$\frac{m}{2} + 1$</td>
<td>$\frac{m}{2} + 1$</td>
</tr>
<tr>
<td>$P_m \otimes P_n$</td>
<td>$n \geq m \geq 3$, $m$ is odd</td>
<td>$\frac{m+1}{2}$</td>
<td>$\frac{m+1}{2} + 1$</td>
</tr>
<tr>
<td>$C_m \otimes P_n$</td>
<td>$m \geq 3, n \geq 2$</td>
<td>$\min{\left\lceil \frac{m+1}{2} \right\rceil, n+1}$</td>
<td>$\min{\left\lceil \frac{m+1}{2} \right\rceil, 2\left\lceil \frac{n+1}{2} \right\rceil + 1}$</td>
</tr>
<tr>
<td>$C_m \otimes C_n$</td>
<td>$n \geq m \geq 3$</td>
<td>$\left\lceil \frac{m+1}{2} \right\rceil$</td>
<td>$m + 2$</td>
</tr>
<tr>
<td>$Q_n$</td>
<td>$n \geq 2$</td>
<td>$\sum_{k=0}^{n-2} \left( \frac{k}{2} \right) + 1$</td>
<td>$\sum_{k=0}^{n-2} \left( \frac{k}{2} \right) + 1$</td>
</tr>
</tbody>
</table>

5. Lower bounds and upper bounds on the cop number of a number of classes of graphs, including graph joins, lexicographic products of graphs, complete multipartite graphs and split graphs, are presented in this thesis.

The definitions of two searching models along with preliminaries are given in Chapter 2. Chapters 3 to 5 include our results on fast searching. Chapter 3 gives lower bounds on fast search number, then gives fast searching results on cartesian products, hypercubes and toroidal grids. Algorithms for clearing complete $k$-partite graphs are described in Chapter 4, and algorithms for
clearing cacti and cartesian products of a tree and an edge are described in Chapter 5. Chapters 6 and 7 include our results on the zero-visibility cops and robber game. A partition method and lower bounds on the zero-visibility cop number are proposed in Chapter 6. Based on the lower bounds, results on cartesian products, strong products and hypercubes are given. In addition, a few more graphs are investigated and results including their cop numbers and cop-win strategies are given in Chapter 7. Chapter 8 concludes this thesis, and lists a few open problems that are worth studying in the future.

Partial results included in this thesis were published as or are to be submitted as:


- Chapter 5: Yuan Xue, Boting Yang, and Sandra Zilles. Fast searching on \(k\)-combinable graphs. To be submitted.

Chapter 2

Preliminaries

Throughout this thesis, we only consider finite undirected graphs with no loops or multiple edges. Let $G = (V, E)$ be a graph with vertex set $V$ and edge set $E$. We also use $V(G)$ and $E(G)$ to denote the vertex set and edge set of $G$ respectively. We use $uv$ to denote an edge with endpoints $u$ and $v$. For a vertex $v \in V$, the degree of $v$ is the number of edges incident on $v$, denoted $\deg_G(v)$. A leaf is a vertex that has degree one. A vertex is odd when its degree is odd. An odd graph is a graph with vertex degrees all odd. Similarly, a vertex is even when its degree is even; and an even graph is a graph with vertex degrees all even. Define $V_{\text{odd}}(G) = \{v \in V : v \text{ is odd}\}$.

For a subset $V' \subseteq V$, we use $G[V']$ to denote the subgraph induced by $V'$, which consists of all vertices of $V'$ and all the edges of $G$ between vertices in $V'$. We use $G - V'$ to denote the induced subgraph $G[V \setminus V']$. For a subset $E' \subseteq E$, we use $G - E'$ to denote the subgraph $(V, E \setminus E')$. Let
Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two subgraphs of $G$ (the intersection of $V_1$ and $V_2$ may not be empty). The union of two graphs $G_1$ and $G_2$ is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. We use $G_1 + V_2$ to denote the induced subgraph $G[V_1 \cup V_2]$ and we also use $G_1 + E_2$ to denote the subgraph $(V_1 \cup V(E_2), E_1 \cup E_2)$, where $V(E_2)$ is the vertex set of all edges in $E_2$.

Given two graphs $G$ and $H$, the cartesian product of $G$ and $H$, denoted $G \square H$, is the graph whose vertex set is the cartesian product $V(G) \times V(H)$, and in which two vertices $(u, v)$ and $(u', v')$ are adjacent if and only if $u = u'$ and $v$ is adjacent to $v'$ in $H$, or $v = v'$ and $u$ is adjacent to $u'$ in $G$. The strong product of $G$ and $H$, denoted $G \boxtimes H$, has the same vertex set as $G \square H$. Two vertices $(u, v)$ and $(u', v')$ are adjacent in $G \boxtimes H$ if and only if $u = u'$ and $v$ is adjacent to $v'$ in $H$, or $v = v'$ and $u$ is adjacent to $u'$ in $G$, or $u$ is adjacent to $u'$ in $G$ and $v$ is adjacent to $v'$ in $H$. The lexicographic product of $G$ and $H$, denoted as $G \cdot H$, is the graph whose vertex set is the cartesian product $V(G) \times V(H)$, and two vertices $(a, a')$ and $(b, b')$ are adjacent in $G \cdot H$ if and only if either $ab \in E(G)$ or $a = b$ and $a'b' \in E(H)$. Note that the lexicographic product is not commutative.

A walk is a sequence $v_0, e_1, v_1, \ldots, e_k, v_k$ of vertices and edges such that each edge $e_i$, $1 \leq i \leq k$, has endpoints $v_{i-1}$ and $v_i$. A path is a walk in which every vertex appears once, except that its first vertex might be the same as its last. We use $v_0v_1\ldots v_k$ to denote a path with ends $v_0$ and $v_k$. A cycle is a path in which its first vertex is the same as its last vertex. We use $v_0v_1\ldots v_kv_0$ to denote a cycle with $k+1$ vertices. We will also use $P_n$ to denote a path with $n$ vertices and $C_n$ to denote a cycle with $n$ vertices, respectively. A trail is a walk
that does not contain the same edge twice. For a connected subgraph $G'$ with at least one edge, an *Eulerian trail* of $G'$ is a trail that traverses every edge of $G'$ exactly once. A *circuit* is a trail that begins and ends on the same vertex. An *Eulerian circuit* of $G'$ is an Eulerian trail of $G'$ that begins and ends on the same vertex. A graph is called *Eulerian* if it contains an Eulerian circuit that traverses all its edges. We will use $B_m$ to denote an Eulerian graph with $m$ vertices. Note that we only consider finite graphs with no loops or multiple edges. So an Eulerian circuit or Eulerian subgraph contains at least three edges throughout this thesis. A *trail cover* of a graph $G$ is a family of edge-disjoint trails in $G$ that contain every edge of $G$. The minimum number of such trails is called the *trail cover number* of $G$ and is denoted by $\tau(G)$. Let $\text{pw}(G)$ denote the pathwidth of a graph $G$.

### 2.1 Fast Searching Model

In the fast search model introduced by Dyer et al. [22], an invisible fugitive hides either on a vertex or along an edge, and it can move at a high speed at any moment from a vertex to another vertex along a path that contains no searchers. We call an edge $uv$ *contaminated* if $uv$ may contain the fugitive. An edge $uv$ that does not contain the fugitive is called *cleared*. One of two actions can happen in each step of the fast search model:

- *placing* a searcher on a vertex; or
- *sliding* a searcher along a contaminated edge from one endpoint to the
other.

Note that the above sliding action is slightly different from the one used in the edge search model, in which a searcher is allowed to slide along a cleared edge. An edge $uv$ can be cleared in one of the following two ways:

- if $u$ is occupied by at least two searchers, one of them slides along $uv$ from $u$ to $v$; or
- if $u$ is occupied by only one searcher and $uv$ is the only contaminated edge incident on $u$, the searcher on $u$ slides to $v$ along $uv$.

In the fast search problem, we always suppose that all edges are contaminated initially and each edge will be cleared by a sliding action, that is, the fugitive is captured at the moment when the last contaminated edge is cleared; we also suppose that for each contaminated edge, after it is cleared, it will not get recontaminated in the remaining steps. Since searchers are allowed to slide only on contaminated edges, every edge will be traversed exactly once when all edges are cleared. A fast search strategy of a graph is a sequence of placing and sliding actions that clear all edges of the graph. Since searchers cannot be removed from the graph or “jump” from a vertex to another vertex, we can assume, without loss of generality, that all placing actions in a fast search strategy take place before all sliding actions. The fast search number of a graph $G$, denoted by $fs(G)$, is the smallest number of searchers needed to capture the fugitive in $G$. We say that a fast search strategy for $G$ is optimal if we can use this strategy to capture the fugitive and the number of searchers
required by this strategy is \( \text{fs}(G) \). As examples, some graphs and their fast search numbers are given in Figure 2.1.

![Figure 2.1: fs\((G_1) = 4, fs(G_2) = 2, fs(G_3) = 7\) and fs\((G_4) = 3\).](image)

The fast search model is a variant of the edge search model [43]. Specifically, if searchers cannot be removed from the graph and every edge can be traversed exactly once, then the edge search model is transformed to the fast search model. A relationship between the fast search model and the node search model [36] is described in Theorem 4.6 from [62]. Since the mixed search model [8] is a combination of the edge search model and the node search model, and the fast-mixed search model [63] is a combination of the mixed search model and the fast search model, it is easy to see that the fast search model is also related to these two models. Although the fast search model is related to the above graph search models, it can reflect different graph structures. To show the difference, let us consider Figure 2.1 again. Table 1 gives a comparison of the search numbers of the four graphs in Figure 2.1.

The following lemmas give two known lower bounds on the fast search number.

**Lemma 2.1.1** [22] For any connected graph \( G \), \( \text{fs}(G) \geq \frac{1}{2} |V_{\text{odd}}(G)| \).
Table 2.1: A comparison of the search numbers of the graphs in Figure 2.1.

<table>
<thead>
<tr>
<th></th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge search number</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Node search number</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Mixed search number</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>Fast search number</strong></td>
<td><strong>4</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td>Fast-mixed search number</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

**Lemma 2.1.2** [52] For any connected graph $G$ with no leaves, $\text{fs}(G) \geq \frac{1}{2} |V_{\text{odd}}(G)| + 2$.

Let $K_{n_1,\ldots,n_k} = (V_1,\ldots,V_k,E)$ denote a complete $k$-partite graph, where $V_1,\ldots,V_k$ are disjoint independent sets, $|V_i| = n_i$ and $n_i \leq n_{i+1}$ for all $1 \leq i \leq k - 1$. Each vertex in $V_i$ is adjacent to all the vertices in $V(K_{n_1,\ldots,n_k}) \setminus V_i$. We use $K_{m,n} = (V_1,V_2,E)$ to denote a complete bipartite graph, where $|V_1| = m$, $|V_2| = n$ and $1 \leq m \leq n$. We use $S_{m,n} = (V_1,V_2,E)$ to denote a complete split graph, where $V_1$ and $V_2$ are disjoint sets, $V_1$ induces a clique with $m$ vertices and $V_2$ is an independent set with $n$ vertices. In $S_{m,n}$, each vertex in $V_1$ is adjacent to all the other vertices in $V_1 \cup V_2$.

Note that for any connected graph $G$, the fast search number of $G$ is always at least the edge search number of $G$. From Theorem 2 in [3], we have the next lemma.

**Lemma 2.1.3** For any connected graph $G$ that contains a clique $K_m$ of order $m$, where $m \geq 4$, we have $\text{fs}(G) \geq m$. 19
2.2 Zero-Visibility Cops and Robber Game

A matching in $G$ is a set of edges of $G$ that share no common vertices. A perfect matching in $G$ is a matching that includes all vertices of $G$. A maximum matching in $G$ is a matching in $G$ which contains the largest possible number of edges among all the matchings in $G$. We use $\mathcal{M}(G)$ to denote a maximum matching in $G$. The number of vertices of $G$ that are not matched by edges in $\mathcal{M}(G)$ is equal to $|V(G) \setminus V(\mathcal{M}(G))|$. Thus, $|V(G)| = 2|\mathcal{M}(G)| + |V(G) \setminus V(\mathcal{M}(G))|$. Let $\mu(G) = |\mathcal{M}(G)| + |V(G) \setminus V(\mathcal{M}(G))| = |V(G)| - |\mathcal{M}(G)|$.

We now give a formal definition of the zero-visibility cops and robber game. There are two players involved in the game on graph $G$: the cop player and the robber player. The cop player controls a set of cop pieces while the robber player controls a single robber piece. The game is played in a sequence of rounds. At each round $i$, where $i \geq 0$, the cop player plays first, followed by the robber player. At round 0, both players place their pieces on some vertices of $G$. More than one piece may occupy the same vertex. All the cop pieces are visible to the robber player, while the robber piece is invisible to the cop player. In the following rounds, both players alternately move any one of their pieces from its current vertex to an adjacent vertex. The cops win if a cop piece and the robber piece occupy the same vertex after a finite number of rounds. The robber wins if such situation never happens. A cop strategy is a sequence of actions for the cop player; a cop-win strategy is a cop strategy that leads the cop player to win irrespective of the robber player’s sequence of actions. The cop number of $G$, denoted by $c_0(G)$, is the minimum number
of cops required in a cop-win strategy for $G$. A cop-win strategy for $G$ is 
optimal if it uses $c_0(G)$ cops to capture the robber on $G$. We say a cop visits
a vertex $v \in V$ at round $i$, if the cop occupies $v$ at the beginning or at the
end of round $i$. We call a vertex contaminated if it may contain the robber,
and we call a vertex cleared if the cop player can be certain that it does not
contain the robber. We refer to any unoccupied cleared vertex that turns into
a contaminated vertex as recontamination. For simplicity, we use the cops to
denote the cop pieces and the robber to denote the robber piece.
Chapter 3

Fast Searching on Cartesian Products

In this chapter, we present new lower bounds on the fast search number. Using the lower bounds, we consider the fast search number of Cartesian products of graphs and their variants.

3.1 Lower Bounds and Cartesian Products

First of all, we give new lower bounds on the fast search number. We then apply the lower bounds to prove a formula for the fast search number of the Cartesian product of an Eulerian graph and a path.
3.1.1 Lower Bounds

We first consider two lower bounds on $\text{fs}(G)$.

We now establish relations between a graph and its subgraph under some constraints.

**Lemma 3.1.1** Given a graph $G$, let $W$ be a subset of $V(G)$. If $G'$ is a graph obtained from $G$ by adding a pendant edge to each vertex in $W$, then $\text{fs}(G) \leq \text{fs}(G')$.

**Proof.** For an optimal fast search strategy $S$ of $G'$, we can modify the strategy in the following way so that the new strategy $S'$ can clear $G$. In $S$, if a searcher is placed on a vertex $v \in V(G') \setminus V(G)$, which is a leaf neighbor of a vertex $v' \in W$, then let the searcher be placed on $v'$ and remove the sliding action from $v$ to $v'$; if a searcher stays on a vertex $v \in V(G') \setminus V(G)$ at the end of searching, which is a leaf neighbor of a vertex $v' \in W$, then let the searcher stay on $v'$ and remove the sliding action from $v'$ to $v$. Clearly, the modified strategy $S'$ can clear all edges of $G$. Therefore, $\text{fs}(G) \leq \text{fs}(G')$.  

**Lemma 3.1.2** Given a graph $G$, let $W$ be a subset of $V_{\text{odd}}(G)$. If $H$ is a graph obtained from $G$ by adding a pendant edge to every vertex in $W$, then $\text{fs}(G) = \text{fs}(H)$.

**Proof.** Let $S$ be an optimal fast search strategy for $G$. From [22] we know every vertex in $V_{\text{odd}}(G)$ should contain at least one searcher at the beginning or at the end of the search process. For each searcher used in $S$, we do the
following modification to obtain a fast search strategy for $G'$: if the searcher is placed on a vertex $v \in W$ in $\mathcal{S}$, then we place him on its leaf neighbor $v' \in V(H) \setminus V(G)$, and let him slide from $v'$ to $v$; if the searcher stays on a vertex $v \in W$ in the end of $\mathcal{S}$, then we let the searcher take an extra sliding action from $v$ to its leaf neighbor $v' \in V(H) \setminus V(G)$ along the edge $vv'$.

It is easy to see that the modified search strategy can clear all edges of $H$. So $\text{fs}(G) \geq \text{fs}(H)$. It follows from Lemma 3.1.1 that $\text{fs}(G) \leq \text{fs}(H)$. Therefore $\text{fs}(G) = \text{fs}(H)$. □

The following lemma shows a relation between the trail cover number and the number of odd vertices.

**Lemma 3.1.3** If $G$ is a graph that contains at least one edge, then

$$\tau(G) = \mu(G) + |V_{\text{odd}}(G)|/2,$$

where $\mu(G)$ is the number of connected components in $G$ that are Eulerian.

**Proof.** If $V_{\text{odd}}(G) = \emptyset$, then each connected component of $G$, which contains at least one edge, is Eulerian. Thus $\tau(G) = \mu(G)$. Suppose that $V_{\text{odd}}(G) \neq \emptyset$. Let $G'$ be a graph obtained from $G$ by deleting all connected components of $G$ that are Eulerian. Note that the number of odd vertices in a graph is even. Since each odd vertex must be an end vertex of a trail in any trail cover of $G'$, we know that $\tau(G') \geq |V_{\text{odd}}(G')|/2$.

We now show that there is a trail cover of $G'$ whose cardinality is at most $|V_{\text{odd}}(G')|/2$. Let $a$ and $b$ be any two odd vertices in a connected component
of \( G' \) and let \( P \) be a trail between them. After deleting all edges of \( P \) from \( G' \), the remaining graph \( G' - E(P) \) has two fewer odd vertices than \( G' \), that is, 

\[ V_{\text{odd}}(G' - E(P)) = V_{\text{odd}}(G') \setminus \{a, b\}. \]

We repeat this process until the remaining graph \( H \) contains no odd vertices. Thus \( H \) is a graph with vertex degrees all even. Let \( S \) be the set of all trails deleted from \( G' \) and \( H' \) be the graph obtained from \( H \) by deleting all isolated vertices from \( H \). Each connected component \( H'' \) in \( H' \) must have a vertex that is contained in some trail of \( S \), say \( P \). Since \( H'' \) is Eulerian, we can merge \( H'' \) and \( P \) into a new trail \( P' \), and replace \( P \) of \( S \) by \( P' \). In this way, every edge of \( G' \) is contained in a trail of \( S \). Thus \( S \) is a trail cover of \( G' \) whose cardinality is \( |V_{\text{odd}}(G')|/2 \). Hence \( \tau(G') \leq |V_{\text{odd}}(G')|/2 \). Therefore, we have

\[
\tau(G) = \mu(G) + \tau(G') = \mu(G) + |V_{\text{odd}}(G')|/2 = \mu(G) + |V_{\text{odd}}(G)|/2.
\]

\[
\textbf{Theorem 3.1.4} \quad \text{Let} \ G \ \text{be a connected graph and} \ H \ \text{be a graph obtained from} \ G \ \text{by adding two pendant edges on each vertex of} \ G.\\

\begin{enumerate}
  \item \text{If there is an odd vertex in} \ G, \ \text{then} \ \text{fs}(H) = \tau(H) = |V(G)| + \tau(G).
  \item \text{If there is no odd vertex in} \ G, \ \text{then} \ \text{fs}(H) = \tau(H) = |V(G)|.
\end{enumerate}

\textbf{Proof.} \quad (i) \ \text{From Lemma 3.1.3 and the fact that} \ |V_{\text{odd}}(H)| = |V_{\text{odd}}(G)| + 2|V(G)|, \ \text{we have}

\[
\tau(H) = \frac{1}{2}|V_{\text{odd}}(H)| = \frac{1}{2}|V_{\text{odd}}(G)| + |V(G)| = |V(G)| + \tau(G).
\]
We now show that $\text{fs}(H) = |V(G)| + \tau(G)$. From the definition of fast searching, we know that a fast search strategy for $G$ corresponds to a trail covering of $G$. Thus, $\text{fs}(H) \geq \tau(H) = |V(G)| + \tau(G)$. On the other hand, for a vertex $v \in V(G)$, let $v'$ and $v''$ be two leaf neighbors of $v$. If we place a searcher on each $v'$ in $H$ and slide them to each vertex $v$ in $G$, then we can use $\tau(G)$ additional searchers to clear all edges of $G$. Finally, we slide a searcher on each vertex $v$ in $G$ to $v''$ in $H$. In this way we clear $H$ using $|V(G)| + \tau(G)$ searchers. So $\text{fs}(H) \leq |V(G)| + \tau(G)$, and hence, $\text{fs}(H) = |V(G)| + \tau(G)$.

(ii) Since $G$ is connected and has no odd vertex, we know that $\tau(H) = \frac{1}{2}|V_{\text{odd}}(H)| = |V(G)|$. It follows from Lemma 2.1.1 that $\text{fs}(H) \geq \frac{1}{2}|V_{\text{odd}}(H)| = |V(G)|$. If $G$ contains only one vertex, then the statement is trivial. Suppose that $G$ contains at least one edge. Since $G$ is Eulerian, it must contain at least one cycle. Then we can use a fast search strategy, which is similar to the one described in the proof of Lemma 3.1.10, to clear the graph $H$ using $|V(G)| + \tau(G)$ searchers. Thus $\text{fs}(H) \leq |V(G)|$, and therefore, $\text{fs}(H) = |V(G)|$.

We can extend the fast search strategy in the proof of Theorem 3.1.4(i) to clear the Cartesian product of $G$ and a path. So we have the following corollary.

**Corollary 3.1.5** If $G$ is a connected graph and $P$ is a path, then $\text{fs}(G \square P) \leq |V(G)| + \tau(G)|V(P)|$, where equality holds if $G$ is Eulerian and $P$ contains only one edge.

**Proof.** Note that $G \square P$ can be decomposed into $|V(P)|$ vertex-disjoint copies of $G$, which are denoted as $G^1, \ldots, G^{|V(P)|}$ respectively. We first place a
searcher on each vertex of \( G^1 \); we can use \( \tau(G) \) additional searchers to clear every edge of \( G^1 \). Then we slide a searcher from each vertex in \( G^1 \) to its corresponding vertex in \( G^2 \) along the edge between them. Similarly, we can use \( \tau(G) \) additional searchers to clear every edge of \( G^2 \). We can repeat this process to clear all the remaining contaminated edges of \( G \Box P \) if \( |V(P)| \geq 3 \).

If \( G \) is Eulerian and \( P \) contains only one edge, then \( \tau(G) = 1 \) and \( |V(P)| = 2 \), and so, from Lemma 2.1.2, we have \( \text{fs}(G \Box P) \geq \frac{1}{2}|V(\text{odd}(G \Box P))| + 2 = |V(G)| + \tau(G)|V(P)| \). Thus \( \text{fs}(G \Box P) = |V(G)| + \tau(G)|V(P)| \).

A subset \( E' \) of the edge set of a connected graph \( G \) is an edge cut of \( G \), if \( G - E' \) is disconnected. We now use an edge cut to give a lower bound on the fast search number.

**Theorem 3.1.6** Let \( G \) be a connected graph and \( E_\chi \) be an edge cut of \( G \) such that the graph \( G - E_\chi \) consists of two connected components \( G_1 \) and \( G_2 \). If each edge of \( E_\chi \) connects a vertex of \( V(G_1) \) to a vertex of \( V(G_2) \), then

\[
\text{fs}(G) \geq \text{fs}(G_1 + E_\chi) + \text{fs}(G_2 + E_\chi) - |E_\chi|.
\]

**Proof.** Let \( \mathcal{S} \) be an optimal fast search strategy for \( G \). We first consider the graph \( G_1 + E_\chi \). We modify \( \mathcal{S} \) to obtain a fast search strategy \( \mathcal{S}_1 \) that can clear \( G_1 + E_\chi \) in the following way: We first delete all actions from \( \mathcal{S} \) that are related only to \( G_2 \), i.e., “placing a searcher on a vertex of \( G_2 \)” or “sliding a searcher along an edge of \( G_2 \)”; for each edge \( v_1v_2 \) of \( E_\chi \) with \( v_1 \in V(G_1) \) and \( v_2 \in V(G_2) \), if it is cleared by the action of “sliding a searcher from \( v_2 \) to
in \( S \), then immediately before this sliding action, we insert a new placing action, i.e., “placing a searcher on \( v_2 \)”. Let \( m_1 \) be the total number of new placing actions added to \( S_1 \).

We now show how to use \( S_1 \) to clear \( G_1 + E_\chi \). Considering an edge \( v_1v_2 \in E_\chi \) with \( v_1 \in V(G_1) \) and \( v_2 \in V(G_2) \), if it is cleared by sliding a searcher from \( v_2 \) to \( v_1 \) in \( S \), since we have inserted the action of placing a searcher on \( v_2 \) in \( S_1 \), then the searcher can be used to clear \( v_1v_2 \) by sliding from \( v_2 \) to \( v_1 \). Further, in \( S_1 \), since we keep all sliding actions of \( S \) on edges of \( G_1 + E_\chi \), we know \( G_1 + E_\chi \) can be cleared in the same way as in \( S \). Therefore, \( S_1 \) can clear \( G_1 + E_\chi \).

Similarly, we can also modify the fast search strategy \( S \) to obtain a fast search strategy \( S_2 \) that can clear \( G_2 + E_\chi \). Let \( m_2 \) be the total number of new placing actions added to \( S_2 \).

From the above, we know that the total number of searchers used to clear \( G_1 + E_\chi \) and \( G_2 + E_\chi \) is \( \text{fs}(G) + m_1 + m_2 \). Thus, \( \text{fs}(G_1 + E_\chi) + \text{fs}(G_2 + E_\chi) \leq \text{fs}(G) + m_1 + m_2 \). It is easy to see that \( m_1 \) is the number of edges in \( E_\chi \) that are cleared by sliding actions in \( S \) from \( G_2 \) to \( G_1 \), and \( m_2 \) is the number of edges in \( E_\chi \) that are cleared by sliding actions in \( S \) from \( G_1 \) to \( G_2 \). Since every edge of \( E_\chi \) can be traversed exactly once, we have \( |E_\chi| = m_1 + m_2 \). Therefore, \( \text{fs}(G_1 + E_\chi) + \text{fs}(G_2 + E_\chi) \leq \text{fs}(G) + |E_\chi| \).

### 3.1.2 Cartesian Products of Eulerian Graphs and Paths

Recall that \( B_m \) is an Eulerian graph with \( m \geq 3 \) vertices and \( P_n \) is a path with \( n \) vertices. We will use \( G_{m \times n} \) to denote \( B_m \square P_n \). Let \( v_{i,j} \) denote a vertex of
$G_{m \times n}$, which corresponds to the vertex $v_i$ on $B_m$ and the vertex $v_j$ on $P_n$. So we use $B^j_m$, $1 \leq j \leq n$, to denote the Eulerian graph in $G_{m \times n}$ with vertex set \{\(v_{1,j}, v_{2,j}, \ldots, v_{m,j}\)$ (the $j$-th copy of $B_m$). Similarly, we use $P^i_n$, $1 \leq i \leq m$, to denote the path in $G_{m \times n}$ with vertex set \{\(v_{i,1}, v_{i,2}, \ldots, v_{i,n}\)$ (the $i$-th copy of $P_n$). We first give an upper bound on $fs(G_{m \times n})$.

**Lemma 3.1.7** For $m \geq 3$ and $n \geq 2$, $fs(G_{m \times n}) \leq m + n$.

**Proof.** Here is a fast search strategy that clears all edges of $G_{m \times n}$ using $m + n$ searchers.

1. Place a searcher $\lambda_i$ on each vertex $v_{i,1} \in V(G_{m \times n})$, $1 \leq i \leq m$, and place a searcher $\gamma_j$ on each vertex $v_{1,j} \in V(G_{m \times n})$, $1 \leq j \leq n$.

2. Slide $\gamma_1$ along the Eulerian circuit of $B^1_m$ to clear all its edges. Let $j = 1$.

3. Slide each $\lambda_i$ to $B^{j+1}_m$, $1 \leq i \leq m$, and then slide $\gamma_{j+1}$ along the Eulerian circuit of $B^{j+1}_m$ to clear all its edges. If $j + 1 = n$, then stop; otherwise, $j \leftarrow j + 1$ and repeat step 3.

Since the above fast search strategy has $m + n$ placing actions, we know that $fs(G_{m \times n}) \leq m + n$. 

For the fast search strategy given in the proof of Lemma 3.1.7, each $P^i_n$, $1 \leq i \leq m$, is associated with a searcher, i.e., $\lambda_i$, who clears the path, and each $B^j_m$, $1 \leq j \leq n$, is also associated with a searcher, i.e., $\gamma_j$, who clears the Eulerian graph. Note that all $\lambda_i$’s move in the same orientation simultaneously. We can also arrange all $\gamma_j$’s to make them move in the same orientation. If
this is the only way to clear $G_{m \times n}$ using $m + n$ searchers, it would not be too hard to show the optimality of this strategy. Unfortunately, we also have “non-typical” ways to clear $G_{m \times n}$ using $m + n$ searchers. For example, consider the graph $B_3 \square P_3$ (see Figure 3.1), where the $B_3$ is a 3-cycle. Besides the fast search strategy given in the proof of Lemma 3.1.7 that can clear $B_3 \square P_3$ using 6 searchers, we can also use the following “non-typical” way to clear $B_3 \square P_3$ using 6 searchers.

(1) Place two searchers on $v_{1,2}$; and place one searcher on $v_{1,1}$, $v_{2,1}$, $v_{2,2}$ and $v_{1,3}$ respectively.

(2) Slide a searcher from $v_{1,2}$ to $v_{3,1}$ along the path $v_{1,2}v_{1,1}v_{2,1}v_{3,1}$.

(3) Slide a searcher from $v_{1,1}$ to $v_{3,2}$ along the path $v_{1,1}v_{3,1}v_{3,2}$.

(4) Slide a searcher from $v_{2,1}$ to $v_{2,3}$ along the path $v_{2,1}v_{2,2}v_{3,2}v_{1,2}v_{2,2}v_{2,3}$.

(5) Slide a searcher from $v_{3,2}$ to $v_{3,3}$.

(6) Slide a searcher from $v_{1,2}$ to $v_{1,3}$ along the path $v_{1,2}v_{1,3}v_{2,3}v_{3,3}v_{1,3}$.

Figure 3.1: Cartesian product of a cycle $C_3$ and a path $P_3$.  
30
Due to the “non-typical” ways to clear $G_{m \times n}$, we need the following 6 lemmas to show that $\text{fs}(G_{m \times n}) = m + n$.

**Lemma 3.1.8** For $m \geq 3$, $\text{fs}(G_{m \times 2}) = m + 2$.

**Proof.** From Lemma 3.1.7, we have $\text{fs}(G_{m \times 2}) \leq m + 2$. Since $G_{m \times 2}$ contains $2m$ odd vertices, it follows from Lemma 2.1.2 that $\text{fs}(G_{m \times 2}) \geq m + 2$. Therefore, $\text{fs}(G_{m \times 2}) = m + 2$. □

In order to obtain a lower bound for $\text{fs}(G_{m \times 3})$, we need to consider a special graph $G'_{m \times 1}$ obtained from $G_{m \times 1}$. In general, let $G'_{m \times n}$ denote a graph obtained from $G_{m \times n}$ by adding a pendant edge to every vertex of $B_m^n$ (see Figures 3.2 and 3.3). Since $G'_{m \times 1}$ has $2m$ odd vertices, it follows from Lemma 2.1.1 that $\text{fs}(G'_{m \times 1}) \geq m$. In Lemma 3.1.10, we will give a fast search strategy for $G'_{m \times 1}$ that uses $m$ searchers. Before that, we need a structure property of the Eulerian subgraph in $G'_{m \times 1}$, which is described in Lemma 3.1.9.

Let $B$ be an Eulerian graph and $a$ be a vertex of $B$. Let $d = \deg_B(a)/2$. We repeat the following process $d$ times until the degree of $a$ is dropped to
0: select a cycle containing $a$, which has the shortest length among all the
cycles containing $a$, and then remove all the edges of the cycle from $B$. Let
$C_a^1, C_a^2, \ldots, C_a^d$ be the cycles selected in each iteration. The cycle $C_a^d$ has the
following property.

**Lemma 3.1.9** Let $a$ be a vertex of an Eulerian graph $B$. Let $C_a^1, C_a^2, \ldots, C_a^d$
be the cycles described above. Then $C_a^d$ contains two neighbors of the vertex $a$
which are not contained in any $C_a^j$, $j < d$.

**Proof.** If $d = 1$, there is only one cycle containing $a$; so the lemma is trivial.
If $d \geq 2$, since $C_a^d$ contains at least three vertices, then let $C_a^d = av_1 \ldots v_k a,$
k $\geq 2$. Suppose that $v_1$ is also contained in another cycle $C_a^j$, $1 \leq j < d$.
Note that $C_a^1, C_a^2, \ldots, C_a^d$ are edge disjont cycles. So $C_a^d$ does not contain the
edge $av_1$. Let $C_a^j = av_p \ldots v_q v_1 v_s \ldots v_t a$. It is easy to see that both cycles
$av_1 v_s \ldots v_t a$ and $av_p \ldots v_q v_1 v_s \ldots v_t a$ have shorter lengths than $av_p \ldots v_q v_1 v_s \ldots v_t a$.
This is a contradiction. Thus, $v_1$ is contained only in $C_a^d$.

Similarly, we can show that $v_k$ is only contained in $C_a^d$. Therefore, $C_a^d$
contains two neighbors of the vertex $a$ which are not contained in any $C_a^j$,
$1 \leq j < d$. ■

From Lemma 3.1.9, we can prove the following results.

**Lemma 3.1.10** $\text{fs}(G'_{m \times 1}) = m$.

**Proof.** It follows from Lemma 2.1.1 that $\text{fs}(G'_{m \times 1}) \geq m$. So we only need
to describe a fast search strategy that uses $m$ searchers to clear $G'_{m \times 1}$. Let
$B$ be the Eulerian graph obtained from $G'_{m \times 1}$ by deleting all its leaves. Let
a be a vertex that has the minimum degree among all vertices of B, and let \( d = \deg_B(a)/2 \). Let \( u \in V(B) \) be a neighbor of \( a \) and \( C^1_a, C^2_a, \ldots, C^d_a \) be the cycles in Lemma 3.1.9 such that \( u \in V(C^d_a) \) and \( u \notin V(C^j_a) \) for \( j < d \). If \( d \geq 2 \), then let \( H_u \) be a connected component that contains \( u \) after all edges of \( \cup_{1 \leq i \leq d-1} E(C^i_a) \) are deleted from \( B \), and let \( \overline{H}_u \) be a subgraph of \( B \) obtained from \( B \) by deleting all edges of \( H_u \) from \( B \). Note that both \( H_u \) and \( \overline{H}_u \) are Eulerian, and \( E(H_u) \) and \( E(\overline{H}_u) \) form a partition of \( E(B) \). If \( d = 1 \), then let \( H_u = B \). Now we give a fast search strategy for \( G'_{m \times 1} \) that uses \( m \) searchers.

1. Place a searcher on every leaf of \( G'_{m \times 1} \), except the leaf neighbor of \( u \); then slide these searchers to their non-leaf neighbors. Place another searcher on \( a \). If \( H_u = B \), go to Step 3.

2. Note that \( \overline{H}_u \) is an Eulerian subgraph, in which \( a \) is occupied by two searchers and each other vertex is occupied by one searcher. Slide one of the two searchers on \( a \) from \( a \) to itself along all edges of \( \overline{H}_u \).

3. Note that \( H_u \) is an Eulerian subgraph with \( \deg_{H_u}(a) = 2 \), in which \( a \) is occupied by two searchers, \( u \) is not occupied, and each other vertex is occupied by one searcher. Slide one of the two searchers on \( a \) from \( a \) to \( u \) along the edge \( au \), and slide the other searcher from \( a \) to \( u \) along all edges of \( H_u \) except the edge \( au \). After \( H_u \) is cleared, slide one searcher from \( u \) to its leaf neighbor on \( G'_{m \times 1} \).

Since only \( m \) searchers are placed on \( G'_{m \times 1} \) in Step 1, the above strategy clears \( G'_{m \times 1} \) using \( m \) searchers.
Lemma 3.1.11 Each optimal fast search strategy for $G'_{m \times 1}$ satisfies the following properties:

(i) the first cleared edge is cleared by sliding a searcher from a leaf to its neighbor; and

(ii) the last cleared edge is cleared by sliding a searcher from a non-leaf vertex to its leaf neighbor.

Proof. Note that $G'_{m \times 1}$ contains $2m$ odd vertices and $fs(G'_{m \times 1}) = m$. From the observation in [22], we know that in any fast search strategy, an odd vertex must contain a searcher either before the first sliding action or after the last sliding action. So, in any optimal fast search strategy for $G'_{m \times 1}$, every vertex contains exactly one searcher either before the first sliding action or after the last sliding action. For each vertex $v_{i,1}$, $1 \leq i \leq m$, which contains a searcher before the first sliding action, the searcher cannot move because $v_{i,1}$ has three contaminated edges incident on it. Thus, the first cleared edge of $G'_{m \times 1}$ must be cleared by sliding a searcher from a leaf to its neighbor.

We now consider the last sliding action that leaves $G'_{m \times 1}$ cleared. If the last action is sliding a searcher from a leaf $v_{j,2}$ to its non-leaf neighbor $v_{j,1}$, $1 \leq j \leq m$, then we know $v_{j,1}$ must contain at least two searchers after the last action. This contradicts the fact that every vertex contains at most one searcher after the last sliding action. Therefore, the last cleared edge must be cleared by sliding a searcher from a non-leaf vertex to its leaf neighbor. □
From Theorem 3.1.6, Lemmas 3.1.2, 3.1.8, 3.1.10 and 3.1.11, we can prove the following result.

**Lemma 3.1.12** For \( m \geq 3 \), \( \text{fs}(G'_m \times 3) = m + 3 \).

**Proof.** Recall that \( G'_m \times 3 \) is a graph obtained from \( B_m \square P_3 \) by adding a pendant edge to every vertex of \( B^3_m \) (see Figure 3.2). Let \( G'_m \times 1 \) be the subgraph of \( G'_m \times 3 \) which contains \( B^1_m \) (see Figure 3.3) and \( G''_m \times 2 \) be the subgraph of \( G'_m \times 3 \) after deleting all edges on \( B^1_m \) (see Figure 3.4). Recall that \( B^j_m, 1 \leq j \leq 3 \), is an Eulerian graph with vertex set \( \{v_{1,j}, v_{2,j}, \ldots, v_{m,j}\} \). For convenience, we will use \( v_{i,4}, 1 \leq i \leq m \), to denote the leaf neighbor of vertex \( v_{i,3} \).

Note that \( G''_m \times 2 \) is the graph obtained from \( G_m \times 2 \) by adding a pendant edge to every odd vertex of \( G_m \times 2 \). From Lemmas 3.1.2 and 3.1.8, we have \( \text{fs}(G''_m \times 2) = \text{fs}(G_m \times 2) = m + 2 \). So it follows from Theorem 3.1.6 and Lemma 3.1.10 that \( \text{fs}(G'_m \times 3) \geq \text{fs}(G''_m \times 2) + \text{fs}(G'_m \times 1) - m = m + 2 \).

For the sake of contradiction, we assume that \( \text{fs}(G'_m \times 3) = m + 2 \). Then let \( S_{G'_m \times 3} \) be an optimal fast search strategy for \( G'_m \times 3 \) that uses \( m + 2 \) searchers. Note that all placing actions in \( S_{G'_m \times 3} \) take place before all sliding actions. So each odd vertex of \( G'_m \times 3 \) contains at least one searcher either before the first sliding action or at the end of \( S_{G'_m \times 3} \). Thus, \( V_{\text{odd}}(G'_m \times 3) \) should contain at least \( m \) searchers at the beginning or at the end. If \( V_{\text{odd}}(G'_m \times 3) \) contains at least \( m \) searchers at the end of \( S_{G'_m \times 3} \), then, using the method in the proof of Theorem 4.4 in [52], we can reverse the strategy \( S_{G'_m \times 3} \) so that \( V_{\text{odd}}(G'_m \times 3) \) contains at least \( m \) searchers initially. So we can assume that \( V_{\text{odd}}(G'_m \times 3) \) contains \( m \) searchers before the first sliding action in \( S_{G'_m \times 3} \).
Note that the edge set \( \{ v_{i,1}v_{i,2} \mid 1 \leq i \leq m \} \) is an edge cut of \( G'_{m \times 3} \). We can modify the strategy \( S_{G'_{m \times 3}} \), using the method described in the first paragraph of the proof of Theorem 3.1.6, to obtain two separate fast search strategies \( S_{G'_{m \times 1}} \) and \( S_{G'_{m \times 2}} \) for \( G'_{m \times 1} \) and \( G'_{m \times 2} \), respectively. Since \( \{ v_{i,1}v_{i,2} \mid 1 \leq i \leq m \} \) contains \( m \) edges, we need to add \( m \) additional placing actions to obtain \( S_{G'_{m \times 1}} \) and \( S_{G'_{m \times 2}} \). Totally, \( 2m + 2 \) searchers are used in \( S_{G'_{m \times 1}} \) and \( S_{G'_{m \times 2}} \). Because \( \text{fs}(G'_{m \times 1}) = m \) and \( \text{fs}(G'_{m \times 2}) = m + 2 \), we know \( S_{G'_{m \times 1}} \) uses at least \( m \) searchers and \( S_{G'_{m \times 2}} \) uses at least \( m + 2 \) searchers. Therefore, \( S_{G'_{m \times 1}} \) uses exactly \( m \) searchers and \( S_{G'_{m \times 2}} \) uses exactly \( m + 2 \) searchers, which also means that \( S_{G'_{m \times 1}} \) and \( S_{G'_{m \times 2}} \) are both optimal. We can assume that all placing actions in \( S_{G'_{m \times 1}} \) and \( S_{G'_{m \times 2}} \) are carried out before all sliding actions.

From Lemma 3.1.11, the optimal fast search strategy \( S_{G'_{m \times 1}} \) must have the following properties: The first cleared edge of \( G'_{m \times 1} \) must be cleared by sliding a searcher from a leaf, say \( v_{i,2} \), to its neighbor \( v_{i,1} \); and the last cleared edge of \( G'_{m \times 1} \) must be cleared by sliding a searcher from a non-leaf vertex, say \( v_{i,1} \), to its leaf neighbor \( v_{i,2} \).

Since \( G'_{m \times 1} \) and \( G''_{m \times 2} \) share each edge \( v_{i,1}v_{i,2} \), \( 1 \leq i \leq m \), we know \( S_{G'_{m \times 1}} \) and \( S_{G''_{m \times 2}} \) must share the sliding action on each edge \( v_{i,1}v_{i,2} \), \( 1 \leq i \leq m \). Thus \( S_{G''_{m \times 2}} \) must satisfy the following conditions: The first cleared edge in the set \( \{ v_{i,1}v_{i,2} \mid 1 \leq i \leq m \} \) must be cleared by sliding a searcher from \( v_{i,2} \) to \( v_{i,1} \); and the last cleared edge in the set \( \{ v_{i,1}v_{i,2} \mid 1 \leq i \leq m \} \) must be cleared by sliding a searcher from \( v_{i,2} \) to \( v_{i,1} \).
From the assumption for $S_{G_m \times 3}$, we know that $V_{odd}(G''_{m \times 2})$ contains at least $m$ searchers before the first sliding action in $S_{G_m \times 2}$. Thus, $B_m^2$ and $B_m^3$ contain at most two searchers before the first sliding action of $S_{G_m \times 2}$. Moreover, $B_m^2$ should contain at least one searcher before the first sliding action of $S_{G_m \times 2}$. This is because the first cleared edge in the set $\{v_{i,1}v_{i,2} \mid 1 \leq i \leq m\}$ is cleared by sliding a searcher from $v_{i,2}$ to $v_{i,1}$. If $B_m^2$ does not contain any searcher before the first sliding action, then we know no searcher can slide from $v_{i,2}$ to $v_{i,1}$. Suppose that $B_m^2$ contains two searchers before the first sliding action of $S_{G_m \times 2}$. One of the following two cases must happen.

Case 1. There are two vertices of $B_m^2$, each of which contains one searcher before the first sliding action of $S_{G_m \times 2}$. It is easy to see that the two searchers cannot move until another searcher slides to them. We also know that each searcher placed on $v_{j,1}$, $1 \leq j \leq m$, cannot move until a searcher slides from $v_{i,2}$ to $v_{i,1}$. Searchers that are placed on a subset of $\{v_{j,4} \mid 1 \leq j \leq m\}$ can slide to $B_m^3$. However, they would be stuck on $B_m^3$ since $B_m^3$ contains no searchers. Thus, no edges between $B_m^2$ and $B_m^3$ can be cleared, which is a contradiction.

Case 2. There is a vertex $v_{k,2}$, $1 \leq k \leq m$, of $B_m^2$ that contains two searchers before the first sliding action of $S_{G_m \times 2}$. Then we have three subcases.

Case 2.1. The first cleared edge incident on $v_{k,2}$ is cleared by sliding one of the two searchers from $v_{k,2}$ to $v_{k,1}$ (in this case, $k = i_1$). After this action, the other searcher contained in $v_{k,2}$ cannot move until another searcher slides to $v_{k,2}$. Further, since all the vertices of $B_m^2$ and $B_m^3$ except $v_{k,2}$ contain no
searcher initially, we know searchers sliding from $v_{i,1}$ to $v_{i,2}$, $1 \leq i \leq m$, or from $v_{j,4}$ to $v_{j,3}$, $1 \leq j \leq m$, are stuck on $B^2_m$ or $B^3_m$ respectively. Thus, no edges between $B^2_m$ and $B^3_m$ can be cleared. This is a contradiction.

Case 2.2. The first cleared edge incident on $v_{k,2}$ is cleared by sliding one of the two searchers from $v_{k,2}$ to $v_{k,3}$. After this action, the other searcher contained in $v_{k,2}$ cannot move until another searcher slides to $v_{k,2}$. If a searcher slides from $v_{j,3}$ to $v_{j,2}$, $1 \leq j \leq m$ and $j \neq k$, then we know the searcher must be stuck on $v_{j,2}$ since then. Note that the first cleared edge between $B^2_m$ and $B^1_m$ is cleared by sliding a searcher from $v_{i_{1,2}}$ to $v_{i_{1,1}}$. But no searcher can slide from $v_{i_{1,2}}$ to $v_{i_{1,1}}$. This is a contradiction.

Case 2.3. The first cleared edge incident on $v_{k,2}$ is cleared by sliding one of the two searchers from $v_{k,2}$ to $v_{k',2}$, which is a neighbor of $v_{k,2}$ on $B^2_m$. Similar to Case 1, no edges between $B^2_m$ and $B^3_m$ can be cleared, which brings a contradiction.

Therefore, $B^2_m$ must contain exactly one searcher before the first sliding action of $S^m_{G''_{m \times 2}}$. Further, $B^3_m$ also contains exactly one searcher before the first sliding action of $S^m_{G''_{m \times 2}}$; otherwise, no searcher can slide from $B^3_m$ to $B^2_m$. Note that the remaining $m$ searchers are placed on $m$ odd vertices of $G''_{m \times 2}$ respectively. Since each leaf contains at most one searcher all the time in an optimal strategy, we know these $m$ searchers are placed on $m$ distinct odd vertices of $G''_{m \times 2}$.

Before the first sliding action of $S^m_{G''_{m \times 2}}$, let $v_{t_1,2}$ be a vertex of $B^2_m$ that contains a searcher, and let $v_{t_2,3}$ be a vertex of $B^3_m$ that contains a searcher.
Since both \( \deg(v_{\ell_1,2}) \) and \( \deg(v_{\ell_2,3}) \) are at least 4, each of \( v_{\ell_1,2} \) and \( v_{\ell_2,3} \) will be occupied by at least one searcher throughout \( S_{G_{m \times 2}''} \). Note that all other \( m \) searchers will stop on the \( m \) leaves of \( G_{m \times 2}''' \), on which no searchers are placed before the first sliding action of \( S_{G_{m \times 2}'''} \). Hence, at the end of \( S_{G_{m \times 2}'''} \), \( B_m^2 \) contains exactly one searcher that occupies \( v_{\ell_1,2} \), and \( B_m^3 \) contains exactly one searcher that occupies \( v_{\ell_2,3} \). If no searcher slides from \( v_{\ell_1,3} \) to \( v_{\ell_1,2} \), then searchers sliding from \( B_m^3 \) to \( B_m^2 \) would be stuck on \( B_m^2 \) and no searcher can clear the edge \( v_{i_1,2}v_{i_1,1} \). Thus the edge \( v_{i_1,2}v_{i_1,3} \) is cleared by sliding a searcher from \( v_{i_1,3} \) to \( v_{i_1,2} \). Similarly, the edge \( v_{i_2,3}v_{i_2,4} \) is cleared by sliding a searcher from \( v_{i_2,4} \) to \( v_{i_2,3} \); otherwise, searchers sliding from a subset of \( \{ v_{j,4} \mid 1 \leq j \leq m \} \) to \( B_m^3 \) would be stuck on \( B_m^3 \).

Let \( t \) denote the moment just after the last contaminated edge, i.e., \( v_{i_2,1}v_{i_2,2} \), in \( \{ v_{i,1}v_{i,2} \mid 1 \leq i \leq m \} \) is cleared. Since the edge \( v_{i_2,1}v_{i_2,2} \) is cleared by sliding a searcher from \( v_{i_2,1} \) to \( v_{i_2,2} \), we know that at the moment \( t \), \( v_{i_2,2} \) should contain at least one searcher.

If all edges between \( B_m^2 \) and \( B_m^3 \) are cleared at the moment \( t \), \( B_m^2 \) would contain at least two searchers at the end because \( v_{\ell_1,2} \) always contains a searcher throughout \( S_{G_{m \times 2}'''} \). This is a contradiction. If there are edges between \( B_m^2 \) and \( B_m^3 \) that are not cleared at the moment \( t \), we have two cases.

Case 1. All edges of \( B_m^2 \) are cleared at \( t \). Then \( v_{i_2,2} \) contains two searchers at \( t \). We have two subcases:

Case 1.1. \( i_2 = \ell_1 \). Because the edge \( v_{\ell_1,2}v_{\ell_1,3} \) is cleared by sliding a searcher from \( v_{\ell_1,3} \) to \( v_{\ell_1,2} \), the searchers contained in \( v_{i_2,2} \) cannot slide to \( B_m^3 \) along
Since all edges of $B_m^2$ are cleared, we know $v_{i_2,2}$ would contain at least two searchers at the end of $S_{G_m^{n \times 2}}$, which is a contradiction.

Case 1.2. $i_2 \neq \ell_1$. Since all edges of $B_m^2$ are cleared, there is at most one contaminated edge incident on $v_{i_2,2}$. Thus, $v_{i_2,2}$ must contain at least one searcher at the end of $S_{G_m^{n \times 2}}$. This is also a contradiction.

Case 2. There are edges of $B_m^2$ that are not cleared at $t$. Consider a sliding action that leaves all edges of $B_m^2$ cleared. Let $v_{j_2,1}v_{j_2,2}$ denote the last cleared edge of $B_m^2$ which is cleared by sliding a searcher from $v_{j_1,2}$ to $v_{j_2,2}$. Then $v_{j_2,2}$ contains two searchers at the moment when $v_{j_1,2}v_{j_2,2}$ becomes cleared. If $j_2 = \ell_1$, $v_{j_2,2}$ would contain two searchers at the end since the edge $v_{j_2,2}v_{j_2,3}$ must be cleared by sliding a searcher from $v_{j_2,3}$ to $v_{j_2,2}$. This is a contradiction. If $j_2 \neq \ell_1$, since the edge $v_{j_2,2}v_{j_2,1}$ has been cleared, we know that $v_{j_2,2}$ would contain a searcher at the end, which is a contradiction.

From the above, we know $fs(G_m^{n \times 3}) \geq m + 3$. Therefore, from Lemmas 3.1.7 and 3.1.2, we have $fs(G_m^{n \times 3}) = m + 3$. 

Lemma 3.1.13 For $m \geq 3$ and $n \geq 2$, $fs(G_{m \times n}) \geq m + n$.

Proof. If $n = 2$, it follows from Lemma 3.1.8 that $fs(G_{m \times 2}) = m + 2$. If $n = 3$, from Lemmas 3.1.2 and 3.1.12, we have $fs(G_{m \times 3}) = fs(G_m^{n \times 3}) = m + 3$. We now suppose that $n \geq 4$.

If $n$ is odd, then we can decompose $G_{m \times n}$ into one $G_{m \times 3}$ and $(n - 3)/2$ copies of $G_{m \times 2}$ (see Figure 3.5). From Theorem 3.1.6, we have
\[
\begin{align*}
\text{fs}(G_{m \times n}) & \geq \text{fs}(G'_{m \times 3}) + \text{fs}(G'_{m \times (n-3)}) - m \\
& \geq m + 3 + \text{fs}(G''_{m \times 2}) + \text{fs}(G''_{m \times (n-5)}) - 2m \\
& \geq m + 3 + \frac{1}{2}(n - 3)(m + 2) - \frac{1}{2}(n - 3)m \\
& = m + n.
\end{align*}
\]

If \( n \) is even, then we decompose \( G_{m \times n} \) into \( n/2 \) copies of \( G_{m \times 2} \) (see Figure 3.6). Similar to the above case, we have

\[
\text{fs}(G_{m \times n}) \geq \frac{1}{2}n(m + 2) - (\frac{1}{2}n - 1)m = m + n.
\]

Therefore, \( \text{fs}(G_{m \times n}) \geq m + n \) when \( n \geq 2 \).
From Lemmas 3.1.7 and 3.1.13, we have the main result of this section.

**Theorem 3.1.14** For $m \geq 3$ and $n \geq 2$, $fs(B_m \square P_n) = m + n$.

### 3.1.3 Variants of $B_m \square P_n$

From the proofs in Section 3.1.2, we know that even if each Eulerian graph $B^i_m$, $1 \leq j \leq n$, is replaced by an arbitrary Eulerian graph with $m$ vertices (all these Eulerian graphs may be different), we can still prove the same results. This means Theorem 3.1.14 holds for a larger class of graphs including all $B_m \square P_n$.

**Theorem 3.1.15** Let $W_{m,n}$ be a graph obtained from $B_m \square P_n$ ($m \geq 3$, $n \geq 2$) by replacing each Eulerian graph $B^i_m$ ($1 \leq j \leq n$) by an arbitrary Eulerian graph with $m$ vertices. Then $fs(W_{m,n}) = m + n$.

We now consider another variant of the Cartesian product $B_m \square P_n$. For every $B^i_m$ on $B_m \square P_n$, $1 \leq i \leq n$, we select a vertex $v_{x,i}$, $1 \leq x_i \leq m$. We then add an edge between $v_{x,i}$ and $v_{x+i, i+1}$ for every $i = 1, \ldots, n - 1$. Let the new graph denoted by $Z_{m \times n}$ (see Figure 3.7).

**Lemma 3.1.16** For $m \geq 3$ and $n \geq 2$, $fs(Z_{m \times n}) = m + 1$.

**Proof.** From Lemma 2.1.2, we have $fs(Z_{m \times n}) \geq \frac{1}{2}V_{\text{odd}}(Z_{m \times n}) + 2 = m + 1$.

The following is a fast search strategy that uses $m + 1$ searchers to clear $Z_{m \times n}$.

1. Place a searcher $\lambda_i$ on each vertex $v_{i,1} \in V(Z_{m \times n})$, $1 \leq i \leq m$, and place a searcher $\gamma$ on the vertex $v_{x,1}$.
2. Slide $\gamma$ along the edges of the Eulerian graph $B_m^1$ to clear all its edges. Let $j = 1$.

3. Slide $\gamma$ to $v_{x_j,j+1}$ along the edge $v_{x_j,j}v_{x_{j+1},j+1}$, and slide each $\lambda_i$ to $B_m^{j+1}$, $1 \leq i \leq m$. Then slide $\gamma$ along the edges of the Eulerian graph $B_m^{j+1}$ to clear all its edges. If $j + 1 = n$, then stop; otherwise, $j \leftarrow j + 1$ and repeat step 3.

It is easy to see that the lower bound and the fast search strategy in the proof of Lemma 3.1.16 can also be applied to the graph $W_m'$ defined in the following corollary.

**Corollary 3.1.17** Let $W_m'$ be a graph obtained from $Z_m\times n$ ($m \geq 3$, $n \geq 2$) by replacing each Eulerian graph $B_m^j$ ($1 \leq j \leq n$) by an arbitrary Eulerian graph with $m$ vertices. Then $fs(W_m') = m + 1$.

Note that by adding a path with $n$ vertices to $B_m\square P_n$, the fast search number of the new graph $Z_m\times n$ is greatly reduced compared to $fs(B_m\square P_n)$. 

43
This demonstrates that the fast searching problem is not subgraph-closed.

3.2 Hypercubes and Toroidal Grids

In this section, we investigate the fast search number of hypercubes and the fast search number of toroidal grids. Let $Q_k, k \geq 0$, denote a $k$-dimensional hypercube.

**Theorem 3.2.1** If $k$ is odd and $k \geq 3$, then $fs(Q_k) = 2^{k-1} + 2$.

**Proof.** Note that $Q_k$ has $2^k$ vertices and every vertex has degree $k$. Since $k$ is odd and $k \geq 3$, we know that $Q_{k-1}$ is an Eulerian graph with $2^{k-1}$ vertices. It follows from Theorem 3.1.14 that $fs(Q_k) = fs(Q_{k-1} \Box P_2) = 2^{k-1} + 2$.

Observe that $Q_k = Q_{k-2} \Box Q_2 = Q_{k-2} \Box C_4$ and $Q_{k-2}$ is an Eulerian graph when $k$ is even and $k \geq 4$. This motivates us to consider $fs(B_m \Box C_n)$, where $B_m$ is an Eulerian graph with $m$ vertices. Although $B_m \Box C_n$ is a simple extension of $B_m \Box P_n$ which was considered in the previous section, it turns out to be much more difficult to find a nontrivial lower bound on $fs(B_m \Box C_n)$. We first give an upper bound on $fs(B_m \Box C_n)$.

**Lemma 3.2.2** If $m \geq 3$ and $n \geq 3$, then $fs(B_m \Box C_n) \leq 2m + n - 2$.

**Proof.** Let $v_{i,j}$ denote a vertex of $B_m \Box C_n$, which corresponds to the vertex $v_i$ on $B_m$ and the vertex $v_j$ on $C_n$, and let $B_{m,j}^i$, $1 \leq j \leq n$, denote the Eulerian graph in $B_m \Box C_n$ with vertex set $\{v_{1,j}, v_{2,j}, \ldots, v_{m,j}\}$. The following is a fast search strategy that clears $B_m \Box C_n$ using $2m + n - 2$ searchers.
(1) For $B^1_m$, place two searchers on each $v_{i,1}$, $1 \leq i \leq m$.

(2) For each $B^j_m$, $2 \leq j \leq n - 1$, place one searcher on $v_{1,j}$.

(3) Since $B^1_m$ is an Eulerian graph, slide a searcher from $v_{1,1}$ to itself along all edges of $B^1_m$.

(4) After Step (3), each vertex of $B^1_m$ still contains two searchers, and all its edges are cleared. So for each vertex $v_{i,1}$, $1 \leq i \leq m$, we can slide one searcher from $v_{i,1}$ to its neighbor on $B^2_m$, and slide the other searcher from $v_{i,1}$ to its neighbor on $B^a_m$, along the edge between them.

(5) After Step (4), each vertex of $B^2_m$ contains one searcher, except $v_{1,2}$, which contains two searchers. First slide a searcher from $v_{1,2}$ to itself along all edges of $B^2_m$, and then slide a searcher from each $v_{i,2}$, $1 \leq i \leq m$, to its neighbor on $B^3_m$ along the edge between them. Continuing like this (if $n \geq 4$), we can clear each $B^j_m$, $2 \leq j \leq n - 1$. Then, slide a searcher from each $v_{i,n-1}$, $1 \leq i \leq m$, to its neighbor on $B^a_m$ along the edge between them.

(6) After Step (5), each vertex of $B^a_m$ contains two searchers. So we can easily slide a searcher from $v_{1,m}$ to itself along all edges of $B^a_m$.

From Steps (1) and (2), we know that $2m + n - 2$ searchers are placed on $B_m \square C_n$. Thus, $fs(B_m \square C_n) \leq 2m + n - 2$. $lacksquare$

From the above proof, we know that even if each Eulerian graph $B^j_m$, $1 \leq j \leq n$, is any arbitrary Eulerian graph with $m$ vertices, we can still prove the same results.

**Corollary 3.2.3** Let $D_{m,n}$ be a graph obtained from $B_m \square C_n$ ($m \geq 3$, $n \geq 3$).
by replacing each Eulerian graph $B_j^m$ ($1 \leq j \leq n$) by an arbitrary Eulerian graph with $m$ vertices. Then $fs(D_{m,n}) \leq 2m + n - 2$.

From the structure of $B_m \Box C_n$, the fast search strategy described in the proof is essential and the upper bound in Lemma 3.2.2 seems hard to beat. But to our surprise, when $B_m$ is a cycle with at least four vertices, we can use a new technique to improve this upper bound for toroidal grids $C_m \Box C_n$.

**Theorem 3.2.4** If $n \geq m \geq 4$, then $fs(C_m \Box C_n) \leq 2m + n - 3$.

**Proof.** Let $u_{i,j}$ denote a vertex of $C_m \Box C_n$, which corresponds to the $i$-th vertex of $C_m$ and the $j$-th vertex of $C_n$. Let $C_m^j$, $1 \leq j \leq n$, denote the cycle $u_{1,j}u_{2,j} \ldots, u_{m,j}u_{1,j}$ (the $j$-th copy of $C_m$ in $C_m \Box C_n$). We now give a fast search strategy that uses $2m + n - 3$ searchers to clear $C_m \Box C_n$ (see Figure 3.8).

(1) For $C_m^1$, place two searchers on $u_{1,1}$, one searcher on $u_{2,1}$, and three searchers on $u_{m,1}$; if $m \geq 5$, place two searchers on each $u_{i,1}$, $4 \leq i \leq m - 1$. 

![Figure 3.8: $C_4 \Box C_n$.](image)
(2) For $C_m^2$, place two searchers on $u_{3,2}$; and for each $C_m^j$, $3 \leq j \leq n - 1$, place one searcher on $u_{3,j}$.

(3) Slide a searcher from $u_{m,1}$ to $u_{2,1}$ along the path $u_{m,1}u_{1,1}u_{2,1}$.

(4) After Step (3), each vertex of $C_m^1$, except $u_{3,1}$, contains two searchers. Slide a searcher from each $u_{i,1}$, $1 \leq i \leq m$ and $i \neq 3$, to its neighbor on $C_m^2$ along the edge between them.

(5) After Step (4), each vertex of $C_m^2$ contains one searcher, except $u_{3,2}$ that contains two searchers. First slide a searcher from $u_{3,2}$ to itself along all edges of $C_m^2$, and then slide this searcher from $u_{3,2}$ to $u_{3,1}$ along the edge between them.

(6) After Step (5), each vertex of $C_m^1$ and $C_m^2$ contains exactly one searcher. Since all edges of $C_m^2$ are cleared, slide a searcher from each $u_{i,2}$, $1 \leq i \leq m$, to its neighbor on $C_m^3$ along the edge between them.

(7) Since each vertex of $C_m^3$ contains one searcher, except $u_{3,3}$, which contains two searchers, slide a searcher from $u_{3,3}$ to itself along the edges of $C_m^3$. Then slide a searcher from each $u_{i,3}$, $1 \leq i \leq m$, to its neighbor on $C_m^4$ along the edge between them. Continuing like this we can clear each $C_m^j$, $3 \leq j \leq n - 1$. After that, slide a searcher from each $u_{i,n-1}$, $1 \leq i \leq m$, to its neighbor on $C_m^n$ along the edge between them.

(8) After Step (7), each vertex of $C_m^1$ and $C_m^n$ contains exactly one searcher. From Step (3), we know that $u_{1,1}$ has only one contaminated edge, i.e., $u_{1,1}u_{1,n}$, incident on it. So, slide the searcher on $u_{1,1}$ to $u_{1,n}$ along this edge, and then slide this searcher from $u_{1,n}$ to itself along the edges of $C_m^n$.  

47
(9) Slide a searcher from each $u_i, 2 \leq i \leq m$, to its neighbor on $C_m^1$ along the edge between them. After this, each vertex of $C_m^1$, except $u_{1,1}$, contains two searchers. Slide a searcher from $u_{2,1}$ to $u_{m,1}$ along the path $u_{2,1}u_{3,1}\ldots u_{m,1}$.

Now every edge of $C_m \square C_n$ is cleared.

From Steps (1) and (2), we know that $2(m - 1)$ searchers are placed on $C_m^1$, 2 searchers are placed on $C_m^2$, and one searcher is placed on each $C_m^j$, $3 \leq j \leq n - 1$. In total, $2m + n - 3$ searchers are placed on $C_m \square C_n$. Therefore, $\text{fs}(C_m \square C_n) \leq 2m + n - 3$. 

We now consider the upper bound of $\text{fs}(Q_k)$ when $k$ is even. Since $Q_k = Q_{k-2} \square C_4$ and $Q_{k-2}$ is an Eulerian graph when $k$ is even and $k \geq 4$, it follows from Lemma 3.2.2 that $\text{fs}(Q_k) \leq 2^{k-1}+2$. In the following theorem, we improve this upper bound using a strategy similar to the one described in the proof of Theorem 3.2.4.

**Theorem 3.2.5** If $k$ is even and $k \geq 4$, then $\text{fs}(Q_k) \leq 2^{k-1} + 1$.

**Proof.** If $k = 4$, then from Theorem 3.2.4 we have $\text{fs}(Q_4) \leq 9$, and so the theorem is proved. Suppose that $k \geq 6$ and $k$ is even. We observe that $Q_k = Q_{k-4} \square Q_4 = Q_{k-4} \square C_4 \square C_4$. Let $Q_{k-4}^{(i,j)}$, $1 \leq i, j \leq 4$, denote a copy of $Q_{k-4}$ in $C_4 \square C_4$ (see Figure 3.9). Let $Q_{k-2}^{(j)}$, $1 \leq j \leq 4$, denote a copy of $Q_{k-2}$ in $Q_k$ which is induced by the vertices of $Q_{k-4}^{(1,j)}$, $Q_{k-4}^{(2,j)}$, $Q_{k-4}^{(3,j)}$, and $Q_{k-4}^{(4,j)}$. We now describe a fast search strategy that clears $Q_k$ using $2^{k-1} + 1$ searchers.

(1) Place two searchers on each vertex of $Q_{k-4}^{(1,1)}$, place one searcher on each vertex of $Q_{k-4}^{(2,1)}$, place three searchers on each vertex of $Q_{k-4}^{(3,1)}$; place two
searchers on each vertex of $Q_{k-4}^{(3,2)}$; and place an additional searcher on just one vertex of $Q_{k-2}^{(3)}$.

(2) For $Q_{k-4}^{(1,1)}$, slide a searcher from one of its vertices along all its edges and back to this vertex. At the end of this step, all edges of $Q_{k-4}^{(1,1)}$ are cleared.

(3) Slide a searcher from each vertex of $Q_{k-4}^{(4,1)}$ to its neighbor on $Q_{k-4}^{(1,1)}$, and then slide further to a new neighbor on $Q_{k-4}^{(2,1)}$. At the end of this step, each vertex of $Q_{k-4}^{(1,1)}, Q_{k-4}^{(2,1)}$ and $Q_{k-4}^{(4,1)}$ contains two searchers.

(4) Slide a searcher from each vertex of $Q_{k-4}^{(1,1)}, Q_{k-4}^{(2,1)}$ and $Q_{k-4}^{(4,1)}$ to its neighbor on $Q_{k-2}^{(2)}$ along the edge between them.

(5) Slide a searcher from a vertex of $Q_{k-4}^{(3,2)}$ along all the edges of $Q_{k-2}^{(2)}$ and back to this vertex. Then slide a searcher from each vertex of $Q_{k-4}^{(3,2)}$ to its neighbor on $Q_{k-4}^{(3,1)}$ along the edge between them.

(6) After Step (5), each vertex of $Q_{k-2}^{(1)}$ and $Q_{k-2}^{(2)}$ contains exactly one searcher. Since all edges of $Q_{k-2}^{(2)}$ are cleared, slide a searcher from each vertex of $Q_{k-2}^{(2)}$ to its neighbor on $Q_{k-2}^{(3)}$ along the edge between them.

(7) First slide the additional searcher placed in Step (1) along all edges of $Q_{k-2}^{(3)}$ to clear them, and then slide a searcher from each vertex of $Q_{k-2}^{(3)}$ to its neighbor on $Q_{k-2}^{(4)}$ along the edge between them.

(8) After Step (7), each vertex of $Q_{k-2}^{(1)}$ and $Q_{k-2}^{(4)}$ contains exactly one searcher. From Steps (2) and (3), we know that each vertex of $Q_{k-4}^{(1,1)}$ has only one contaminated edge incident on it. So, slide a searcher from each vertex of $Q_{k-4}^{(1,1)}$ to its neighbor on $Q_{k-4}^{(1,4)}$ along the edge between them.
Figure 3.9: $Q_k = Q_{k-4} \square Q_4$: the vertex $Q(i,j)$, $1 \leq i, j \leq 4$, represents $Q_{k-4}^{(i,j)}$.

(9) For $Q_{k-2}^{(4)}$, slide a searcher from one of the vertices on $Q_{k-4}^{(1,4)}$ along all edges of $Q_{k-2}^{(4)}$. Then slide a searcher from each vertex of $Q_{k-4}^{(i,4)}$, $2 \leq i \leq 4$, to its neighbor on $Q_{k-2}^{(1)}$ along the edge between them.

(10) For each of $Q_{k-4}^{(i,1)}$, $2 \leq i \leq 4$, slide one searcher from one of its vertices along all its edges and back to this vertex. At the end of this step, all edges of $Q_{k-4}^{(i,1)}$, $2 \leq i \leq 4$, are cleared.

(11) Slide a searcher from each vertex of $Q_{k-4}^{(2,1)}$ to its neighbor on $Q_{k-4}^{(3,1)}$ and then slide further to a new neighbor on $Q_{k-4}^{(4,1)}$. At the end of this step, every edge of $Q_k$ is cleared.

From Step (1), we know that $3 \cdot 2^{k-3}$ searchers are placed on $Q_{k-2}^{(1)}$, $2^{k-3}$ searchers are placed on $Q_{k-2}^{(2)}$, and one additional searcher is placed on $Q_{k-2}^{(3)}$. In total, $2^{k-1} + 1$ searchers are placed on $Q_k$. Therefore, $fs(Q_k) \leq 2^{k-1} + 1$.  

Applying the same idea as that used in the proof of Theorem 3.2.5, we can show the following.
Corollary 3.2.6 For $m \geq 3$, $\text{fs}(B_m \Box Q_4) \leq 8m + 1$.

Similar to Corollary 3.2.3, we can extend Corollary 3.2.6 to a broader class of graphs.

Corollary 3.2.7 Let $Q_{m,n}$ be a graph obtained from $B_m \Box Q_4$ ($m \geq 3$) by replacing each copy of $B_m$ by an arbitrary Eulerian graph with $m$ vertices. Then $\text{fs}(Q_{m,n}) \leq 8m + 1$.

Recall that a graph is even if every vertex has an even degree, and it is odd if every vertex has an odd degree.

Corollary 3.2.8 Let $n \geq 6$ and $H$ be a graph with $m$ vertices. If $n$ and $H$ are even, or $n$ and $H$ are odd, then $\text{fs}(H \Box Q_n) \leq m2^{n-1} + 1$. If one of $n$ and $H$ is even and the other is odd, then $\text{fs}(H \Box Q_n) = m2^{n-1} + 2$.

Proof. If $n$ is even and $H$ is an even graph, then $H \Box Q_{n-4}$ is Eulerian since $n \geq 6$. Similarly, if $n$ is odd and $H$ is an odd graph, then $H \Box Q_{n-4}$ is also Eulerian. Thus, from Corollary 3.2.6, we have $\text{fs}(H \Box Q_n) = \text{fs}(H \Box Q_{n-4} \Box Q_4) \leq 8m2^{n-4} + 1 = m2^{n-1} + 1$.

If one of $n$ and $H$ is even and the other is odd, then $H \Box Q_{n-1}$ is Eulerian. So, it follows from Theorem 3.1.14 that $\text{fs}(H \Box Q_n) = \text{fs}(H \Box Q_{n-1} \Box Q_1) = m2^{n-1} + 2$. 

Before giving a lower bound for $\text{fs}(Q_k)$ when $k$ is even, we first consider the treewidth of $Q_k$. Given a graph $G = (V, E)$, a tree decomposition of $G$ is a pair $(T, W)$ with a tree $T = (I, F)$, $I = \{1, 2, \ldots, m\}$, and a family of non-empty
subsets $W = \{W_i \subseteq V : i = 1, 2, \ldots, m\}$, satisfying that

1. $\bigcup_{i=1}^{m} W_i = V$,

2. for each $uv \in E$, there is an $i \in I$ with $\{u, v\} \subseteq W_i$, and

3. for all $i, j, k \in I$, if $j$ is on the path from $i$ to $k$ in $T$, then $W_i \cap W_k \subseteq W_j$.

The width of $(T, W)$ is $\max\{|W_i| - 1 : 1 \leq i \leq m\}$. The treewidth of $G$, denoted by $\text{tw}(G)$, is the minimum width over all tree decompositions of $G$. A tree decomposition $(T, W)$ is a path decomposition if $T$ is a path; the path-width of a graph $G$, denoted by $\text{pw}(G)$, is the minimum width over all path decompositions of $G$.

From [17], we have the following lower bound for $\text{tw}(Q_k)$.

**Lemma 3.2.9** ([17]) *There is a constant $k_0$ such that, for any $k \geq k_0$, $\text{tw}(Q_k) \geq \frac{12.2^k}{25\sqrt{k}}$.*

Since $\text{fs}(G) \geq \text{pw}(G) \geq \text{tw}(G)$, we have the following result (we rewrite the lower bound in a power of 2 so that it can be easily compared with the upper bound of $\text{fs}(Q_k)$ when $k$ is even).

**Corollary 3.2.10** *There is a constant $k_0$ such that $\text{fs}(Q_k) \geq \frac{3}{25} 2^{k+2-\log \sqrt{k}}$ for any $k \geq k_0$.***
Chapter 4

Fast Searching on Complete $k$-partite Graphs

In this chapter, we investigate the fast search number of complete $k$-partite graphs, as well as their special cases including complete bipartite graphs and complete split graphs.

4.1 Lower Bounds and Algorithms for Complete $k$-partite Graphs

First of all, we give two lower bounds on the fast search number of complete $k$-partite graphs, then we present algorithms for clearing complete $k$-partite graphs. We will use $\langle V_p, S \rangle$ to denote the output of algorithms in this section, where $V_p$ is a multiset of vertices on which we place searchers, and $S$ is a
sequence of sliding actions. The size of $V_p$ gives an upper bound on the fast search number of $K_{n_1,\ldots,n_k}$.

**Lemma 4.1.1** For a complete $k$-partite graph $K_{n_1,\ldots,n_k}$, where $k \geq 2$ and $n_1 \leq \cdots \leq n_k$, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq \sum_{i=1}^{k-1} n_i$.

**Proof.** Consider an optimal fast search strategy $S_K$ for clearing $K_{n_1,\ldots,n_k}$. Let $w_1 \in V_j$, where $1 \leq j \leq k$, denote the first cleared vertex in $S_K$. It is easy to see that at the moment when $w_1$ is cleared, every vertex of $K_{n_1,\ldots,n_k} - V_j$ must contain a searcher. Therefore, $\text{fs}(K_{n_1,\ldots,n_k}) \geq \sum_{i=1}^{k-1} n_i - n_j \geq \sum_{i=1}^{k-1} n_i$.

**Lemma 4.1.2** For a complete $k$-partite graph $K_{n_1,\ldots,n_k}$, where $k \geq 3$ and $n_1 \leq \cdots \leq n_k$, if $\sum_{i=1}^{k-1} n_i \geq 3$ and $n_k \geq 3$, then $\text{fs}(K_{n_1,\ldots,n_k}) \geq 2 + \sum_{i=1}^{k-1} n_i$.

**Proof.** For any graph $G$, $\text{fs}(G)$ is greater than or equal to the edge search number of $G$. Thus, it follows from Theorem 6 in [3] that $\text{fs}(K_{n_1,\ldots,n_k}) \geq 2 + \sum_{i=1}^{k-1} n_i$.

**Theorem 4.1.3** For a complete $k$-partite graph $K_{n_1,\ldots,n_k}$, where $k \geq 3$, $n_1 \leq \cdots \leq n_k$ and $\sum_{i=1}^{k} n_i = n$, if $\sum_{i=1}^{k-1} n_i \geq n_k = 3$, then $\text{fs}(K_{n_1,\ldots,n_k}) = n - 1$.

**Proof.** From Lemma 4.1.2, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq n - n_k + 2 = n - 1$. To prove $\text{fs}(K_{n_1,\ldots,n_k}) = n - 1$, we only need to show that $\text{fs}(K_{n_1,\ldots,n_k}) \leq n - 1$. The algorithm SEARCHKPARTITE1 (see Algorithm 1) produces a fast search strategy that uses exactly $n - 1$ searchers to clear $K_{n_1,\ldots,n_k}$. The input of the algorithm is a complete $k$-partite graph $K_{n_1,\ldots,n_k}$, where $k \geq 3$ and $\sum_{i=1}^{k-1} n_i \geq $
Algorithm 1 SearchKPartite1($K_{n_1,...,n_k}$)

Input: A complete $k$-partite graph $K_{n_1,...,n_k}$ satisfying the above conditions.

Output: $(V_p, S)$.

1: Let $V_k = \{v_1, v_2, v_3\}$ and $X = K_{n_1,...,n_k} - V_k$. Place $n - 3$ searchers on $v_1$
   and slide them to each vertex of $X$.

2: if $X$ is Eulerian then

3:   place one searcher on a vertex $u$ of $X$. Slide one of the two searchers
    on $u$ along the Eulerian circuit of $X$ to clear all its edges. Slide the
    two searchers on $u$ to $v_2$ and $v_3$ respectively. Place one searcher on $v_2$.

4:   if $\deg_Y(v_2)$ is even then

5:     slide one of the two searchers on $v_2$ along the Eulerian circuit of $Y$
       to clear all its edges.

6:   else if $\deg_Y(v_2)$ is odd then

7:     slide one of the two searchers on $v_2$ to $v_3$ along the Eulerian trail of
       $Y$ to clear all its edges.

8:   end if

9: else if $X$ is odd then

10:   place two searchers on $v_2$. Slide one of the two searchers on $v_2$ along
    the Eulerian circuit of $X + \{v_2\}$ to clear all its edges. Slide all searchers
    on $X$ to $v_3$ to clear all the remaining contaminated edges of $K_{n_1,...,n_k}$.

55
11: else

12: if $k = 3$ and $X$ contains only one even vertex, say $u$ then

13: let $\{P_1\}$ be the output of $\text{SelectPath}(X)$, and let $a_1$ and $b_1$ be the two endpoints of $P_1$. Place one searcher on $a_1$ and $v_2$ respectively. Slide a searcher from $a_1$ to $b_1$ along $P_1$. Slide the two searchers on $b_1$ to $v_2$ and $v_3$ respectively. Slide a searcher on $v_2$ along the path $v_2uv_3a_1v_2$. $\triangleright$ Note that $V_1 = \{u\}$ and $V_2 = \{a_1, b_1\}$ in this case.

14: else if $k > 3$ or $X$ contains at least two even vertices then

15: Let $\mathcal{P}$ be a set returned by $\text{SelectPath}(X)$. Let $G_\mathcal{P}$ be the graph formed by all paths in $\mathcal{P}$. Let $h = \frac{|\text{odd}(X)|}{2}$. Let $H_0 = X$. For $i$ from 1 to $h$, let $a_i$ and $b_i$ be the two endpoints of $P_i$, and let $H_i = H_{i-1} - E(P_i)$.

16: Let $u$ be an even vertex of $X$ satisfying that $u$ is not included in any of $P_1, \ldots, P_h$. Let $U$ be a connected component in $H_h$ that contains $u$.

17: Place one searcher on the vertex $u$. Slide one of the two searchers on $u$ along the Eulerian circuit of $U$ to clear all its edges. Slide the two searchers on $u$ to $v_2$ and $v_3$ respectively. Place one searcher on $v_2$. Let $H$ be the graph formed by all the remaining contaminated edges of $K_{n_1, \ldots, n_k}$ except edges in $\bigcup_{i=1}^h E(P_i)$.

18: if $\deg_H(v_2)$ is even, then

19: slide one of the two searchers on $v_2$ along the Eulerian circuit of $H$ to clear all its edges.
else if $\deg_H(v_2)$ is odd, then

slide one of the two searchers on $v_2$ from $v_2$ to $v_3$ along the Eulerian trail of $H$ to clear all its edges.

end if

For $i$ from $h$ down to 1, slide the searcher on $a_i$ along $P_i$ to $b_i$.

end if

end if

Output the multiset $V_p$ of vertices on which searchers are placed and output the sequence $S$ of sliding actions.

Note that $X$ is a complete $(k - 1)$-partite graph in SearchKPartite1. The function $\text{SelectPath}(X)$ finds a set of paths of $X$ that has both even and odd vertices, which satisfy that: (1) the two endpoints of each path are in $V_{\text{odd}}(X)$, and (2) each path does not contain any vertex in $V_{\text{odd}}(X)$ as an internal vertex.

1: function $\text{SelectPath}(X)$

2: Let $h = \frac{|V_{\text{odd}}(X)|}{2}$.

3: if $X$ contains at least two even vertices then

4: let $u$ and $u'$ denote two even vertices of $X$. Let $V_{\text{odd}}(X) = \{a_1, \ldots, a_h, b_1, \ldots, b_h\}$. For $i$ from 1 to $h$, let $P_i = a_iu'b_i$.

5: else if $X$ contains only one even vertex then

6: let $u$ denote the even vertex of $X$. ▷ Note that $V_1 = \{u\}$. 

57
Consider SelectPath($X$). If $X$ contains at least two even vertices, say $u$ and $u'$, then $u'$ is adjacent to all odd vertices of $X$, and we let $P_i = a_i u' b_i$. If $X$ contains only one even vertex, say $u$, then we first show that $V_1 = \{u\}$. Note that all vertices in $V_j$, $1 \leq j \leq k - 1$, have the same degree in $X$. Therefore, we know $|V_1| = 1$ and $u$ is the only vertex in $V_1$. Further, if there is a vertex set $V_j$, $2 \leq j \leq k - 1$, which contains three vertices, then each of the three vertices is even in $X$. This is a contradiction. Hence, $|V_j| = 2$ for all $2 \leq j \leq k - 1$. We have two cases for $k$: (1) If $k > 3$, then we can find a matching for all odd vertices of $X$. Note that there are $2k - 4$ odd vertices on $X$. Let $V_2 = \{a_1, b_{k-2}\}$ and $V_j = \{a_{j-1}, b_{j-2}\}$, where $3 \leq j \leq k - 1$. For $i = 1, \ldots, k - 2$, it is easy to see that $a_i$ is adjacent to $b_i$. Hence, we can let $P_i = a_i b_i$. Clearly, $u$ is not included in $P_i$. (2) If $k = 3$, then we have $|V_1| = 1$, $|V_2| = 2$ and $|V_3| = 3$. Further, $a_1$ and $b_1$ are the only two odd vertices of $X$. 

------

7: \textbf{if} $k > 3$ \textbf{then}
8: \quad \text{let} $V_2 = \{a_1, b_{k-2}\}$ and $V_j = \{a_{j-1}, b_{j-2}\}$, $3 \leq j \leq k - 1$. For $i$ from 1 to $k - 2$, let $P_i = a_i b_i$.
9: \quad \textbf{else if} $k = 3$ \textbf{then}
10: \quad \quad \text{let} $V_2 = \{a_1, b_1\}$ and $P_1 = a_1 u b_1$. \quad \triangleright \text{Note that} \ h = 1.$
11: \quad \textbf{end if}
12: \textbf{end if}
13: \quad \text{Let} $P = \{P_1, \ldots, P_h\}$.
14: \quad \text{return} $P$.
15: \textbf{end function}

58
Let $V(X) = \{u, a_1, b_1\}$ and $P_1 = a_1ub_1$. 

**Theorem 4.1.4** For a complete $k$-partite graph $K_{n_1, \ldots, n_k}$, if there is an $n_j$, $1 \leq j \leq k$, such that $\sum_{i=1}^{k} n_i - n_j \geq 4$ and $\sum_{i=1}^{k} n_i - n_j$ is even, then $fs(K_{n_1, \ldots, n_k}) \leq \sum_{i=1}^{k} n_i - n_j + 3$.

**Proof.** If $n_j \leq 3$, from Theorem 5.1 in [62], we see that the claim holds. If $k = 2$ and $\sum_{i=1}^{k} n_i - n_j \geq 6$, from Lemma 5 in [22], we know that the claim holds. If $k = 2$ and $\sum_{i=1}^{k} n_i - n_j = 4$, similar to Lemma 5 in [22], we can show that the claim also holds. So we assume that $n_j \geq 4$ and $k \geq 3$ in the rest of the proof.

We first present function $\text{SelectPath2}(X)$. For a complete $k$-partite graph $K_{n_1, \ldots, n_k}$, let $V(X) = V(K_{n_1, \ldots, n_k}) \setminus V_j$, where $1 \leq j \leq k$. $\text{SelectPath2}(X)$ will find a set of paths of $X$, satisfying that the two endpoints of each path are in $V_{\text{odd}}(X)$. Comparing with $\text{SelectPath2}(X)$, in $\text{SelectPath}(X)$: $X$ has to satisfy that $V(X) = V(K_{n_1, \ldots, n_k}) \setminus V_k$ where $V_k = 3$; further, the paths returned by $\text{SelectPath}(X)$ do not contain any vertex in $V_{\text{odd}}(X)$ as an internal vertex.

---

1: function $\text{SelectPath2}(X)$

2: if $X$ is Eulerian then

3: return $\mathcal{P} = \emptyset$.

4: end if

5: Let $h = \frac{|V_{\text{odd}}(X)|}{2}$.

59
if \(X\) contains at least one even vertex then

let \(V_{\text{odd}}(X) = \{a_1, \ldots, a_h, b_1, \ldots, b_h\}\), and let \(u\) be an even vertex of \(X\). Let \(P_i = a_iub_i\), for \(i = 1, \ldots, h\).

else if \(X\) contains no even vertex then

let \(V_{j_1} = \{v_1, \ldots, v_{n_{j_1}}\}\) and \(V_{j_2} = \{u_1, \ldots, u_{n_{j_2}}\}\), where \(1 \leq j_1 < j_2 \leq k\). Let \(P_1 = v_1u_1\). Let \(V'_{j_1} = V_{j_1} \setminus \{v_1\}\), and let \(i = 2\).

while \(V'_{j_1}\) is not empty do

Let \(a_i\) and \(b_i\) be two vertices in \(V'_{j_1}\). Let \(P_i = a_iu_1b_i\).

Let \(V'_{j_1} \leftarrow V'_{j_1} \setminus \{a_i, b_i\}\), and \(i \leftarrow i + 1\).

end while

Let \(V'_{j_2} = V(X) \setminus \{V_{j_1} \cup \{u_1\}\}\).

while \(V'_{j_2}\) is not empty do

Let \(a_i\) and \(b_i\) be two vertices in \(V'_{j_2}\). Let \(P_i = a_iv_1b_i\).

Let \(V'_{j_2} \leftarrow V'_{j_2} \setminus \{a_i, b_i\}\), and \(i \leftarrow i + 1\).

end while

end if

Let \(\mathcal{P} = \{P_1, \ldots, P_h\}\).

return \(\mathcal{P}\).

end function

Consider \(\text{SelectPath2}(X)\). We now show that if \(X\) is odd, then \(|V_i|\) is odd for all \(1 \leq i \leq k\) and \(i \neq j\). For the sake of contradiction, assume that there is a vertex set \(V_q\) of \(X\), where \(1 \leq q \leq k\) and \(q \neq j\), satisfying that \(|V_q|\) is even. Note that every vertex in \(V_q\) is odd. Hence, \(|V(X) \setminus V_q|\) is odd. Since
$|V_q|$ is even, we know that $|V(X)|$ is odd. Therefore, there must be a vertex set $V_r$ of $X$, where $1 \leq r \leq k$ and $r \neq j$, satisfying that $|V_r|$ is odd. Since $|V(X) \setminus V_r|$ is even, we know that every vertex in $V_r$ is even. This contradicts that every vertex of $X$ is odd. Therefore, $|V_i|$ is odd for all $1 \leq i \leq k$ and $i \neq j$.

The algorithm SEARCHKPARTITE2 (see Algorithm 2) produces a fast search strategy for clearing $K_{n_1, \ldots, n_k}$ using $\sum_{i=1}^{k} n_i - n_j + 3$ searchers. The input of the algorithm includes a complete $k$-partite graph where $\sum_{i=1}^{k} n_i - n_j \geq 4$ and $\sum_{i=1}^{k} n_i - n_j$ is even, and an integer $j$ that corresponds to the index of vertex set $V_j$.

Algorithm 2 SEARCHKPARTITE2($K_{n_1, \ldots, n_k}, j$)

**Input:** A complete $k$-partite graph $K_{n_1, \ldots, n_k}$ and an integer $j$, both of which satisfy the above conditions.

**Output:** $\langle V_p, S \rangle$.

1. Let $V_j = \{v_1, v_2, \ldots, v_{n_j}\}$ and $X = K_{n_1, \ldots, n_k} - V_j$. Let $\sum_{i=1}^{k} n_i - n_j = m$ and $V(X) = \{u_1, u_2, \ldots, u_m\}$.
2. if $n_j$ is odd then
3. place $m$ searchers on $v_{n_j}$, and slide them to each vertex of $X$.
4. else if $n_j$ is even then
5. place $m$ searchers on each vertex of $X$.
6. end if
7. Place one searcher on $u_1$, $u_2$ and $u_3$ respectively. Let $\mathcal{P}$ be a set returned by SELECTPATH2($X$).
8: Let $H_0 = X$. If $\mathcal{P} \neq \emptyset$, then let $G_\mathcal{P}$ be the graph formed by all paths in $\mathcal{P}$; for $i$ from 1 to $h$, let $a_i$ and $b_i$ be the two endpoints of $P_i$, and let $H_i = H_{i-1} - E(P_i)$.

9: In Steps 11-15, at any moment when a vertex $u_i$ ($1 \leq i \leq m$) contains two searchers, if $H_h$ has a connected component that contains $u_i$ and no edges of the component are cleared, then slide a searcher from $u_i$ along the Eulerian circuit of the component to clear all its edges.

10: for $r$ from 0 to $\left\lfloor \frac{n_j}{2} \right\rfloor - 1$ do

11: Slide a searcher from $u_1$ to $v_{2r+1}$ along $u_1v_{2r+1}$. Slide a searcher from $u_2$ to $v_{2r+1}$ along $u_2v_{2r+1}$. Slide a searcher from $u_3$ to $v_{2r+2}$ along $u_3v_{2r+2}$.

12: Let $Y$ denote the subgraph formed by all the edges with one point in \{u_4, \ldots, u_m\} and the other endpoint in \{v_1, v_2\}. Slide a searcher from $v_1$ to $v_2$ along the Eulerian trail of $Y$ to clear all its edges.

13: Slide a searcher from $v_{2r+1}$ to $u_3$ along $v_{2r+1}u_3$. Slide a searcher from $v_{2r+2}$ to $u_1$ along $v_{2r+2}u_1$. Slide a searcher from $v_{2r+2}$ to $u_2$ along $v_{2r+2}u_2$.

14: end for

15: If $G_\mathcal{P}$ is not empty, then for $i$ from $h$ down to 1, slide the searcher on $a_i$ along $P_i$ to $b_i$.

16: Output the multiset $V_\mathcal{P}$ of vertices on which searchers are placed and output the sequence $S$ of sliding actions.
Theorem 4.1.5  For a complete $k$-partite graph $K_{n_1, \ldots, n_k}$, if there is an $n_j$, $1 \leq j \leq k$, such that $\sum_{i=1}^{k} n_i - n_j \geq 3$ and $\sum_{i=1}^{k} n_i - n_j$ is odd, then $fs(K_{n_1, \ldots, n_k}) \leq \sum_{i=1}^{k} n_i - \left\lfloor \frac{n_j}{2} \right\rfloor$.

Proof.  If $n_j \leq 3$, similar to Theorem 5.1 in [62], we can prove the claim. If $k = 2$, from Lemma 7 in [22], we see that the claim holds. So we assume that $n_j \geq 4$ and $k \geq 3$ in the remainder of the proof. The algorithm SEARCHKPARTITE3 (see Algorithm 3) produces a fast search strategy for clearing $K_{n_1, \ldots, n_k}$ using $\sum_{i=1}^{k} n_i - \left\lfloor \frac{n_j}{2} \right\rfloor$ searchers. The input of the algorithm is $K_{n_1, \ldots, n_k}$ and $j$, where $\sum_{i=1}^{k} n_i - n_j \geq 3$, $\sum_{i=1}^{k} n_i - n_j$ is odd, and $j$ is the index of $V_j$.

Algorithm 3 SEARCHKPARTITE3($K_{n_1, \ldots, n_k}, j$)

Input: A complete $k$-partite graph $K_{n_1, \ldots, n_k}$ and an integer $j$, both of which satisfy the above conditions.

Output: $\langle V_p, S \rangle$.

1: Let $V_j = \{v_1, v_2, \ldots, v_{n_j}\}$ and $X = K_{n_1, \ldots, n_k} - V_j$. Let $\sum_{i=1}^{k} n_i - n_j = m$, $h = \left\lfloor \frac{|V_{odd}(X)|}{2} \right\rfloor$, and $V(X) = \{u_1, u_2, \ldots, u_m\}$. Let $\mathcal{P}$ be a set returned by SELECTPATH2($X$).

2: Let $H_0 = X$. If $\mathcal{P} \neq \emptyset$, then let $G_\mathcal{P}$ be the graph formed by all paths in $\mathcal{P}$; for $i$ from 1 to $h$, let $a_i$ and $b_i$ be the two endpoints of $P_i$, and let $H_i = H_{i-1} - E(P_i)$. Let $\ell = \left\lfloor \frac{n_j}{4} \right\rfloor$.

3: if $n_j = 4\ell + 1$ then

4: place $m$ searchers on $v_1$, place one searcher on each of $u_1$, $v_2$ and $v_3$.

Place one searcher on each of $v_{4i+2}$ and $v_{4i+3}$ for $i = 1, \ldots, \ell - 1$.

5: Call CLEARCROSSEDGES($X, V_j, H_h$). If $\mathcal{P} \neq \emptyset$, for $i$ from $h$ down to 1, slide the searcher on $a_i$ along $P_i$ to $b_i$. 

63
6: else if $n_j = 4\ell + 2$ then
7: If $\mathcal{P} \neq \emptyset$ and $u_1 \neq a_h$, then let $u' = a_h$, and switch the labels of vertices $u_1$ and $u'$ such that $u_1 = a_h$.
8: Place $m$ searchers on $v_1$, place one searcher on each of $u_1$, $v_2$ and $v_3$. Place one searcher on each of $v_{4i+2}$ and $v_{4i+3}$ for $i = 1, \ldots, \ell - 1$.
9: Call $\text{ClearCrossEdges}(X, V_j, H_h)$. For each vertex $u \in V(X)$ whose $\deg_X(u)$ is even, if $u \notin V(G_P)$, then slide a searcher on $u$ to $v_{n_j}$ along $uv_{n_j}$.
10: if $\mathcal{P} \neq \emptyset$ then
11: slide a searcher from $u_1$ to $v_{n_j}$ along $u_1v_{n_j}$. Slide the other searcher on $u_1$ along $P_h$ to $b_h$, during which, when a vertex $u_i$ of $P_h$ has only one contaminated edge incident on it, slide a searcher on $u_i$ along $u_iv_{n_j}$ to $v_{n_j}$.
12: If $h \geq 2$, then for $r$ from $h - 1$ down to 1, slide a searcher from $v_{n_j}$ to $a_r$ along $v_{n_j}a_r$; slide this searcher along $P_r$ to $b_r$, during which, when a vertex $u_i$ of $P_r$ has only one contaminated edge incident on it, slide a searcher on $u_i$ along $u_iv_{n_j}$ to $v_{n_j}$.
13: end if
14: else if $n_j = 4\ell + 3$ then
15: If $\mathcal{P} \neq \emptyset$ and $u_m \neq a_h$, then let $u' = a_h$, and switch the labels of vertices $u_m$ and $u'$ such that $u_m = a_h$. If $\mathcal{P} \neq \emptyset$ and $u_1 = b_h$, then switch the labels of vertices $u_1$ and $u_2$ such that $u_1 \neq b_h$. 
16: Place \( m \) searchers on \( v_1 \), place one searcher on each of \( u_1, v_2 \) and \( v_3 \).

Place one searcher on each of \( v_{4i+2} \) and \( v_{4i+3} \) for \( i = 1, \ldots, \ell - 1 \). Place another searcher on \( u_m \).

17: Call \textsc{ClearCrossEdges}(X,V_j,H_h).

18: \textbf{if} \( P = \emptyset \) \textbf{then}

19: slide one of the two searchers on \( u_m \) along \( u_m v_{n_j-1} \) and \( u_m v_{n_j} \) respectively. Slide a searcher on \( u_1 \) to clear the Eulerian circuit formed by all the edges with one endpoint in \( V(X) \setminus \{u_m\} \) and the other endpoint in \( \{v_{n_j-1}, v_{n_j}\} \).

20: \textbf{else if} \( P \neq \emptyset \) \textbf{then}

21: slide one of the two searchers on \( u_m \) along \( P_h \) to \( b_h \) to clear all its edges. Slide a searcher on \( b_h \) along \( b_h v_{n_j-1} \) and \( b_h v_{n_j} \) respectively. Slide a searcher on \( u_1 \) to clear the Eulerian circuit formed by all the edges with one endpoint in \( V(X) \setminus \{b_h\} \) and the other endpoint in \( \{v_{n_j-1}, v_{n_j}\} \). If \( h \geq 2 \), then for \( r \) from \( h - 1 \) down to 1, slide the searcher on \( a_r \) along \( P_r \) to \( b_r \).

22: \textbf{end if}

23: \textbf{else if} \( n_j = 4\ell \) \textbf{then}

24: If \( P \neq \emptyset \) and \( u_1 \neq a_h \), then let \( u' = a_h \), and switch the label of \( u_1 \) and \( u' \) such that \( u_1 = a_h \). If \( P \neq \emptyset \) and \( u_m = b_h \), then switch the labels of vertices \( u_m \) and \( u_2 \) such that \( u_m \neq b_h \).

25: Place a searcher on every vertex in \( \{u_1, u_2, \ldots, u_{m-1}, v_1, v_2, \ldots, v_{2\ell}\} \) and place a second searcher on \( u_1 \).
Slide the searcher from $u_1$ along the Eulerian circuit formed by all the edges with one endpoint in \{u_1, u_2, \ldots, u_{m-1}\} and the other endpoint in \{v_1, v_2, \ldots, v_{2\ell}\}. Then slide each searcher on $v_i \in \{v_1, v_2, \ldots, v_{2\ell}\}$ along $v_i u_m$.

Slide a searcher on $u_m$ to each vertex in \{v_{2\ell+1}, v_{2\ell+2}, \ldots, v_{4\ell-2}\}. Slide a searcher on $u_1$ to clear the Eulerian circuit formed by all the edges with one endpoint in \{u_1, u_2, \ldots, u_{m-1}\} and the other endpoint in \{v_{2\ell+1}, v_{2\ell+2}, \ldots, v_{4\ell-2}\}.

In Steps 29-33, at any moment when a vertex $u_i$ ($1 \leq i \leq m$) contains two searchers, if $H_h$ has a connected component that contains $u_i$ and no edges of the component are cleared, then slide a searcher from $u_i$ along the Eulerian circuit of the component to clear all its edges.

if $\mathcal{P} = \emptyset$ then

slide a searcher on $u_1$ along $u_1 v_{4\ell-1}$ and $u_1 v_{4\ell}$ respectively. Then, slide a searcher on $u_m$ to clear the Eulerian circuit formed by all the edges with one endpoint in $V(X) \setminus \{u_1\}$ and the other endpoint in $\{v_{4\ell-1}, v_{4\ell}\}$.

else if $\mathcal{P} \neq \emptyset$ then

slide a searcher on $u_1$ to $b_h$ along $P_h$. Slide a searcher on $b_h$ along $b_h v_{4\ell-1}$ and $b_h v_{4\ell}$ respectively. Then, slide a searcher on $u_m$ to clear the Eulerian circuit formed by all the edges with one endpoint in $V(X) \setminus \{b_h\}$ and the other endpoint in $\{v_{4\ell-1}, v_{4\ell}\}$. If $h \geq 2$, then for $r$ from $h - 1$ down to 1, slide the searcher on $a_r$ along $P_r$ to $b_r$.

end if
34: end if
35: Output the multiset $V_p$ of vertices on which searchers are placed and output
the sequence $S$ of sliding actions.

The function $\text{ClearCrossEdges}(X,V_j,H_h)$ is called by $\text{SearchKPartite3}$ to clear the edges across $V(X)$ and $\{v_2,\ldots,v_{4\ell+1}\}$, where $\ell = \left\lfloor \frac{n_j}{4} \right\rfloor$.

1: function $\text{ClearCrossEdges}(X,V_j,H_h)$
2: In Steps 3-7, at any moment when a vertex $u_i$ ($1 \leq i \leq m$) contains
two searchers, if $H_h$ has a connected component that contains $u_i$ and
no edges of the component are cleared, then slide a searcher from $u_i$
along the Eulerian circuit of the component to clear all its edges.
3: Slide $m$ searchers from $v_1$ to each vertex of $X$. Let $\ell = \left\lfloor \frac{n_j}{4} \right\rfloor$.
4: for $r$ from 0 to $\ell - 1$ do
5: Slide one of the two searchers on $u_1$ along the Eulerian circuit
formed by all the edges with one endpoint in $\{u_1,u_2,\ldots,u_{m-1}\}$ and
the other endpoint in $\{v_{4r+2},v_{4r+3}\}$ to clear all its edges.
6: Slide a searcher from $v_{4r+2}$ to $v_{4r+4}$ along $v_{4r+2}u_mv_{4r+4}$ and slide a
searcher from $v_{4r+3}$ to $v_{4r+5}$ along $v_{4r+3}u_mv_{4r+5}$. Slide a searcher on
$u_1$ along the Eulerian circuit formed by all the edges with one
endpoint in $\{u_1,u_2,\ldots,u_{m-1}\}$ and the other endpoint in
$\{v_{4r+4},v_{4r+5}\}$.
7: end for
8: \textbf{return} the sequence of sliding actions.

9: \textbf{end function}

\textbf{Corollary 4.1.6} For a complete \(k\)-partite graph \(K_{n_1, \ldots, n_k}\), define \(\alpha_j\), \(1 \leq j \leq k\), as

\[
\alpha_j = \begin{cases} 
\sum_{i=1}^{k} n_i - n_j + 3, & \text{if } \sum_{i=1}^{k} n_i - n_j \text{ is even and } \sum_{i=1}^{k} n_i - n_j \geq 4, \\
\sum_{i=1}^{k} n_i - \left\lfloor \frac{n_j}{2} \right\rfloor, & \text{if } \sum_{i=1}^{k} n_i - n_j \text{ is odd and } \sum_{i=1}^{k} n_i - n_j \geq 3, \\
\sum_{i=1}^{k} n_i, & \text{else.}
\end{cases}
\]

Then \(fs(K_{n_1, \ldots, n_k}) \leq \min_{1 \leq j \leq k} \alpha_j\).

If \(k \geq 4\) or \(k = 3\) and \(n_{k-1} \geq 2\), then the algorithm \textsc{FastSearchKPartite} (see Algorithm 4) produces a fast search strategy for clearing a complete \(k\)-partite graph \(K_{n_1, \ldots, n_k}\).

\textbf{Algorithm 4 FastSearchKPartite}(\(K_{n_1, \ldots, n_k}\))

\textbf{Input:} A complete \(k\)-partite graph \(K_{n_1, \ldots, n_k}\) satisfying the above conditions.

\textbf{Output:} \(\langle V_p, S \rangle\).

1: Let \(j_1 \leftarrow \max\{j|\sum_{i=1}^{k} n_i - n_j \text{ is odd}\}\).
2: Let \(j_2 \leftarrow \max\{j|\sum_{i=1}^{k} n_i - n_j \text{ is even}\}\).
3: Let \(\langle V_p^1, S^1 \rangle\) be the output of \textsc{FastSearch}(\(K_{n_1, \ldots, n_k}\)) in [62]. Let \(V_p \leftarrow V_p^1\) and \(S \leftarrow S^1\).
4: if $n_{j_1} \geq 4$ then
5:     let $\langle V_p^2, S^2 \rangle$ be the output of \textsc{SearchKPartite3}($K_{n_1,\ldots,n_k,j_1}$).
6:     if $|V_p^2| < |V_p|$ then
7:         let $V_p \leftarrow V_p^2$ and $S \leftarrow S^2$.
8:     end if
9: end if
10: if $n_{j_2} \geq 4$ then
11:     let $\langle V_p^3, S^3 \rangle$ be the output of \textsc{SearchKPartite2}($K_{n_1,\ldots,n_k,j_2}$).
12:     if $|V_p^3| < |V_p|$ then
13:         let $V_p \leftarrow V_p^3$ and $S \leftarrow S^3$.
14:     end if
15: end if
16: Output the multiset $V_p$ of vertices on which searchers are placed and output
    the sequence $S$ of sliding actions.

\textbf{Theorem 4.1.7} For a complete $k$-partite graph $K_{n_1,\ldots,n_k}$ with $k \geq 4$ or $k = 3$
and $n_{k-1} \geq 2$, let $\beta$ be the number of searchers required by \textsc{FastSearchK-Partite}.
Then $fs(K_{n_1,\ldots,n_k}) \geq \beta - 3$.

Theorem 4.1.7 will be proved after Lemma 4.2.1 because we need some
idea and notation which will be used in the proof of Lemma 4.2.1.
4.2 Complete Bipartite Graphs $K_{m,n}$

In the following sections, we focus on some special classes of complete $k$-partite graphs. When $k = 2$, $K_{n_1,\ldots,n_k}$ is a complete bipartite graph. Dyer et al. [22] proved several results on the fast search number of $K_{m,n}$. The fast search problem on $K_{m,n}$ has been solved when $m$ is even. However, the fast search problem remains open when $m$ is odd, and they only gave lower and upper bounds on $fs(K_{m,n})$ in [22]:

- When $m$ is odd, $n$ is even and $3 \leq m \leq n$, we have $\max\{m + 2, \frac{n}{2}\} \leq fs(K_{m,n}) \leq \min\{n + 3, m + \frac{n}{2}\}$.
- When $m$ and $n$ are odd and $3 \leq m \leq n$, we have $\max\{m + 2, \frac{m+n}{2}\} \leq fs(K_{m,n}) \leq m + \frac{n+1}{2}$.

In the following, we will prove that for a complete bipartite graph $K_{m,n}$ with $3 \leq m \leq n$, if $m$ is odd, then $fs(K_{m,n})$ is equal to the upper bounds given above. Let $S_{K_{m,n}}$ denote an optimal fast search strategy for $K_{m,n}$, which uses the minimum number of sliding actions to clear the first cleared vertex of $K_{m,n}$ among all optimal fast search strategies for $K_{m,n}$. We use $w_1$ to denote the first cleared vertex of $K_{m,n}$. Let $t_1$ denote the moment at which $w_1$ is cleared (see Figure 4.1(1)). Note that vertices of $K_{m,n}$ are partitioned into two vertex sets $V_1$ and $V_2$. We use $w_2$ to denote the first cleared vertex in another vertex set of $K_{m,n}$ which does not contain $w_1$. That is, if $w_1 \in V_1$, then $w_2 \in V_2$; if $w_1 \in V_2$, then $w_2 \in V_1$. Let $t_2$ denote the moment after which the next sliding
action clears $w_2$ (see Figure 4.1(2)). Without loss of generality, we first assume that $w_1 \in V_2$. In a similar way, we can prove the lower bound on $fs(K_{m,n})$ when $w_1 \in V_1$.

Figure 4.1: (1) After searcher $\lambda$ slides from $v_1$ to $u_1$, $v_1$ becomes the first cleared vertex of $K_{3,3}$. Let this moment be denoted by $t_1$, and we have $w_1 = v_1$. (2) Searcher $\lambda$ will slide from $u_3$ to $v_3$ in the next step. After that, $u_3$ becomes the first cleared vertex in $V_1$. Let $t_2$ denote this moment, and we have $w_2 = u_3$.

Throughout this section, we assume $m$ is odd. We use $A_1$ to denote the set of all vertices in $V_2 \setminus \{w_1\}$ which contain a searcher at $t_1$ and have cleared incident edges at $t_2$. We use $A_2$ to denote the set of all vertices in $V_2 \setminus \{w_1\}$ which contain a searcher and have cleared incident edges at $t_2$. Let $a_1 = |A_1|$ and $a_2 = |A_2|$, it is easy to see that $a_1 + a_2 \geq |A_1 \cup A_2|$. Figures 4.2 and 4.3 illustrate $A_1$ and $A_2$ respectively.

Note that at the moment $t_1$, all vertices in $A_2 \setminus \{A_1 \cap A_2\}$ are contaminated and contain no searchers, and hence contain no searchers at the beginning of $S_{K_{m,n}}$ either. Since $m$ is odd, we know all vertices in $A_2$ are odd. Therefore, each vertex in $A_2 \setminus \{A_1 \cap A_2\}$ must contain a searcher at the end of $S_{K_{m,n}}$.

Lemma 4.2.1 For a complete bipartite graph $K_{m,n}$ with $m, n \geq 3$, let $S_{K_{m,n}}$ be an optimal fast search strategy for clearing $K_{m,n}$. If $w_1 \in V_2$ in $S_{K_{m,n}}$, then $a_1 + a_2 \geq |A_1 \cup A_2| \geq n - 2$. 

71
Figure 4.2: At the moment $t_1$, each vertex in $A_1$ contains a searcher. Further, each vertex in $A_1$ has cleared incident edges at $t_2$ (see Figure 4.3). In this case, $A_1 = \{v_2, v_3\}$.

Figure 4.3: At the moment $t_2$, each vertex in $A_2$ contains a searcher, and all vertices in $A_1$ and $A_2$ have cleared incident edges. In this case, $A_2 = \{v_4\}$.

**Proof.** Since $m$ is odd, each vertex in $V_2$ has odd degree, and therefore, every vertex in $V_2$ contains a searcher at the beginning or at the end of $S_{K_{m,n}}$. Note that the next sliding action after $t_2$ clears $w_2$. We know at the moment $t_2$, there is at most one vertex in $V_2$ which has no cleared incident edges, and all other vertices in $V_2$ have cleared incident edges. Let $v$ denote a vertex in $V_2 \setminus \{w_1\}$ which has cleared incident edges at $t_2$. Consider the moment $t_2$: (1) If $v$ is cleared and contains no searchers, then it must contain a searcher at the beginning of $S_{K_{m,n}}$, and hence we know $v \in A_1$; (2) if $v$ is cleared and contains searchers, then clearly $v \in A_2$; (3) if $v$ is contaminated, since $v$ has cleared incident edges at $t_2$, then $v \in A_2$. Therefore, we have $a_1 + a_2 \geq |A_1 \cup A_2| \geq n-2$. 

72
In the following, we give a proof of Theorem 4.1.7.

**Proof.** Let $V_p$ denote the multiset of vertices on which we place searchers in algorithm \textsc{FastSearchKPartite}. Hence, $|V_p| = \beta$ and we only need to show that $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq 3$. Let $S_K$ denote an optimal fast search strategy for clearing $K_{n_1,\ldots,n_k}$. Consider $S_K$. We use $w'_1 \in V_j$ to denote the first cleared vertex by $S_K$. Let $t'_1$ denote the moment at which $w'_1$ is cleared. Let $w'_2$ denote the first cleared vertex in the vertex set $K_{n_1,\ldots,n_k} \setminus V_j$. Let $t'_2$ denote the moment at which $w'_2$ is cleared. We use $A'_1$ to denote the set of all vertices in $V_j \setminus \{w'_1\}$ which contain a searcher at $t'_1$ and have cleared incident edges at $t'_2$. We use $A'_2$ to denote the set of all vertices in $V_j \setminus \{w'_1\}$ which contain a searcher and have cleared incident edges at $t'_2$. Similar to the proof of Lemma 4.2.1, we can show that $|A'_1 \cup A'_2| \geq n_j - 2$.

We have two cases regarding $n_j$:

**Case 1.** $n_j \geq 4$. Note that $1 \leq n_1 \leq \cdots \leq n_k$. If $k \geq 4$, then we have $\sum_{i=1}^k n_i - n_j \geq 3$; if $k = 3$ and $n_{k-1} \geq 2$, then we also have $\sum_{i=1}^k n_i - n_j \geq 3$. Consider the vertex set $V(K_{n_1,\ldots,n_k}) \setminus V_j$. We have two subcases.

**Case 1.1.** $\sum_{i=1}^k n_i - n_j$ is even. Note that $\sum_{i=1}^k n_i - n_j \geq 3$ and $j_2 = \max\{j | \sum_{i=1}^k n_i - n_j \text{ is even}\}$. Thus, we know $\sum_{i=1}^k n_i - n_{j_2} \geq 4$ and $n_{j_2} \geq n_j \geq 4$ in this subcase. Consider the moment $t'_1$. It is easy to see that every vertex of $K_{n_1,\ldots,n_k} \setminus V_j$ should be occupied by a searcher. Hence, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq \sum_{i=1}^k n_i - n_{j_2}$. By algorithm \textsc{FastSearchKPartite}, $|V_p| \leq \sum_{i=1}^k n_i - n_{j_2} + 3$. Therefore, $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq \sum_{i=1}^k n_i - n_{j_2} + 3 - (\sum_{i=1}^k n_i - n_j) = n_j - n_{j_2} + 3$.  

73
Recall that $n_j \leq n_{j2}$, we have $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq 3$.

Case 1.2. $\sum_{i=1}^k n_i - n_j$ is odd. Note that $j_1 = \max\{j | \sum_{i=1}^k n_i - n_j \text{ is odd}\}$. Thus, we know $n_j, n_j, \geq 4$ in this subcase. Note that $|A_1'| + |A_2'| \geq n_j - 2$.

Hence, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq \sum_{i=1}^k n_i - n_j + \left\lceil \frac{n_j - 2}{2} \right\rceil = \sum_{i=1}^k n_i - \left\lfloor \frac{n_j + 2}{2} \right\rfloor$.

By algorithm FastSearchKPartite, $|V_p| \leq \sum_{i=1}^k n_i - \left\lfloor \frac{n_j}{2} \right\rfloor$. Therefore, $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq \sum_{i=1}^k n_i - \left\lfloor \frac{n_j}{2} \right\rfloor - (\sum_{i=1}^k n_i - \left\lfloor \frac{n_j}{2} \right\rfloor - 1) = \left\lceil \frac{n_j}{2} \right\rceil - \left\lfloor \frac{n_j}{2} \right\rfloor + 1$.

Recall that $n_j \leq n_{j1}$, we have $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq 1$.

Case 2. $n_j \leq 3$. Then, we use algorithm FastSearch(K_{n_1,\ldots,n_k}), which was presented in [62], to clear $K_{n_1,\ldots,n_k}$ by using at most $|V(K_{n_1,\ldots,n_k})|$ searchers. Therefore, $|V_p| \leq |V(K_{n_1,\ldots,n_k})|$. Note that at the moment $t'_1$, every vertex of $K_{n_1,\ldots,n_k} - V_j$ should be occupied by a searcher. Hence, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq \sum_{i=1}^k n_i - n_j$. Thus, $|V_p| - \text{fs}(K_{n_1,\ldots,n_k}) \leq \sum_{i=1}^k n_i - (\sum_{i=1}^k n_i - n_j) = n_j \leq 3$.

From the above cases, we have $\text{fs}(K_{n_1,\ldots,n_k}) \geq |V_p| - 3$. \hfill \[ 

**Lemma 4.2.2** For a complete bipartite graph $K_{m,n}$ with $m, n \geq 3$, let $S_{K_{m,n}}$ be an optimal fast search strategy for clearing $K_{m,n}$. Suppose that $w_1 \in V_2$ in $S_{K_{m,n}}$. If (1) each vertex in $V_1 \cup A_1$ contains exactly one searcher at $t_1$, and (2) $w_1$ contains no searchers at $t_1$, then each vertex in $A_1$ has at least two contaminated incident edges at $t_1$.

**Proof.** Let $t_0$ denote the moment in $S_{K_{m,n}}$ at which $w_1$ has exactly two contaminated incident edges, and one of the contaminated edges is cleared by the next sliding action after $t_0$. Since $w_1$ has degree at least 3, $w_1$ must be occupied by a searcher at $t_0$. If one of the last two contaminated edges
incident on $w_1$ is cleared by sliding a searcher from a vertex in $V_1$ to $w_1$, then $w_1$ contains at least one searcher at $t_1$, which contradicts the premise that no searchers are located on $w_1$ at $t_1$. Hence, both of the last two contaminated edges incident on $w_1$ are cleared by sliding a searcher from $w_1$ to $V_1$. Further, since $w_1$ contains no searchers at $t_1$, $w_1$ must contain exactly two searchers at $t_0$.

Consider the moment $t_0$. Then assume that there is a vertex $x_1 \in A_1$ which has exactly one contaminated incident edge. Hence, we know $x_1$ must contain a searcher, say $\lambda$, at the moment $t_0$. Let $x_1x_2$ denote the last cleared edge incident on $x_1$. In the following, we show that there exists a fast search strategy that uses fewer sliding actions than $S_{K_{m,n}}$ to clear the first cleared vertex of $K_{m,n}$. Let $S'_{K_{m,n}}$ denote a fast search strategy obtained from $S_{K_{m,n}}$ by making the following modifications:

1. Delete the sliding action on $x_1x_2$ from $S_{K_{m,n}}$.
2. Just after $t_0$, immediately insert a new sliding action by letting $\lambda$ slide from $x_1$ to $x_2$.

Clearly, $S'_{K_{m,n}}$ uses fewer sliding actions than $S_{K_{m,n}}$ to clear the first cleared vertex of $K_{m,n}$. This contradicts that $S_{K_{m,n}}$ uses the minimum number of sliding actions to clear the first cleared vertex among all optimal fast search strategies for $K_{m,n}$. Hence, each vertex in $A_1$ has at least two contaminated incident edges at $t_0$. Note that $w_1$ contains two searchers at $t_0$ and the last two contaminated edges incident on $w_1$ are both cleared by sliding a searcher.
from $w_1$ to $V_1$. Thus, $w_1$ has to be cleared in exactly two steps after $t_0$. Hence, each vertex in $A_1$ still has at least two contaminated incident edges at $t_1$. ■

### 4.2.1 Both $m$ and $n$ are Odd

**Lemma 4.2.3** For a complete bipartite graph $K_{m,n}$ with $3 \leq m \leq n$, suppose that both $m$ and $n$ are odd. If $w_1 \in V_2$, then $fs(K_{m,n}) \geq m + \frac{n+1}{2}$.

**Proof.** If $3 = m \leq n$, then it follows from Lemma 2.1.2 that $fs(K_{m,n}) \geq \frac{m+n}{2} + 2 = \frac{n+1}{2} + 3 = m + \frac{n+1}{2}$. So we only need to consider $5 \leq m \leq n$ in the following. Since $w_1 \in V_2$ and $w_1$ is cleared at $t_1$, we know each vertex in $V_1$ must be guarded by a searcher at the moment $t_1$. If $\max\{a_1, a_2\} \geq \frac{n+1}{2}$, then $fs(K_{m,n}) \geq m + \frac{n+1}{2}$. Suppose that $\max\{a_1, a_2\} \leq \frac{n-1}{2}$. Note that $a_1 + a_2 \geq n - 2$ and both $m$ and $n$ are odd. We know $\min\{a_1, a_2\} \geq \frac{n-3}{2}$. Further, $a_1$ and $a_2$ cannot both equal to $\frac{n-3}{2}$; otherwise, $a_1 + a_2 = n - 3 < n - 2$. Hence, there are two cases.

**Case 1.** $a_1 = \frac{n-1}{2}$. If $w_1$ contains a searcher at $t_1$, then $fs(K_{m,n}) \geq |V_1| + |A_1| + 1 = m + a_1 + 1 = m + \frac{n+1}{2}$. If $w_1$ contains no searchers at $t_1$, then for the sake of contradiction, we assume that $m + \frac{n-1}{2}$ searchers can clear $K_{m,n}$. Since $|V_1 \cup A_1| = m + \frac{n-1}{2}$, we know each vertex in $V_1 \cup A_1$ contains exactly one searcher at $t_1$, and no searchers are located on other vertices. Consider the moment $t_1$. From Lemma 4.2.2, we know each vertex in $A_1$ has at least two contaminated incident edges at $t_1$. Further, since $|V_2 \setminus \{A_1 \cup \{w_1\}\}| = n - \frac{n-1}{2} - 1 \geq 2$, there are at least two vertices in $V_2$ which have no cleared incident edges. Therefore, each vertex in $V_1$ has at least two contaminated
incident edges. Observe that every vertex in \( V_1 \cup A_1 \) contains exactly one searcher and has at least two contaminated incident edges. Therefore, all searchers get stuck at \( t_1 \), which contradicts the assumption that \( m + \frac{n-1}{2} \) searchers can clear \( K_{m,n} \). Hence, \( fs(K_{m,n}) \geq m + \frac{n+1}{2} \).

Case 2. \( a_1 = \frac{n-3}{2} \). Since \( \max\{a_1, a_2\} \leq \frac{n-1}{2} \) and \( a_1 + a_2 \geq n - 2 \), we know \( a_2 = \frac{n-1}{2} \). Further, since \( a_1 + a_2 = n - 2 \), we know \( A_1 \cap A_2 = \emptyset \), and hence each vertex in \( A_2 \) should always contain a searcher after \( t_2 \). For the sake of contradiction, assume that \( m + \frac{n-1}{2} \) searchers can clear \( K_{m,n} \). Recall that at the moment \( t_2 \), each vertex in \( A_2 \cup V_1 \) is occupied by a searcher and \( |A_2 \cup V_1| = m + \frac{n-1}{2} \), we know each vertex in \( A_2 \cup V_1 \) is occupied by exactly one searcher at \( t_2 \). Let \( x_1 x_2 \) denote the last cleared edge before \( t_2 \), which is cleared by sliding a searcher from \( x_1 \) to \( x_2 \). Note that each vertex in \( V_1 \) is occupied by a searcher between \( t_1 \) and \( t_2 \). We know \( x_2 \) must be in \( A_2 \), and \( x_2 \) contains no searchers before \( x_1 x_2 \) is cleared. Thus, \( x_1 x_2 \) is the only cleared edge incident on \( x_2 \) at \( t_2 \). Recall that \( a_1 + a_2 = n - 2 \), it is easy to see that there is still a vertex in \( V_2 \), say \( x_3 \), which has no cleared incident edges at \( t_2 \). Hence, \( w_2 x_3 \) must be cleared by the next sliding action after \( t_2 \). When \( w_2 \) is cleared, we know both of \( x_2 \) and \( x_3 \) have exactly one cleared incident edge, and the two edges must be \( w_2 x_2 \) and \( w_2 x_3 \). Therefore, when \( w_2 \) is cleared, each vertex in \( V_1 \) except \( w_2 \) has at least two contaminated incident edges. Note that each vertex in \( A_2 \) should be guarded by a searcher after \( t_2 \). Hence, every searcher gets stuck after \( w_2 \) is cleared. This contradicts the assumption that \( m + \frac{n-1}{2} \) searchers can clear \( K_{m,n} \). Therefore, \( fs(K_{m,n}) \geq m + \frac{n+1}{2} \).

**Corollary 4.2.4** For a complete bipartite graph \( K_{m,n} \) with \( 3 \leq m \leq n \), sup-
pose that both $m$ and $n$ are odd. If $w_1 \in V_1$, then $fs(K_{m,n}) \geq m + \frac{n+1}{2}$ when $m = 3$, and $fs(K_{m,n}) \geq n + \frac{m+1}{2}$ when $m \geq 5$.

From Lemma 4.2.3 and Corollary 4.2.4, we are ready to present the lower bound on $fs(K_{m,n})$ when both $m$ and $n$ are odd. Note that since $m \leq n$, $\min\{m + \frac{n+1}{2}, n + \frac{m+1}{2}\} = m + \frac{n+1}{2}$.

**Theorem 4.2.5** Given a complete bipartite graph $K_{m,n}$ with $3 \leq m \leq n$, if both $m$ and $n$ are odd, then $fs(K_{m,n}) \geq m + \frac{n+1}{2}$.

### 4.2.2 $m$ is Odd and $n$ is Even

**Lemma 4.2.6** For a complete bipartite graph $K_{m,n}$ with $3 \leq m < n$, suppose that $m$ is odd and $n$ is even. If $w_1 \in V_2$, then $fs(K_{m,n}) \geq m + \frac{n}{2}$.

**Proof.** If $\max\{a_1, a_2\} \geq \frac{n}{2}$, then it is easy to see that $fs(K_{m,n}) \geq m + \frac{n}{2}$.

Suppose that $\max\{a_1, a_2\} < \frac{n}{2}$. Since $a_1 + a_2 \geq n - 2$ and $n$ is even, we know $a_1 = a_2 = \frac{n-2}{2}$ and $A_1 \cap A_2 = \emptyset$. Consider the moment $t_1$. We know each vertex in $V_1 \cup A_1$ contains a searcher. For the sake of contradiction, we assume that $m + \frac{n-2}{2}$ searchers can clear $K_{m,n}$. Then each vertex in $V_1 \cup A_1$ contains exactly one searcher at $t_1$. From Lemma 4.2.2, we know each vertex in $A_1$ has at least two contaminated incident edges. Further, since $A_1 \cap A_2 = \emptyset$ and $|V_2 \setminus \{A_1 \cup \{w_1\}\}| = n - \frac{n-2}{2} - 1 \geq 2$, we know there are at least two vertices in $V_2$ which have no cleared incident edges at $t_1$. Thus, each vertex in $V_1$ has at least two contaminated incident edges at $t_1$, and hence, all searchers get stuck at $t_1$. This contradicts the assumption that $m + \frac{n-2}{2}$ searchers can clear $K_{m,n}$.
Therefore, \( \text{fs}(K_{m,n}) \geq m + \frac{n}{2} \).

In the following, we consider the case when \( w_1 \in V_1 \).

**Lemma 4.2.7** For a complete bipartite graph \( K_{m,n} \) with \( 3 \leq m < n \), suppose that \( m \) is odd and \( n \) is even. If \( w_1 \in V_1 \), then \( \text{fs}(K_{m,n}) \geq n + 1 \) when \( m = 3 \), and \( \text{fs}(K_{m,n}) \geq n + 3 \) when \( m \geq 5 \).

**Proof.** If \( w_1 \in V_1 \), then \( w_2 \in V_2 \). At the moment \( t_1 \), since \( w_1 \) is the first cleared vertex, each vertex in \( V_2 \) is occupied by a searcher. Let \( w_3 \) denote the second cleared vertex of \( K_{m,n} \). If \( w_3 \in V_2 \), then we know each vertex of \( K_{m,n} \) except \( w_1 \) and \( w_3 \) must be occupied by a searcher before \( w_3 \) is cleared. Hence, \( \text{fs}(K_{m,n}) \geq m + n - 2 \). If \( w_3 \in V_1 \), then we have two cases:

Case 1. \( m = 3 \). Assume that \( n \) searchers can clear \( K_{m,n} \). Consider the moment \( t_1 \). Note that \( |V_2| = n \) and each vertex in \( V_2 \) is occupied by a searcher at \( t_1 \). Hence, each vertex in \( V_2 \) contains exactly one searcher at \( t_1 \) and no searchers are located on other vertices. Since there are still two vertices in \( V_1 \) which have no cleared incident edges, then each vertex in \( V_2 \) has two contaminated incident edges. Thus, it is impossible to move any of the searchers located on \( V_2 \) after \( t_1 \). This contradicts our assumption that \( n \) searchers can clear \( K_{m,n} \). Therefore, \( \text{fs}(K_{m,n}) \geq n + 1 \) when \( m = 3 \).

Case 2. \( m \geq 5 \). For the sake of contradiction, we assume that \( n + 2 \) searchers are sufficient to clear \( K_{m,n} \). We have three subcases:

Case 2.1. \( w_3 \) contains no searchers after it is cleared. Then the last two cleared edges incident on \( w_3 \) are both cleared by sliding a searcher from \( w_3 \) to
V2. After w3 is cleared, all searchers will get stuck within five steps. We give an example in the following to illustrate this case in which all searchers get stuck within five steps after w3 is cleared. Consider the moment in Figure 4.4. u1 is the only cleared vertex, and u2 has exactly two contaminated incident edges. Let w1 = u1 and w3 = u2. Since w3 contains no searchers after it is cleared, then w3 contains exactly two searchers at the moment in Figure 4.4. Note that we assume n + 2 searchers can clear Kn,m. Hence, at the moment in Figure 4.4, u2 contains exactly two searchers, and each vertex in V2 is occupied by exactly one searcher. Clear u2 by sliding a searcher from u2 to v4 and to v5 respectively. Therefore, when u2 is cleared, v1, v2 and v3 contain exactly one searcher, v4 and v5 contain exactly two searchers, and no searchers are located on V1. Next, slide a searcher located on v4 from v4 to u3 along the edge v4u3, slide a searcher located on v5 to u4 along the path v5u3v3u4, and slide the searcher located on v3 to u5 along the edge v3u5. Then, it is easy to see that every searcher will get stuck after v3u5 is cleared. Similarly, we can show that in all other possible cases, all searchers will get stuck within five steps after w2 is cleared. This contradicts the assumption that n + 2 searchers are sufficient to clear Kn,m. Therefore, fs(Kn,m) ≥ n + 3.

Case 2.2. w3 contains exactly one searcher after it is cleared. Since w3 has degree at least 6, we know that the last cleared edge incident on w3 has to be cleared by sliding a searcher from w3 to V2. Consider the moment when w3 is cleared. Note that each vertex in V2 is occupied by a searcher between t1 and t2, and there are at least m – 2 ≥ 3 vertices in V1 which contain no searchers and have no cleared incident edges. Since we assume that n + 2 searchers are
Figure 4.4: In this case, $w_1 = u_1$ and $w_3 = u_2$. At this moment, $u_2$ contains exactly two searchers, each vertex in $V_2$ is occupied by exactly one searcher, and no searchers are located on other vertices.

sufficient to clear $K_{m,n}$, hence, there is only one vertex in $V_2$ which contains two searchers. It is easy to see that all searchers get stuck within one step after $w_3$ is cleared, which is a contradiction. Therefore, $fs(K_{m,n}) \geq n + 3$.

Case 2.3. $w_3$ contains exactly two searchers after it is cleared. Consider the moment at which $w_3$ is cleared. Note that there are still at least $m - 2 \geq 3$ vertices in $V_1$ which contain no searchers and have no cleared incident edges. Further, each vertex in $V_2$ is occupied by exactly one searcher. Hence, it is easy to see that all searchers get stuck after $w_3$ is cleared. Therefore, $fs(K_{m,n}) \geq n + 3$.

From the above cases, if $w_1 \in V_1$, then $fs(K_{m,n}) \geq \min\{m+n-2,n+1\} = n+1$ when $m = 3$, and $fs(K_{m,n}) \geq \min\{m+n-2,n+3\} = n+3$ when $m \geq 5$.

From Lemmas 4.2.6 and 4.2.7, we know: (1) when $m = 3$, $fs(K_{m,n}) \geq \min\{m + \frac{n}{2}, n + 1\} = m + \frac{n}{2}$; (2) when $m \geq 5$, $fs(K_{m,n}) \geq \min\{m + \frac{n}{2}, n + 3\}$.

Hence, we are now ready to give the lower bound on $fs(K_{m,n})$ when $m$ is odd, $n$ is even and $3 \leq m \leq n$. 

81
**Theorem 4.2.8** For a complete bipartite graph $K_{m,n}$ with $3 \leq m < n$, if $m$ is odd and $n$ is even, then $fs(K_{m,n}) \geq \min\{n + 3, m + \frac{n}{2}\}$.

From Theorems 4.2.5 and 4.2.8 above, in combination with Lemma 4 and Theorem 4 in [22], we have a complete solution to $fs(K_{m,n})$.

**Theorem 4.2.9** For a complete bipartite graph $K_{m,n}$ with $1 \leq m \leq n$,

$$
fs(K_{m,n}) = \begin{cases}
\left\lceil \frac{n}{2} \right\rceil, & m = 1, \\
2, & m = n = 2, \\
3, & m = 2 \text{ and } n \geq 3, \\
m + \frac{n + 1}{2}, & 3 \leq m \leq n \text{ and both } m \text{ and } n \text{ are odd,} \\
\min\{n + 3, m + \frac{n}{2}\}, & 3 \leq m < n, m \text{ is odd and } n \text{ is even,} \\
6, & m = 4 \text{ and } n \geq 4, \\
m + 3, & 6 \leq m \leq n \text{ and } m \text{ is even.}
\end{cases}
$$

### 4.3 Complete Split Graphs $S_{m,n}$

In this section, we consider complete split graphs $S_{m,n}$ with $m, n \geq 1$, which also form a special class of $k$-partite graphs $K_{n_1,\ldots,n_k}$ when $1 = n_1 = \cdots = n_{k-1} \leq n_k$. We start with some initial cases.

**Lemma 4.3.1** For a complete split graph $S_{m,n}$, if $n = 1$, then
\[ fs(S_{m,1}) = \begin{cases} 
1, & m = 1, \\
2, & m = 2, \\
m + 1, & m \geq 3. 
\end{cases} \]

In the following, we consider the fast search number of \( S_{m,n} \) when \( n \geq 2 \). Let \( S_{S_{m,n}} \) denote an optimal fast search strategy for clearing \( S_{m,n} \). Let \( w'_1 \) denote the first cleared vertex in \( S_{S_{m,n}} \), and let \( t'_1 \) denote the moment at which \( w'_1 \) is cleared.

**4.3.1 \( m \text{ is Odd and } n \geq 2 \)**

When \( m = 1 \) and \( n \geq 2 \), \( S_{m,n} \) is a star with \( n \) leaves. It is easy to see that \( S_{1,n} \) can be cleared with \( \lceil \frac{n}{2} \rceil \) searchers. Further, it follows from Lemma 2.1.1 that \( fs(S_{1,n}) \geq \frac{1}{2}|V_{odd}(S_{1,n})| = \lceil \frac{n}{2} \rceil \). Hence, we have the next lemma.

**Lemma 4.3.2** For a complete split graph with \( m = 1 \), if \( n \geq 2 \), then \( fs(S_{1,n}) = \lceil \frac{n}{2} \rceil \).

**Lemma 4.3.3** For a complete split graph \( S_{m,n} \) with \( m \geq 3 \) and \( n \geq 2 \), if \( m \) is odd, then \( fs(S_{m,n}) = m + \lceil \frac{n}{2} \rceil \).

**Proof.** If \( w'_1 \in V_1 \), then each vertex of \( S_{m,n} \) except \( w'_1 \) should be guarded by a searcher at the moment \( t'_1 \). Hence, \( fs(S_{m,n}) \geq m - 1 + n \). If \( w'_1 \in V_2 \), then we have two cases:
Case 1. $n$ is even. If $n = 2$, then it follows from Lemma 2.1.3 that\[fs(S_{m,n}) \geq m + 1 = m + \frac{n}{2} .\] If $n \geq 4$, then similar to the proof of Lemma 4.2.6, we can show that $fs(S_{m,n}) \geq m + \frac{n}{2}$.

Case 2. $n$ is odd. If $n = 3$, then it follows from Lemma 4.1.2 that $fs(S_{m,n}) \geq 2 + m = m + \frac{n+1}{2}$. If $n = 5$, then similar to the proof of Lemma 4.2.3 when $n \geq 5$, we can show that $fs(S_{m,n}) \geq m + \frac{n+1}{2}$.

From the above cases, when $m \geq 3$ and $n \geq 2$, $fs(S_{m,n}) \geq \min\{m - 1 + n, m + \lceil \frac{n}{2}\rceil\} = m + \lceil \frac{n}{2}\rceil$. In combination with Theorem 4.1.5, we have $fs(S_{m,n}) = m + \lceil \frac{n}{2}\rceil$, when $m \geq 3$ and $n \geq 2$. \[\square\]

From Lemmas 4.3.2 and 4.3.3, we are ready to give the fast search number of $S_{m,n}$ when $m$ is odd and $n \geq 2$.

**Theorem 4.3.4** For a complete split graph $S_{m,n}$, if $m$ is odd, then

$$fs(S_{m,n}) = \begin{cases} \lfloor \frac{n}{2}\rfloor, & m = 1, n \geq 2, \\ m + \lfloor \frac{n}{2}\rfloor, & m \geq 3, n \geq 2. \end{cases}$$

### 4.3.2 $m$ is Even and $n \geq 2$

Now we consider the complete split graph $S_{m,n}$ where $m$ is even and $n \geq 2$. We first give the following upper bound on $fs(S_{m,n})$.

**Lemma 4.3.5** For a complete split graph $S_{m,n}$ with $m = 2$ and $n \geq 2$, we have $fs(S_{2,n}) \leq 3$. 84
Proof. Let $V_1 = \{u_1, u_2\}$ and $V_2 = \{v_1, v_2, \ldots, v_n\}$. Place a searcher on $u_1$ and $u_2$ respectively. Place a second searcher, say $\lambda$, on $u_1$. Hence we use 3 searchers. Let $\lambda$ clear $v_1$ by sliding along the path $u_1v_1u_2$. Next let $\lambda$ clear $v_2$ by sliding along the path $u_2v_2u_1$. Repeat this process to clear all the other vertices of $S_{m,n}$.

Lemma 4.3.6 For a complete split graph $S_{m,n}$ with $m = 4$ and $n \geq 3$, we have $fs(S_{4,n}) \leq 6$.

Proof. We present a fast search strategy for $S_{4,n}$ which uses 6 searchers.

Let $V_1 = \{u_1, u_2, u_3, u_4\}$ and $V_2 = \{v_1, v_2, \ldots, v_n\}$. Place five searchers on $u_1$. Denote these searchers by $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$. Place a searcher, say $\lambda_6$, on $u_2$. So we use 6 searchers in total.

Slide $\lambda_2$ to $u_2$ along $u_1u_2$, slide $\lambda_3$ to $u_3$ along $u_1u_3$ and slide $\lambda_4$ to $u_4$ along $u_1u_4$. Slide $\lambda_6$ along the path $u_2u_3u_4u_2$. Then the subgraph induced by $V_1$ is cleared. Slide $\lambda_5$ and $\lambda_6$ from $u_1$ and $u_2$ to $v_1$ respectively. Then clear $v_1$ by sliding $\lambda_5$ to $u_3$ along $v_1u_3$, and sliding $\lambda_6$ to $u_4$ along $v_1u_4$. Next, slide $\lambda_5$ and $\lambda_6$ from $u_3$ and $u_4$ to $v_2$ respectively. Then clear $v_2$ by sliding $\lambda_5$ to $u_1$ along $v_2u_1$, and sliding $\lambda_6$ to $u_2$ along $v_2u_2$. By repeating the above process, we can easily clear $S_{4,n}$ using 6 searchers.

Lemma 4.3.7 For a complete split graph $S_{m,n}$ with $m \geq 4$ and $n = 2$, we have $fs(S_{m,2}) \leq m + 1$.

Proof. We present a fast search strategy for $S_{m,2}$ which uses $m + 1$ searchers.
Let \( V_1 = \{u_1, u_2, \ldots, u_m\} \) and \( V_2 = \{v_1, v_2\} \). Place \( m \) searchers on \( v_1 \).
Denote these searchers by \( \lambda_1, \lambda_2, \ldots, \lambda_m \). Place a searcher, say \( \lambda_{m+1} \), on \( u_1 \).
Hence, we use \( m + 1 \) searchers.

Clear \( v_1 \) by sliding a searcher from \( v_1 \) to each of its neighbors. Note that the subgraph induced by vertices in \( V_1 \setminus \{u_m\} \) is Eulerian. We slide \( \lambda_{m+1} \) to clear this Eulerian subgraph, and \( \lambda_{m+1} \) will finally stop on \( u_1 \). Next, slide \( \lambda_1 \) to \( u_m \) along the edge \( u_1 u_m \), and slide \( \lambda_{m+1} \) to \( v_2 \) along the edge \( u_1 v_2 \). Observe that the subgraph induced by \( u_2, u_3, \ldots, u_m, v_2 \) is Eulerian and each of those vertices contains exactly one searcher except for \( u_m \) which contains exactly two searchers. Therefore, we can use \( \lambda_1 \) to clear the induced Eulerian subgraph.

**Theorem 4.3.8** For a complete split graph \( S_{m,n} \),

\[
fs(S_{m,n}) = \begin{cases} 
3, & m = 2, n \geq 2, \\
6, & m = 4, n \geq 3, \\
m + 1, & m \geq 4, n = 2, \\
m + 2, & m \geq 6, n = 3.
\end{cases}
\]

**Proof.** (1) \( m = 2 \) and \( n \geq 2 \). If \( w'_1 \in V_1 \), then \( fs(S_{2,n}) \geq |V_1 \cup V_2| - 1 = 2 + n - 1 \geq 3 \). If \( w'_1 \in V_2 \), then let \( w'_1 x_1 \) denote the last sliding action at \( t'_1 \).
Suppose that two searchers are sufficient to clear \( S_{m,n} \). When \( w'_1 \) is cleared, each vertex in \( V_1 \) should be occupied by a searcher. Therefore, at the moment \( t'_1 \), each vertex in \( V_1 \) is occupied by exactly one searcher and no searchers are located on other vertices. Hence, \( x_1 \) has no cleared incident edges before \( w'_1 x_1 \).
is cleared. Further, the only edge between two vertices in \( V_1 \) is contaminated when \( w'_1x_1 \) is cleared. Since there is at least one vertex in \( V_2 \) which has no cleared incident edges, we know each vertex in \( V_1 \) has at least two contaminated incident edges. Therefore, no searchers can move after \( w'_1 \) is cleared. This is a contradiction. Thus, when \( m = 2 \) and \( n \geq 2 \), \( \text{fs}(S_{2,n}) \geq 3 \).

(2) \( m = 4 \) and \( n \geq 3 \). It follows from Lemmas 4.1.2 and 4.3.6 that \( \text{fs}(S_{4,n}) = m + 2 = 6 \).

(3) \( m \geq 4 \) and \( n = 2 \). Clearly, \( S_{m,2} \) contains a clique \( K_{m+1} \). From Lemmas 2.1.3 and 4.3.7, we have \( \text{fs}(S_{m,2}) = m + 1 \).

(4) \( m \geq 6 \) and \( n = 3 \). It follows from Theorem 4.1.3 that \( \text{fs}(S_{m,3}) = m + n - 1 = m + 2 \). \( \blacksquare \)

From Lemma 4.1.2 and Theorem 4.1.4, we give a lower bound and an upper bound on \( \text{fs}(S_{m,n}) \) when \( m \geq 6 \) and \( n \geq 4 \).

**Theorem 4.3.9** For a complete split graph \( S_{m,n} \) with \( m \geq 6 \) and \( n \geq 4 \), if \( m \) is even, then \( m + 2 \leq \text{fs}(S_{m,n}) \leq m + 3 \).
Chapter 5

Fast Searching on $k$-Combinable Graphs

Finding an optimal fast search strategy for a graph is challenging, sometimes even when the graph has very small treewidth, like cacti, cartesian products of a tree and an edge, etc. However, it may be much easier to find an optimal fast search strategy for smaller subgraphs of the given graph. Although fast searching is not subgraph-closed, this observation still motivates us to establish relationships between optimal fast search strategies for a graph and its subgraphs. In the following, we introduce the notion of $k$-combinable graphs and propose a new method for computing their fast search number.
5.1 Align Operation on Pendant Edges

In this section, we will first introduce a class of graphs named \( k \)-combinable graphs. Then we describe our new method for finding an optimal fast search strategy for \( k \)-combinable graphs. Let \( G \) be a connected graph and let \( E'_G \) be the set of all pendant edges of \( G \). The profile of \( G \) is an ordered tuple \( \pi_G = (\pi_1, \ldots, \pi_z) \) of positive integers, which is defined as follows (see Figure 5.1 for an example):

1. If \( E'_G = \emptyset \), then \( z = 1 \) and \( \pi_1 = \text{fs}(G) \).

2. If \( E'_G \neq \emptyset \) and \( |E'_G| = k \), then \( z = k!2^k \) and each component \( \pi_i \) of \( \pi_G \) is associated with a specific permutation \( \sigma \) and a specific orientation of each edge in \( E'_G \). In particular, \( \pi_i \) is the smallest number of searchers with which a fast search strategy can clear \( G \) if it traverses the edges in \( E'_G \) in the order of \( \sigma \) and in the directions as given by the chosen orientations.

Let \( G_1 \) be a connected graph that has \( k_1 \geq 1 \) pendant edges, and let \( G_2 \) be a connected graph having \( k_2 \geq 1 \) pendant edges. We choose \( k \) to be a constant satisfying that \( 1 \leq k \leq \min\{k_1, k_2\} \). Let \( \overrightarrow{e_1} = (u_1u'_1, \ldots, u_ku'_k) \), where \( u_iu'_i \in E(G_1) \) and \( u'_i \) is a leaf node. Let \( \overrightarrow{e_2} = (v_1v'_1, \ldots, v_kv'_k) \), where \( v_iv'_i \in E(G_2) \) and \( v'_i \) is a leaf node. Let \( H \) be the graph obtained from \( G_1 \) and \( G_2 \) by performing the following operations on \( G_1 \) and \( G_2 \) with respect to \( \overrightarrow{e_1} \) and \( \overrightarrow{e_2} \):
Figure 5.1: The graph above has two pendant edges $v_3v_4$ and $v_5v_6$. Each of the pendant edges can be cleared by sliding a searcher from either of the endpoints to the other. The profile of the graph is $(1, 2, 3, 3, 1, 2, 3, 3)$, and Table 5.1 lists each search number that is associated with a specific permutation and orientation.

Table 5.1: Number of searchers for specific permutations and orientations.

<table>
<thead>
<tr>
<th>Permutation</th>
<th>Orientation</th>
<th>Number of Searchers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(v_3v_4, v_5v_6)$</td>
<td>$\leftarrow$</td>
<td>1</td>
</tr>
<tr>
<td>$(v_3v_4, v_5v_6)$</td>
<td>$\leftarrow$</td>
<td>2</td>
</tr>
<tr>
<td>$(v_3v_4, v_5v_6)$</td>
<td>$\rightarrow$</td>
<td>3</td>
</tr>
<tr>
<td>$(v_5v_6, v_3v_4)$</td>
<td>$\leftarrow$</td>
<td>1</td>
</tr>
<tr>
<td>$(v_5v_6, v_3v_4)$</td>
<td>$\leftarrow$</td>
<td>2</td>
</tr>
<tr>
<td>$(v_5v_6, v_3v_4)$</td>
<td>$\rightarrow$</td>
<td>3</td>
</tr>
<tr>
<td>$(v_5v_6, v_3v_4)$</td>
<td>$\rightarrow$</td>
<td>3</td>
</tr>
</tbody>
</table>

1. remove edges $u_iu'_i$ and $v_iv'_i$, for $1 \leq i \leq k$;

2. remove vertices $u'_i$ and $v'_i$, for $1 \leq i \leq k$;

3. connect $u_i$ and $v_i$ by adding a new edge, for $1 \leq i \leq k$.

Note that the above operations depend on the choice of the sequences $\vec{e}_{1}$ and $\vec{e}_{2}$ which we will henceforth call edge pairing sequences. If we permute either of the edge pairing sequences, this would create a different result. Hence,
we define the align operation on $G_1$ and $G_2$ with respect to $\vec{e_1}$ and $\vec{e_2}$, denoted as $(G_1, \vec{e_1}) \triangle (G_2, \vec{e_2})$, to be the graph obtained by performing the above operations.

**Definition 5.1.1** Let $m \geq 2$. Let $G_1, \ldots, G_m$ be connected graphs. The sequence $(G_1, \ldots, G_m)$ is $k$-combinable if there are edge sequences $\vec{e_1}, \ldots, \vec{e_m}$, $\vec{e_{1,2}}, \ldots, \vec{e_{1,m-1}}$ such that:

1. For $1 \leq i \leq m$, $\vec{e_i}$ is a sequence of pendant edges of $G_i$.
2. For $2 \leq i \leq m - 1$, $\vec{e_{1,i}}$ is a sequence of pendant edges of $H_i$, where $H_2 = (G_1, \vec{e_1}) \triangle (G_2, \vec{e_2})$, and $H_{i+1} = (H_i, \vec{e_{1,i}}) \triangle (G_{i+1}, \vec{e_{i+1}})$.
3. For $1 \leq i \leq m$, the set of all edges of $G_i$, which occur in $\vec{e_1}, \ldots, \vec{e_m}$ and $\vec{e_{1,2}}, \ldots, \vec{e_{1,m-1}}$, has size at most $k$.
4. For $2 \leq j \leq m - 1$, the set of all edges of $H_j$, which occur in $\vec{e_1}, \ldots, \vec{e_m}$ and $\vec{e_{1,2}}, \ldots, \vec{e_{1,m-1}}$, has size at most $k$.

Further, we then call $H_m$ a $k$-combination of $(G_1, \ldots, G_m)$, in particular, this is the $k$-combination of $(G_1, \ldots, G_m)$ with respect to $\vec{e_1}, \ldots, \vec{e_m}$, $\vec{e_{1,2}}, \ldots, \vec{e_{1,m-1}}$.

Obviously, there may exist more than one graph that is a $k$-combination of $(G_1, G_2, \ldots, G_m)$. Further, for each $k$-combination $G$ of $(G_1, G_2, \ldots, G_m)$, there exist specific $\vec{e_{1,2}}, \ldots, \vec{e_{1,m-1}}$ and $\vec{e_1}, \ldots, \vec{e_m}$ for obtaining $G$. In the remainder of this section, we always assume that every time an algorithm handles
profiles of graphs, it implicitly associates the profiles with corresponding $e_{1,i}$ and $e_{j}$, where $2 \leq i \leq m - 1$ and $1 \leq j \leq m$.

**Theorem 5.1.2** There exists an algorithm that, given the profiles and edge pairing sequences of $G_1$ and $G_2$ such that $G$ is the $k$-combination of $(G_1, G_2)$ with respect to the edge pairing sequences, runs in $O((k_1 + k_2 - k)!2^{k_1 + k_2 - k})$ time to compute the profile of $G$. Here $k_i$ refers to the number of pendant edges of $G_i$, where $1 \leq i \leq 2$.

**Proof.** We briefly introduce the idea of how to compute the profile of $G$. Since $G_1$ and $G_2$ have $k_1$ and $k_2$ pendant edges respectively, the sizes of profiles of $G_1$ and $G_2$ are $k_1!2^{k_1}$ and $k_2!2^{k_2}$. Let $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$ denote the edge pairing sequences of $G_1$ and $G_2$ respectively. Consider all the edges in $\overrightarrow{e_1}$ and $\overrightarrow{e_2}$. If we are given a set of rules instructing how these edges are cleared in a strategy, then in accordance with the rules, we can figure out the number of searchers that need to be placed on the non-leaf vertices in $V(G_1)$ and $V(G_2)$. For each parameter in the profile of $G$, it takes $O(k!2^k)$ time to compute its value. Further, we know the size of the profile of $G$ is $(k_1 + k_2 - 2k)!2^{k_1 + k_2 - 2k}$. Hence, the time complexity for computing the profile of $G$ is $O((k_1 + k_2 - k)!2^{k_1 + k_2 - k})$.

From Theorem 5.1.2, it is obvious that our technique can be applied to find an optimal fast search strategy for quite complicated graphs, if the graph can be split into two smaller graphs for which fast search strategies are easy to find (see Figure 5.2). Moreover, if we are given $G$ that is a $k$-combination of $(G_1, \ldots, G_m)$ where $m \geq 3$, by repeatedly applying the procedure presented.
in the proof of Theorem 5.1.2, we can find an optimal fast search strategy for $G$ as stated in Theorem 5.1.3. This novel method reveals an interesting property of fast searching that has not been exploited systematically in the literature to date. Moreover, as we will show in the remainder of this chapter, our technique can be applied to a wide class of graphs.

![Graph](image)

Figure 5.2: The graph above can be split into two vertex-disjoint trees. Note that there are exactly four edges connecting the two trees, and the fast search number of a tree can be computed in polynomial time. Upon knowing the profiles of these two trees, we then can apply the new method presented in Theorem 5.1.2 to compute the fast search number of the entire graph.

**Theorem 5.1.3** Let $G$ be a $k$-combination of $(G_1, \ldots, G_m)$ with respect to $\overrightarrow{e_1}, \ldots, \overrightarrow{e_m}, \overrightarrow{e_{1,2}}, \ldots, \overrightarrow{e_{1,m-1}}$, where $G_1, \ldots, G_m$ are connected graphs and $k$ is a constant. There exists an algorithm which, given (1) the profiles of $G_1$, $G_2$, $\ldots$ $G_m$ in sequence, and (2) $\overrightarrow{e_1}, \ldots, \overrightarrow{e_m}$ and $\overrightarrow{e_{1,2}}, \ldots, \overrightarrow{e_{1,m-1}}$, runs in polynomial time to compute the profile of $G$.

**Corollary 5.1.4** Let $G$ be a $k$-combination of $(G_1, \ldots, G_m)$ with respect to $\overrightarrow{e_1}, \ldots, \overrightarrow{e_m}, \overrightarrow{e_{1,2}}, \ldots, \overrightarrow{e_{1,m-1}}$, where $G_1, \ldots, G_m$ are connected graphs and $k$ is a constant. There exists an algorithm which, given (1) the profiles of $G_1$, $G_2$, $\ldots$ $G_m$ in sequence, and (2) $\overrightarrow{e_1}, \ldots, \overrightarrow{e_m}$ and $\overrightarrow{e_{1,2}}, \ldots, \overrightarrow{e_{1,m-1}}$, runs in polynomial time to compute the fast search number of $G$. 
Remarks: Note that $G_i$ may not always be a subgraph of $G$. Consider the graph in Figure 5.3. After removing edges $u_1v_1$, $u_1v_2$, $u_2v_2$, $u_3v_2$, $v_3w_1$, $v_3w_2$, $v_4w_1$ and $v_4w_2$, the graph can be split into three components that are represented by three dotted circles. Let $H_1$, $H_2$ and $H_3$ denote the three components from left to right. Let $H'_1$ be obtained from $H_1$ by adding the edges $u_1v_1$, $u_1v_2$, $u_2v_2$ and $u_3v_2$ respectively. Let $G_1$ be obtained from $H_1$ by adding two pendant edges to $u_1$, and adding one pendant edge to $u_2$ and $u_3$ respectively. Consider $H'_1$. Note that $u_1$, $u_2$ and $u_3$ have a common neighbor $v_2$. Hence, if $v_2$ has both cleared and contaminated incident edges at some moment in a strategy, then at least one searcher must reside on $v_2$. In $G_1$, however, we know that $u_1$, $u_2$ and $u_3$ share no common neighbor that is outside $H_1$. Hence, no searcher needs to reside on the vertices outside $H_1$ whenever its incident edge is cleared.

![Figure 5.3: An example to illustrate that $G_i$ may not be a subgraph of $G$.](image)

### 5.2 Cactus Graphs

A connected graph is a *cactus* if and only if each of its edges is contained in at most one cycle. In the remainder of this section, we use $G$ to denote a cactus graph. Let $v \in V(G)$ and let $G_1, \ldots, G_k$ be all the connected components from
G by deleting v and all its incident edges. We use $G_v^i$ to denote the subgraph of G induced by $V(G_i) \cup \{v\}$, where $1 \leq i \leq k$. Hence, $G_v^1, \ldots, G_v^k$ must satisfy:

(i) $V(G_v^1) \cup \cdots \cup V(G_v^k) = V(G)$,

(ii) $V(G_v^i) \cap V(G_v^j) = \{v\}$, where $1 \leq i \neq j \leq k$, and

(iii) $u_1, u_2 \in V(G)$ are adjacent, only if there exists $i$ such that $u_1, u_2 \in V(G_v^i)$.

**Definition 5.2.1** $G_v^1, \ldots, G_v^k$ are called sub-cacti of G with respect to vertex v.

Consider $G_v^i$, where $1 \leq i \leq k$. Note that v has degree at most two in $G_v^i$. If v is a leaf node in $G_v^i$, then let u be a vertex in $V(G_v^i)$ satisfying $u \sim v$. We use $\pi_I(G_v^i)$ to denote the minimum number of searchers placed on $V(G_v^i) \setminus \{v\}$ in a strategy for $G_v^i$, in which vu is cleared by sliding a searcher from v to u. An I-strategy for $G_v^i$ is a strategy in which (1) vu is cleared by sliding a searcher from v to u, and (2) $\pi_I(G_v^i)$ searchers are placed on $V(G_v^i) \setminus \{v\}$. Note that if vu is cleared by sliding a searcher from v to u in a strategy, then a searcher must be placed on v at the beginning of the strategy. We use $\pi_O(G_v^i)$ to denote the minimum number of searchers placed on $V(G_v^i) \setminus \{v\}$ in a strategy for $G_v^i$, in which vu is cleared by sliding a searcher from u to v. An O-strategy for $G_v^i$ is a strategy for $G_v^i$ in which (1) vu is cleared by sliding a searcher from u to v, and (2) $\pi_O(G_v^i)$ searchers are placed on $V(G_v^i) \setminus \{v\}$.

If v has degree two in $G_v^i$, then let $u_1$ and $u_2$ be the two vertices in $V(G_v^i)$ satisfying that $u_1 \sim v$ and $u_2 \sim v$. For $i \in \{1, 2\}$, we say vu$_i$ is cleared by a slide-in action if a searcher slides from v to $u_i$ along vu$_i$, and we say
vu_i is cleared by a slide-out action if a searcher slides from u_i to v along vu_i. We use \( \pi_{I,I}(G^i_v) \) to denote the minimum number of searchers placed on \( V(G^i_v) \setminus \{v\} \) in a strategy for \( G^i_v \), in which vu_1 and vu_2 are both cleared by slide-in actions. We use \( \pi_{O,O}(G^i_v) \) to denote the minimum number of searchers placed on \( V(G^i_v) \setminus \{v\} \) in a strategy for \( G^i_v \), in which vu_1 and vu_2 are both cleared by slide-out actions. We use \( \pi_{I,O}(G^i_v) \) to denote the minimum number of searchers placed on \( V(G^i_v) \setminus \{v\} \) in a strategy for \( G^i_v \), in which vu_1 or vu_2 is cleared by a slide-in action, and later the other edge is cleared by a slide-out action. We use \( \pi_{O,I}(G^i_v) \) to denote the minimum number of searchers placed on \( V(G^i_v) \setminus \{v\} \) in a strategy for \( G^i_v \), in which vu_1 or vu_2 is cleared by a slide-out action, and later the other edge is cleared by a slide-in action. A strategy for \( G^i_v \) is an II-strategy, in which (1) \( \pi_{I,I}(G^i_v) \) searchers are placed on \( V(G^i_v) \setminus \{v\} \), and (2) vu_1 and vu_2 are both cleared by slide-in actions. In a similar way, we define IO-strategy, OI-strategy and OO-strategy for \( G^i_v \) respectively.

**Definition 5.2.2** Consider a sub-cactus of \( G \) with respect to vertex \( v \), say \( G^i_v \).

1. If \( v \) has exactly one incident edge in \( G^i_v \), then the profile of \( G^i_v \) is defined as the pair \((\pi_{I}(G^i_v), \pi_{O}(G^i_v))\).

2. If \( v \) has exactly two incident edges in \( G^i_v \), then the profile of \( G^i_v \) is defined as the 4-tuple \((\pi_{I,I}(G^i_v), \pi_{I,O}(G^i_v), \pi_{O,I}(G^i_v), \pi_{O,O}(G^i_v))\).

For cactus graph \( G \) and \( v \in V(G) \), we use \( G'_v \) to denote the graph obtained by adding either one or two pendant edges to \( v \). There are two possibilities for \( G'_v \):
(1) \(v\) has one added pendant edge in \(G'_v\), say \(vu\). Let \(\pi_I(G'_v)\) be the minimum number of searchers placed on \(V(G'_v) \setminus \{u\}\) in a strategy for \(G'_v\), in which \(vu\) is cleared by sliding a searcher from \(u\) to \(v\). An \(I\)-strategy for \(G'_v\) is a strategy, in which (a) \(\pi_I(G'_v)\) searchers are placed on \(V(G'_v) \setminus \{u\}\), and (b) \(vu\) is cleared by sliding a searcher from \(u\) to \(v\). In a similar way, we define \(\pi_O(G'_v)\) and \(O\)-strategy for \(G'_v\). The profile of \(G'_v\) is defined as the pair \((\pi_I(G'_v), \pi_O(G'_v))\).

(2) \(v\) has two added pendant edges in \(G'_v\). Notice that there are four distinct ways to clear the two added pendant edges of \(v\). In a similar way, we define (1) \(\pi_{I,I}(G'_v), \pi_{I,O}(G'_v), \pi_{O,I}(G'_v)\) and \(\pi_{O,O}(G'_v)\) for \(G'_v\), and (2) \(II\)-strategy, \(IO\)-strategy, \(OI\)-strategy and \(OO\)-strategy for \(G'_v\). The profile of \(G'_v\) is defined as 4-tuple \((\pi_{I,I}(G'_v), \pi_{I,O}(G'_v), \pi_{O,I}(G'_v), \pi_{O,O}(G'_v))\).

**Definition 5.2.3** For a strategy \(S\) for \(G\), let the reversed strategy of \(S\) be obtained from \(S\) by making the following modifications:

1. Remove all placing actions from \(S\). For each vertex \(v \in V(G)\) that contains searchers at the end of \(S\), insert a placing action at the beginning that places the same number of searchers on \(v\).

2. For each edge \(e \in E(G)\), reverse the sliding action on \(e\) by letting searcher move in the opposite way to clear it.

3. Reverse the order of all sliding actions.

Obviously, the reversed strategy of \(S\) uses the same number of searchers to clear \(G\).
Lemma 5.2.4 $G^i_v$ must have one of the following properties:

1. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) = \pi_{O,I}(G^i_v) = \pi_{O,O}(G^i_v) - 2$;
2. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) = \pi_{O,I}(G^i_v) - 1 = \pi_{O,O}(G^i_v) - 2$;
3. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) = \pi_{O,I}(G^i_v) - 2 = \pi_{O,O}(G^i_v) - 2$;
4. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) - 1 = \pi_{O,I}(G^i_v) - 1 = \pi_{O,O}(G^i_v) - 2$;
5. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) - 1 = \pi_{O,I}(G^i_v) - 2 = \pi_{O,O}(G^i_v) - 2$;
6. $\pi_{I,I}(G^i_v) = \pi_{I,O}(G^i_v) - 2 = \pi_{O,I}(G^i_v) - 2 = \pi_{O,O}(G^i_v) - 2$;
7. $\pi_I(G^i_v) = \pi_O(G^i_v) - 1$.

**Proof.** Consider a strategy for graph $G^i_v$. It is easy to see that if we reverse the strategy in accordance with the rules in Definition 5.2.3, then the reversed strategy can still clear $G^i_v$ using the same number of searchers. If $v$ has exactly one incident edge in $G^i_v$, then $\pi_I(G^i_v) = \pi_O(G^i_v) - 1$. If $v$ has two incident edges in $G^i_v$, then $\pi_{I,I}(G^i_v) = \pi_{O,O}(G^i_v) - 2$; further, we have $\pi_{I,O}(G^i_v) \leq \pi_{O,I}(G^i_v)$.

Therefore, there are seven distinct relations in total among the parameters. It is very easy to see that the 7-th property must be satisfied if $v$ has exactly one incident edge in $G^i_v$. To demonstrate the other six properties, we give an example each in Figures 5.4 to 5.9.

![Figure 5.4: $\pi_{I,I}(G) = \pi_{I,O}(G) = \pi_{O,I}(G) = 2$, and $\pi_{O,O}(G) = 4$.](image-url)
For convenience, we say $G_v$ satisfies $^{(i)}$ if it has the $i$-th property in Lemma 5.2.4, where $1 \leq i \leq 7$. Consider $G_v^1, \ldots, G_v^k$. Let $\chi_v^i$ be the number of sub-cacti that satisfy $^{(i)}$, where $1 \leq i \leq 7$. Obviously, we have $0 \leq \chi_v^i \leq k$. Two strategies for $G$ are said to be equivalent if they use the same number of searchers to clear $G$. 

Figure 5.5: $\pi_{I,I}(G) = \pi_{I,O}(G) = 2$, $\pi_{O,I}(G) = 3$, and $\pi_{O,O}(G) = 4$.

Figure 5.6: $\pi_{I,I}(G) = \pi_{I,O}(G) = 0$, $\pi_{O,I}(G) = \pi_{O,O}(G) = 2$.

Figure 5.7: $\pi_{I,I}(G) = 1$, $\pi_{I,O}(G) = \pi_{O,I}(G) = 2$, $\pi_{O,O}(G) = 3$. 

99
Figure 5.8: $\pi_{I,I}(G) = 0$, $\pi_{I,O}(G) = 1$, $\pi_{O,I}(G) = \pi_{O,O}(G) = 2$.

Figure 5.9: $\pi_{I,I}(G) = 1$, $\pi_{I,O}(G) = \pi_{O,I}(G) = \pi_{O,O}(G) = 3$.

### 5.2.1 Algorithm

For any cactus graph $G$, algorithm FastSearchCactus produces an optimal fast search strategy.
**Algorithm 5** `FastSearchCactus(G)`

**Input:** A cactus graph $G$.

**Output:** The fast search number of $G$.

1: Arbitrarily select a vertex $v$ from $G$. Let $G_1$ be a connected subgraph of $G$ such that: (1) $v \notin V(G_1)$, and (2) there is either one or two edges connecting $v$ and $V(G_1)$. Let $E_{\text{cut}}$ be the set of all edges connecting $v$ and $V(G_1)$. Let $G'_1$ be the subgraph of $G$ induced by $V(G_1) \cup V(E_{\text{cut}})$. Let $G'_2$ be the subgraph of $G$ after removing all vertices in $V(G_1) \setminus V(E_{\text{cut}})$ and all their incident edges.

2: Let $S_{G'_1}$ be the output of `ClearCacti1$(G'_1, v, \emptyset)$`, and let $S_{G'_2}$ be the output of `ClearCacti1$(G'_2, v, E_{\text{cut}})$`.

3: List all the possible combinations of the profiles from $S_{G'_1}$ and $S_{G'_2}$ respectively with respect to sliding actions on the edges connecting $v$ and $V(G_1)$. Output the minimum number of searchers for clearing $G$ among all feasible combinations.

In algorithm `FastSearchCactus`, we define $G'_1$ and $G'_2$. Using algorithm `ClearCacti1` below, we can compute the profiles of $G'_1$ and $G'_2$ respectively. The input of the algorithm is $G'_1$ or $G'_2$, along with a vertex $v \in V(G)$ and an edge set. The output is the profile of the input graph.
Algorithm 6 CLEARCACTI1($G, v, E_{cut}$)

1: Let $H$ be the subgraph of $G$ obtained by removing all edges in $E_{cut}$. Let $G^1_v, \ldots, G^k_v$ be all the sub-cacti of $H$ with respect to $v$.

2: For each sub-cactus $G^i_v$, compute its profile as follows.

(A) If $G^i_v$ is a tree, then let $\pi_I(G^i_v)$ be the number of searchers that are placed on $G^i_v \setminus \{v\}$ in the I-strategy produced by FS($G^i_v$) in [22]. Let $\pi_O(G^i_v) \leftarrow \pi_I(G^i_v) + 1$. Let $(\pi_I(G^i_v), \pi_O(G^i_v))$ be the profile of $G^i_v$.

(B) If $G^i_v$ is a simple cycle, then let $\pi_{I,I}(G^i_v) \leftarrow 0$, $\pi_{I,O}(G^i_v) \leftarrow 0$, $\pi_{O,I}(G^i_v) \leftarrow 2$, and $\pi_{O,O}(G^i_v) \leftarrow 2$. Let $(\pi_{I,I}(G^i_v), \pi_{I,O}(G^i_v), \pi_{O,I}(G^i_v), \pi_{O,O}(G^i_v))$ be the profile of $G^i_v$.

(C) If $G^i_v$ is neither a tree nor a simple cycle, then there are two subcases:

(i) if $v$ is contained in a cycle of $G^i_v$, then let the output of CLEARCACTI2($G^i_v, v$) be the profile of $G^i_v$;

(ii) if $v$ is a leaf node of $G^i_v$, then let $u \in V(G^i_v)$ be the vertex such that $v \sim u$; let the output of CLEARCACTI1($G^i_v, u, \{uv\}$) be the profile of $G^i_v$.

3: If $E_{cut}$ is empty, then compute and return the profile of $G$ based on the profiles of $G^1_v, \ldots, G^k_v$; else, return CLEARCACTI3($G, v, \text{profiles of } G^1_v, \ldots, G^k_v$).

Using algorithm CLEARCACTI2, we can compute the profile of a sub-cactus in which $v$ is contained in a cycle. The input of the algorithm is a sub-cactus $G^i_v$ and a vertex $v \in V(G^i_v)$. The output of the algorithm is the profile of $G^i_v$. 

102
Algorithm 7 CLEARCACTI2($\mathcal{G}_v^j, v$)

1: Let $\mathcal{C} = vu_1 \ldots u_{k'} v$ be the shortest cycle in $\mathcal{G}_v^j$ that contains $v$.

Let $\mathcal{H}_{u_1}, \ldots, \mathcal{H}_{u_{k'}}$ denote those $k'$ connected components that contain $u_1, \ldots, u_{k'}$ respectively, after deleting all edges in $E(\mathcal{C})$ from $\mathcal{G}_v^j$. Let $\mathcal{G}_{u_i}$ denote the connected subgraph obtained from $\mathcal{H}_{u_i}$ by adding two incident edges of $u_i$ that have endpoints in $V(\mathcal{C})$.

2: For $i \in \{1, \ldots, k'\}$, let $E_i \subset E(\mathcal{C})$ be the set containing the two incident edges of $u_i$, and let the output of CLEARCACTI1($\mathcal{G}_{u_i}, u_i, E_i$) be the profile of $\mathcal{G}_{u_i}$.

3: Let $W \leftarrow \mathcal{G}_{u_1}$. If $k' = 1$, then go to Step 6; otherwise, let $i \leftarrow 2$.

4: Note that $W$ and $\mathcal{G}_{u_i}$ have one edge in common. A strategy for $W \cup \mathcal{G}_{u_i}$ can be obtained from strategies for $W$ and $\mathcal{G}_{u_i}$ by reaching an accord on the sliding action on the common edge of $W$ and $\mathcal{G}_{u_i}$. Further, $u_1$ and $u_i$ have a pendant edge in $E(\mathcal{C})$ respectively. Compute the profile of $W \cup \mathcal{G}_{u_i}$, which consists of $\pi_{I,I}(W \cup \mathcal{G}_{u_i}), \pi_{I,O}(W \cup \mathcal{G}_{u_i}), \pi_{O,I}(W \cup \mathcal{G}_{u_i}), \pi_{O,O}(W \cup \mathcal{G}_{u_i})$.

5: Let $W \leftarrow W \cup \mathcal{G}_{u_i}$. If $i = k'$, then go to Step 6; otherwise, let $i \leftarrow i + 1$ and go to Step 4.

6: return the profile of $W$.

Algorithm CLEARCACTI3 is used for computing the profile of $G'_v$, which is obtained by adding either one or two pendant edges to $v \in V(G)$. The input of the algorithm includes $G'_v, v$ and $\mathcal{P}$, which includes all the profiles of $\mathcal{G}_{u_1}^1, \ldots, \mathcal{G}_{u_i}^k$. The output of the algorithm is the profile of $G'_v$. 

103
Algorithm 8 ClearCacti3($G'_v, v, P$)

1: If $v$ has one pendant edge in $G'_v$, then let $\pi_I(G'_v)$ be obtained from the output of ClearCacti4($G'_v, 1, 1, P$). Let $\pi_O(G'_v) \leftarrow \pi_I(G'_v) + 1$.

2: If $v$ has two pendant edges in $G'_v$, then

   (i) let $\pi_{I,I}(G'_v)$ be obtained from the output of ClearCacti4($G'_v, 2, 2, P$).

   (ii) let $\pi_{O,O}(G'_v) \leftarrow \pi_{I,I}(G'_v) + 2$.

   (iii) let $\pi_{I,O}(G'_v)$ be obtained from the output of ClearCacti4($G'_v, 0, 1, P$).

   (iv) if $v$ has at most four incident edges in $G'_v$, then let $\pi_{O,I}(G'_v)$ be obtained from the output of ClearCacti4($G'_v, 1, 1, P$) plus one; otherwise, if $v$ has more than four incident edges in $G'_v$, then let $\pi_{O,I}(G'_v)$ be obtained from the output of ClearCacti4($G'_v, 0, 0, P$).

3: return the profile of $G'_v$.

Algorithm ClearCacti4 is called by ClearCacti3 as a subroutine, which computes the profile of $G'_v$. Let $P$ be the set containing the profiles of all the sub-cacti of $G'_v - E_{cut}$ with respect to vertex $v$. We use $\sigma_1$ to record the number of available searchers on $v$ which could be used in an II-strategy or an I-strategy for a sub-cactus. We use $\sigma_2$ to denote the maximum number of searchers residing on $v$ during a strategy for $G'_v$. For simplicity, $\sigma_2$ cannot be greater than 2, and $\sigma_2$ is set to 2 if there exists some moment in a strategy for $G'_v$ at which $v$ contains two or more searchers.
Algorithm 9 ClearCacti4($H'_v, \sigma_1, \sigma_2, \mathcal{P}$)

1: For $i$ from 1 to 7, compute $\chi^i_v$ based on $\mathcal{P}$.

2: If $1 \leq \chi^7_v \leq 2$ and $\sigma_2 \leq 1$, then $\sigma_2 \gets \sigma_2 + 1$. If $\chi^7_v \geq 3$, then $\sigma_2 \gets 2$.

(a) If $\chi^7_v$ is odd, then consider $\sigma_1$. If $\sigma_1 \geq 1$, then (1) let $\frac{\chi^7_v + 1}{2}$ sub-cacti satisfying (7) be cleared by an I-strategy, and let all the other sub-cacti satisfying (7) be cleared by an O-strategy; (2) $\sigma_1 \gets \sigma_1 - 1$. If $\sigma_1 = 0$, then (1) let $\frac{\chi^7_v + 1}{2}$ sub-cacti satisfying (7) be cleared by an O-strategy, and let all the other sub-cacti satisfying (7) be cleared by an I-strategy; (2) $\sigma_1 \gets 1$.

(b) If $\chi^7_v \geq 2$ and is even, then consider $\sigma_1$. If $\sigma_1 = 2$, then (1) let $\frac{\chi^7_v - 2}{2} + 2$ sub-cacti satisfying (7) be cleared by an I-strategy, and let all the other sub-cacti satisfying (7) be cleared by an O-strategy; (2) $\sigma_1 \gets \sigma_1 - 2$. Otherwise, let $\frac{\chi^7_v}{2}$ sub-cacti satisfying (7) be cleared by an I-strategy, and let all the other sub-cacti satisfying (7) be cleared by an O-strategy.
3. If $\chi_v^6 \geq 1$, then $\sigma_2 \leftarrow 2$.

(a) If $\chi_v^6 \geq 2$ and is even, then let $\frac{\chi_v^6}{2}$ sub-cacti satisfying (6) be cleared by an II-strategy, and let all the other sub-cacti satisfying (6) be cleared by an OO-strategy.

(b) If $\chi_v^6$ is odd, then consider $\sigma_1$. If $\sigma_1 = 0$ and there is a sub-cactus satisfying (7) which is cleared by an I-strategy, then (1) let the sub-cactus cleared by an O-strategy; and (2) $\sigma_1 \leftarrow 2$.

(i) $\sigma_1 \geq 1$. If $\sigma_1 = 1$, then place one searcher on $v$. Let $\frac{\chi_v^6+1}{2}$ sub-cacti satisfying (6) be cleared by an II-strategy, and let all the other sub-cacti satisfying (6) be cleared by an OO-strategy. Let $\sigma_1 \leftarrow 0$.

(ii) $\sigma_1 = 0$. Let $\frac{\chi_v^6+1}{2}$ sub-cacti satisfying (6) be cleared by an OO-strategy, and let $\frac{\chi_v^6-1}{2}$ sub-cacti satisfying (6) be cleared by an II-strategy. Let $\sigma_1 \leftarrow 2$. 
4: If $\chi^1_v + \chi^4_v \geq 2$, then $\sigma_2 \leftarrow 2$. If $\chi^1_v + \chi^4_v = 1$ and $\sigma_2 \leq 1$, then $\sigma_2 \leftarrow \sigma_2 + 1$.

(a) If $\sigma_2 = 2$ and $\sigma_1 \leq 1$, then let all the sub-cacti satisfying (1) and (4) be cleared by an OI-strategy, let all the sub-cacti satisfying (2), (3) and (5) be cleared by an IO-strategy. If $\sigma_2 = 2$ and $\sigma_1 = 2$, then select a sub-cactus satisfying (i), where $1 \leq i \leq 5$ and $i$ is the largest, and let it be cleared by an II-strategy; for all the other sub-cacti satisfying (i), where $1 \leq i \leq 5$, let all the sub-cacti satisfying (1) and (4) be cleared by an OI-strategy, let all the sub-cacti satisfying (2), (3) and (5) be cleared by an IO-strategy.

(b) If $\sigma_2 = 1$, then place one searcher on $v$. If $\sigma_1 = 0$, then let all the sub-cacti satisfying (1) and (4) be cleared by an OI-strategy, let all the sub-cacti satisfying (2), (3) and (5) be cleared by an IO-strategy. If $\sigma_1 = 1$, then select a sub-cactus satisfying (i), where $1 \leq i \leq 5$ and $i$ is the largest, and let it be cleared by an II-strategy; for all the other sub-cacti satisfying (i) where $1 \leq i \leq 5$, let all the sub-cacti satisfying (1) and (4) be cleared by an OI-strategy, let all the sub-cacti satisfying (2), (3) and (5) be cleared by an IO-strategy.

(c) If $\sigma_2 = 0$, then consider $\sum_{i=1}^5 \chi_i^v$: (a) If $\sum_{i=1}^5 \chi_i^v = 1$ and $\chi_2^v = 1$, then let the only sub-cactus cleared by an OI-strategy. (b) Otherwise, let one sub-cactus satisfying (i), where $i$ is the largest, be cleared by an OO-strategy. If $\sum_{i=1}^5 \chi_i^v \geq 2$, then let another sub-cactus satisfying (i), where $i$ is the largest, be cleared by an II-strategy. For all the other sub-cacti satisfying (2), (3) and (5), let them be cleared by an IO-strategy.
5: return the total number of searchers that are placed on $v$ and the searchers that are used in all sub-cacti.

5.2.2 Analysis

**Lemma 5.2.5** Consider all the sub-cacti of $G$ with respect to $v$. For any strategy for $G$, there exists an equivalent strategy such that all the sub-cacti are cleared in the following order:

1. all the sub-cacti that are cleared by an O-strategy or an OO-strategy;
2. all the sub-cacti that are cleared by an OI-strategy (for each sub-cactus, perform all actions of searchers in its strategy until one of $v$’s incident edges is cleared);
3. all the sub-cacti that are cleared by an IO-strategy;
4. all the sub-cacti that are cleared by an OI-strategy (for each sub-cactus, perform all actions of searchers in its strategy after one of $v$’s incident edges is cleared);
5. all the sub-cacti that are cleared by an I-strategy or an II-strategy.

**Lemma 5.2.6** For any strategy for $G$, there exists an equivalent strategy in which:

1. if $G_v^i$ satisfies $(1)$, then it is cleared by an OI-strategy or an OO-strategy;
2. if \( G_v^i \) satisfies (2), then it is cleared by an IO-strategy, an OI-strategy or an OO-strategy;

3. if \( G_v^i \) satisfies (3), then it is cleared by an IO-strategy or an OO-strategy;

4. if \( G_v^i \) satisfies (4), then it is cleared by an II-strategy, an OI-strategy or an OO-strategy;

5. if \( G_v^i \) satisfies (5), then it is cleared by an II-strategy, an IO-strategy or an OO-strategy;

6. if \( G_v^i \) satisfies (6), then it is cleared by an II-strategy, or an OO-strategy.

**Proof.** We first consider the case when \( G_v^i \) satisfies (1). Let \( vv_1 \) and \( vv_2 \) be the two incident edges of \( v \) in \( G_v^i \). Let \( S_{I,I} \) be a strategy for \( G \) in which \( G_v^i \) is cleared by an II-strategy. Without loss of generality, assume that \( vv_1 \) is cleared before \( vv_2 \) in \( S_{I,I} \). By using the idea below, \( S_{I,I} \) is converted to an equivalent strategy for \( G \) in which \( G_v^i \) is cleared by an IO-strategy:

1. Let \( x \) denote the moment in \( S_{I,I} \) after which the next sliding action clears \( vv_1 \). Remove all actions from \( S_{I,I} \) that clear edges in \( E(G_v^i) \).

2. Insert all actions of an OI-strategy for \( G_v^i \) immediately after \( x \).

Let \( S_{I,O} \) be a strategy for \( G \) in which \( G_v^i \) is cleared by an IO-strategy. Without loss of generality, assume that \( vv_1 \) is cleared by sliding a searcher from \( v \) to \( v_1 \) in the strategy. By using the idea below, \( S_{I,O} \) is converted to an equivalent strategy for \( G \) in which \( G_v^i \) is cleared by an OI-strategy.
1. Let $x'$ denote the moment in $S_{I,O}$ after which the next sliding action clears $vv_1$. Remove all actions from $S_{I,O}$ that clear edges in $E(G'_v)$.

2. Insert all actions of an OI-strategy for $G'_v$ immediately after $x'$.

In a similar way, we can show the conversions of strategies when $G'_v$ satisfies $(^j)$, where $2 \leq j \leq 6$. 

**Definition 5.2.7** A strategy is called a standard strategy for $G$ with respect to $v$, where $v \in V(G)$, if (1) all the sub-cacti with respect to $v$ are cleared in the order given in Lemma 5.2.5, and (2) each sub-cactus $G'_v$, where $1 \leq i \leq k$, is cleared by a strategy in accordance with Lemma 5.2.6.

In the remainder of this section, we assume that every strategy for $G'_v$ is a standard strategy with respect to $v$.

**Lemma 5.2.8** Function $\text{ClearCacti4} (H'_v, \sigma_1, \sigma_2, \mathcal{P})$ produces an optimal s-strategy for $H'_v$.

**Proof.** We will show that any optimal strategy for $G'_v$ can be converted into an equivalent strategy produced by $\text{ClearCacti4} (H'_v, \sigma_1, \sigma_2, \mathcal{P})$. Let $\mathcal{S}$ denote an optimal strategy for $G'_v$.

**Correctness of Step 2:**

Consider the sliding actions in $\mathcal{S}$ which are related to all the sub-cacti satisfying $(^7)$. If $1 \leq \chi_v^7 \leq 2$ and $\sigma_2 \leq 1$, then we know there must exist a moment in $\mathcal{S}$ at which $v$ is occupied by a searcher that slides to or from a
sub-cactus satisfying (7). Hence, we have $\sigma_2 \leftarrow \sigma_2 + 1$. If $\sigma_v^7 \geq 3$, then we know there must exist a moment in $S$ at which $v$ is occupied by at least two searchers. Therefore, $\sigma_2 \leftarrow 2$. Assume that $\chi_v^7$ is odd and $\sigma_1 \geq 1$. Let $r_1$ be the number of sub-cacti that satisfy (7) and are cleared by an I-strategy in $S$. Note that $\sigma_1 \leq 2$. If $r_1 \neq \frac{\chi_v^7 + 1}{2}$, then there are two possibilities.

Case 1. If $r_1 > \frac{\chi_v^7 + 1}{2}$, then we know there exist at least $2r_1 - \chi_v^7 - \sigma_1$ searchers, which are provided by sub-cacti that satisfy (i) and are cleared by an OO-strategy, where $1 \leq i \leq 6$, or by placing actions. Hence, we can use the following idea to modify $S$: (1) select $r_1 - \frac{\chi_v^7 + 1}{2}$ sub-cacti that satisfy (7) and are cleared by an I-strategy; let them be cleared by an O-strategy; (2) select corresponding sub-cacti that satisfy (i) and are cleared by an OO-strategy, where $1 \leq i \leq 6$; let these sub-cacti be cleared by an II-strategy in the new strategy.

Case 2. If $r_1 < \frac{\chi_v^7 + 1}{2}$, in a similar way, we can modify $S$ such that there are $\frac{\chi_v^7 + 1}{2}$ subcacti satisfying (7) and are cleared by an I-strategy.

Note that (1) $\chi_v^7$ is odd, and (2) $\frac{\chi_v^7 + 1}{2}$ sub-cacti satisfying (7) are cleared by an I-strategy. Hence, we know the number of searchers that can slide to and stay on other sub-cacti satisfying (i), where $1 \leq i \leq 6$, is equal to $\sigma_1 - 1$ after this step. Thus, $\sigma_1 \leftarrow \sigma_1 - 1$. Similarly, we can show the cases when (1) $\chi_v^7$ is odd and $\sigma_1 = 0$, and (2) $\chi_v^7$ is even.

**Correctness of Step 3:**

Consider the sliding actions in $S$ which are related to all the sub-cacti
satisfying (6). If $\chi_v^6 \geq 1$, then we know there must exist a moment in $S$ at which $v$ is occupied by two searchers. Hence, we have $\sigma_2 \leftarrow 2$. We only consider the case when $\chi_v^6 \geq 2$ and is even. The other cases can be proved in a similar way. Let $r_2$ be the number of sub-cacti that satisfy (6) and are cleared by an II-strategy. There are two possibilities:

Case 1. If $r_2 > \frac{\chi_v^6}{2}$, then we know there are at least $2r_2 - \sigma_1$ searchers that are provided by sub-cacti that are cleared by an OO-strategy, or by placing actions. We use the following idea to modify $S$: (1) select $r_2 - \frac{\chi_v^6}{2}$ sub-cacti that satisfy (6) and are cleared by an II-strategy; let them be cleared by an OO-strategy; (2) select the largest $k \leq r_2 - \frac{\chi_v^6}{2}$ sub-cacti that do not satisfy (6) and are cleared by an OO-strategy; let them be cleared by an II-strategy (note that some placing actions may also be removed).

Case 2. If $r_2 < \frac{\chi_v^6}{2}$, then similar to Case 1, we can show the conversion of $S$.

Correctness of Step 4:

Note that we have shown the correctness of Steps 2 and 3, which assign strategies to sub-cacti satisfying (6) and (7). Hence, we only consider all sub-cacti that satisfy ($i$), where $1 \leq i \leq 5$, in the remainder of the proof. Consider the sliding actions in $S$ which are related to all the sub-cacti that satisfy ($i$), where $1 \leq i \leq 5$. Note that a sub-cactus satisfying (1) is cleared either by an OI-strategy or an OO-strategy, and a sub-cactus satisfying (4) is cleared by an II-strategy, an OI-strategy or an OO-strategy. If $\chi^1_v + \chi^4_v \geq 2$, then there must exist a moment in $S$ at which $v$ contains two searchers that slide to or
from the sub-cacti satisfying (1) and (4). If \( \chi_v^1 + \chi_v^4 = 1 \) and \( \sigma_2 \leq 1 \), then there must exist a moment in \( S \) at which a searcher slides to \( v \) from a sub-cactus satisfying (1) or (4).

Assume that \( \sigma_2 = 2 \) and \( \sigma_1 \leq 1 \). If there is any sub-cactus satisfying (1) or (4) that is cleared by an II-strategy, then there are two possibilities: (1) There must be a corresponding sub-cactus satisfying (i) that is cleared by an OO-strategy, where \( 1 \leq i \leq 5 \); or (2) \( 2 - \sigma_1 \) searchers are placed on \( v \). Hence, for each of the sub-cacti, we can make the following modifications to \( S \) and let it clear \( G'_v \): (1) if it satisfies (2), (3), or (5), then let it be cleared by an IO-strategy; (2) if it satisfies (1) or (4), then let it be cleared by an OI-strategy; (3) remove any placing actions on \( v \).

Similarly, we can show the conversion of \( S \) if there is any sub-cactus satisfying (1) or (4) that is cleared by an OO-strategy. Therefore, we can modify \( S \) such that all the sub-cacti satisfying (1) or (4) are cleared by an OI-strategy.

In addition, we can show the conversions of \( S \) in a similar way when (1) \( \sigma_1 = \sigma_2 = 2 \), and (2) \( \sigma_2 \leq 1 \).

\[ \text{Theorem 5.2.9} \quad \text{For any cactus graph } G, \text{ the fast search number of } G \text{ can be computed in linear time by algorithm FastSearchCactus.} \]

**Proof.** The algorithm FastSearchCactus runs in linear time, as we can verify the time complexity as follows:

1. the profile of the sub-cactus with respect to each vertex in \( V(G) \) has constant size;
2. the profile of the sub-cactus with respect to each vertex in $V(G)$ is computed at most once;

3. the profile of the sub-cactus with respect to each vertex in $V(G)$ is passed as parameter at most once when computing the profile of other sub-cactus;

4. the computation of the profile of the sub-cactus with respect to a vertex in $V(G)$ takes constant time.

Obviously, the algorithm FastSearchCacuts computes the fast search number of $G$ in linear time. ■

Theorem 5.2.10 For any cactus graph $G$, we can obtain an optimal fast search strategy in linear time using FastSearchCactus.

Proof. This can be achieved by first using a back-track method to record how every edge of $G$ is cleared after calling FastSearchCactus. In addition, we can record on which vertices of $G$ searchers are placed throughout FastSearchCactus. Based on these records, we can easily obtain an optimal fast search strategy for $G$ by letting those searchers move along edges following the prescribed directions when available. ■

5.3 Cartesian Product of a Tree and an Edge

In what follows, we apply Theorem 5.1.3 to find an optimal fast search strategy for $T \square P_2$, where $T$ has at least three vertices. Let $S_k$, where $k \geq 1$, denote a
star graph with \( k \) leaves. Let \( H_k \) denote a graph that has a cut edge, whose removal results in two copies \( S_k \). Without loss of generality, let \( S_k^1 \) and \( S_k^2 \) denote the two copies of \( S_k \) in \( H_k \). For any pair of edges that are from \( E(S_k^1) \) and \( E(S_k^2) \) respectively, there are four distinct ways to clear them in a fast search strategy for \( H \):

1. both edges are cleared by sliding a searcher from leaf to center node;

2. one of the two edges is cleared by sliding a searcher from leaf to center node, followed by the other edge being cleared by sliding a searcher from center node to leaf;

3. one of the two edges is cleared by sliding a searcher from center node to leaf, followed by the other edge being cleared by sliding a searcher from leaf to center node;

4. both edges are cleared by sliding a searcher from center node to leaf.

For convenience, we use \( II \), \( IO \), \( OI \) and \( OO \) to represent the above four ways respectively in the remainder of this section. Note that there are two layers in \( T \boxtimes P_2 \). Let \( T_1 \) and \( T_2 \) be the two copies of \( T \) in \( T \boxtimes P_2 \). Let \( v_c^1 \in V(T_1) \) be a vertex of degree \( k + 1 \), where \( k \geq 3 \). Let \( v_c^2 \in V(T_2) \) be the vertex where
\[
v_c^2 \sim v_c^1.
\]
Let \( N(v_c^1) = \{v_1^1, \ldots, v_k^1, v_c^2\} \), and let \( N(v_c^2) = \{v_2^2, \ldots, v_k^2, v_c^1\} \). We use \( G'_c \) to denote the connected subgraph of \( T \boxtimes P_2 \) that includes all the edges in \( E(T \boxtimes P_2) \) whose end points include \( v_c^1 \) or \( v_c^2 \). Let \( G_1, \ldots, G_k \) be the connected components after deleting \( E(G'_c) \) from \( T \boxtimes P_2 \). We use \( G'_i \), where \( 1 \leq i \leq k \), to denote the subgraph of \( T \boxtimes P_2 \), which is obtained from \( G_i \) by adding two
pendant edges in $E(T \square P_2)$ that connect vertices in $\{v^1_c, v^2_c\}$ and $V(G_i)$. Recall that there are four ways to clear the two pendant edges of $G'_i$. We use $s_1(G'_i)$ to denote the minimum numbers of searchers needed to be placed on $V(G_i)$ in a strategy for $G'_i$, in which the two pendant edges are cleared by II. In addition, we use $S_1(G'_i)$ to denote a strategy for $G'_i$, in which (1) $s_1(G'_i)$ searchers are placed on $G_i$, and (2) the two pendant edges are cleared by II. In a similar way, for the ways of IO, OI and OO, we define $s_2(G'_i)$, $s_3(G'_i)$ and $s_4(G'_i)$, as well as $S_2(G'_i)$, $S_3(G'_i)$ and $S_4(G'_i)$.

**Lemma 5.3.1** $G'_i$ must have one of the following properties:

1. $s_1(G'_i) = s_2(G'_i) = s_3(G'_i) = s_4(G'_i) - 2$.

2. $s_1(G'_i) = s_2(G'_i) = s_3(G'_i) - 1 = s_4(G'_i) - 2$.

3. $s_1(G'_i) = s_2(G'_i) = s_3(G'_i) - 2 = s_4(G'_i) - 2$.

4. $s_1(G'_i) = s_2(G'_i) - 1 = s_3(G'_i) - 1 = s_4(G'_i) - 2$.

5. $s_1(G'_i) = s_2(G'_i) - 1 = s_3(G'_i) - 2 = s_4(G'_i) - 2$.

6. $s_1(G'_i) = s_2(G'_i) - 2 = s_3(G'_i) - 2 = s_4(G'_i) - 2$.

**Proof.** It is straightforward to see that $s_1(G'_i) + 2 = s_4(G'_i)$. Further, we have $s_2(G'_i) \leq s_3(G'_i)$, as we can convert an OI-strategy into an IO-strategy by simply letting the slide-in action appear after the slide-out action. Similarly, we can show that $s_3(G'_i) \leq s_4(G'_i)$.

We say $G'_i$ is in category $j$ if it satisfies the $j$-th relationship in Lemma 5.3.1.
Similar to the strategies for Cacti graphs, it is easy to see that every fast search strategy for \( T \square P_2 \) can be modified such that all the subgraphs are cleared in the following order: subgraphs cleared by OO, then subgraphs cleared by OI and subgraphs cleared by IO, and finally subgraphs cleared by II. For convenience, we assume that all subgraphs are cleared in this order in any fast search strategies for \( T \square P_2 \). Let \( S \) denote an optimal fast search strategy for \( T \square P_2 \). Let \( \chi_i \) be the number of connected subgraphs that are in category \( i \). Let \( v^1_c \) and \( v^2_c \) be the two internal nodes in \( G'_c \). Consider \( S \). Let \( t_1 \)
Figure 5.13: An example graph $G_5$ in category 5. $s_1(G_5) = 1$, $s_2(G_5) = 2$, $s_3(G_5) = s_4(G_5) = 3$.

denote the earliest moment such that no subgraphs are cleared by OO after $t_1$. Let $t_2$ denote the earliest moment such that no subgraphs are cleared by IO or OI after $t_2$.

Lemma 5.3.2 The number of searchers on $v^1_c$ is the same at $t_1$ and $t_2$, and the number of searchers on $v^2_c$ is the same at $t_1$ and $t_2$.

Lemma 5.3.3 For any subgraph $G'_i$ of $T \square P_2$, we can modify $S$ such that: if $G'_i$ is in category 3, then (1) $G'_i$ is cleared by IO, and (2) the new strategy uses the same number of searchers as $S$ to clear $T \square P_2$.

Proof. There are three cases:

Case 1. If $S_1(G'_i)$ is a part of $S$, then we modify $S$ in the following way: (1) remove all the sliding actions that clear $G'_i$ from $S$; (2) insert all the sliding actions of $S_2(G'_i)$ into the new strategy after $t_1$. After the modification, it is easy to see that the new strategy clears $T \square P_2$ using the same number of searchers as $S$.

Case 2. If $S_3(G'_i)$ is a part of $S$, then we modify $S$ in the following way: (1) place two searchers on the vertex of $G'_i$, whose incident edge connecting to $G$ is cleared by sliding a searcher from it to $G_i$ in $S$; (2) remove all the sliding
actions that clear $G'_i$ from $S$; (3) insert all the sliding actions of $S_2(G'_i)$ into the new strategy after $t_1$. After the modification, it is easy to see that the new strategy clears $T \square P_2$ using the same number of searchers as $S$.

Case 3. If $S_4(G'_i)$ is a part of $S$, then consider the sliding actions that appear before $t_1$ in $S$. We modify $S$ in the following way: (1) remove all the sliding actions that clear $G'_i$ from $S$; (2) place two searchers on $v^1_c$ in the beginning of the strategy; (3) insert all the sliding actions of $S_2(G'_i)$ after $t_1$. Obviously, after the modifications, the new strategy uses the same number of searchers as $S$ to clear $T \square P_2$.

**Lemma 5.3.4** Suppose that $\chi_i = 0$, for $i = 1, 2, 3, 6$. If $\chi_4 + \chi_5 \geq 2$, then we can modify $S$ such that: (1) $\left\lfloor \frac{\chi_4 + \chi_5}{2} \right\rfloor$ subgraphs in categories 4 and 5 are cleared by $II$, (2) another $\left\lfloor \frac{\chi_4 + \chi_5}{2} \right\rfloor$ subgraphs in categories 4 and 5 are cleared by $OO$, and (3) the new strategy uses the same number of searchers as $S$ to clear $T \square P_2$.

**Proof.** Without loss of generality, assume $G'_i$ is a subgraph in category 4 or 5. We first show some properties of the fast search strategies for $G'_i$. There are two cases:

Case 1. $G'_i$ is in category 4. Note that $s_2(G'_i) = s_3(G'_i)$. If $S_2(G'_i)$ is a part of $S$, then we can modify $S$ such that: (1) $G'_i$ is cleared by $S_3(G'_i)$ in the new strategy, and (2) the new strategy uses the same number of searchers as $S$ to clear $T \square P_2$. 

119
Case 2. $G'_i$ is in category 5. Note that $s_3(G'_i) = s_4(G'_i)$. If $S_3(G'_i)$ is a part of $S$, then we can modify $S$ such that: (1) $G'_i$ is cleared by $S_4(G'_i)$ in the new strategy, and (2) the new strategy uses the same number of searchers as $S$ to clear $T \Box P_2$.

In the next, we consider all those subgraphs that are cleared between $t_1$ and $t_2$. If there are two subgraphs $G'_i$ and $G'_j$ in categories 4 and 5 that are cleared between $t_1$ and $t_2$, then we modify the strategy in the following way: (1) remove from the strategy all the sliding actions that clear $G'_i$ and $G'_j$; (2) insert all the sliding actions of $S_1(G'_i)$ before $t_1$; (3) insert all the sliding actions of $S_4(G'_j)$ after $t_2$. Repeat the above process until there is at most one subgraph in categories 4 and 5, which is cleared between $t_1$ and $t_2$ in the new strategy.

After taking the above actions, if there are $x < \left\lfloor \frac{x_4 + x_5}{2} \right\rfloor$ subgraphs in categories 4 and 5 that are cleared before $t_1$, then we know there must be $2 \left\lfloor \frac{x_4 + x_5}{2} \right\rfloor - x$ subgraphs that are cleared after $t_2$. Further, from Lemma 5.3.2, we know that at least $\left\lfloor \frac{x_4 + x_5}{2} \right\rfloor - x$ searchers must be initially placed on $v_1^c$ and $v_2^c$ respectively. Thus, we first arbitrarily select $\left\lfloor \frac{x_4 + x_5}{2} \right\rfloor - x$ subgraphs that are cleared after $t_2$. For each selected subgraph $G'_i$, we do the following modifications: (1) remove from the strategy all the sliding actions that clear $G'_i$; (2) remove from the strategy the placing actions that place a searcher on $v_1^c$ and $v_2^c$; (3) insert all the placing and sliding actions of $S_4(G'_i)$ before $t_1$. Similarly, if there are $x < \left\lfloor \frac{x_4 + x_5}{2} \right\rfloor$ subgraphs in categories 4 and 5 that are cleared after $t_2$, then we modify the strategy such that $\left\lfloor \frac{x_4 + x_5}{2} \right\rfloor$ subgraphs are cleared after $t_2$. Therefore, it is easy to see that there is at most one subgraph in categories 4 and 5 that is cleared between $t_1$ and $t_2$. [ ]
Lemma 5.3.5  For each connected subgraph $G'_i$, where $1 \leq i \leq k$, if we know $s_1(G'_i)$, $s_2(G'_i)$, $s_3(G'_i)$ and $s_4(G'_i)$, then we can compute the minimum number of searchers for clearing $T\Box P_2$.

Proof.  We give rules for assigning strategies to all the subgraphs:

1. Let subgraphs in category 1 be cleared by OI, and let subgraphs in categories 2 and 3 be cleared by IO.

2. If $\chi_4 + \chi_5 + \chi_6 \geq 2$, then consider the following two cases:
   
   (a) $\chi_4 + \chi_5 + \chi_6$ is even. Let half of the subgraphs in categories 4, 5 and 6 be cleared by II and the other half of the subgraphs be cleared by OO. If (1) $\chi_1 = 0$ and $\chi_4 + \chi_5 + \chi_6 = 2$, or (2) $\chi_1 + \chi_2 + \chi_3$ is even, then we place one searcher on $v^1_c$.

   (b) $\chi_4 + \chi_5 + \chi_6 \geq 2$ is odd. Let $G'_i$ denote a subgraph in category $j$, where $4 \leq j \leq 6$ and $j$ is as small as possible. For all the subgraphs in categories 4, 5 and 6 except $G'_i$, let half of them be cleared by II and the other half be cleared by OO. Considering $G'_i$, there are three cases:

   i. $G'_i$ is in category 4. If $\chi_1 + \chi_2 + \chi_3$ is odd, then place a searcher on $v^1_c$. Let $G'_i$ be cleared by OI.

   ii. $G'_i$ is in category 5. If (1) $\chi_1 = 0$ and $\chi_4 + \chi_5 + \chi_6 = 3$ or (2) $\chi_1 + \chi_2 + \chi_3$ is odd, then place one searcher on $v^1_c$. Let $G'_i$ be cleared by IO.

   iii. $G'_i$ is in category 6. Let $G'_i$ be cleared by OO.
3. If $\chi_4 + \chi_5 + \chi_6 = 1$, then we let $G'_i$ be the only subgraph in categories 4, 5 and 6. There are three cases:

(a) $\chi_4 = 1$. If $\chi_1 \geq 2$, then we let $G'_i$ be cleared by OI; further, if $\chi_1 + \chi_2 + \chi_3 + \chi_4$ is even, then we place one searcher on $v^1_c$. If $\chi_1 = 1$, then we let $G'_i$ be cleared by OO. If $\chi_1 = 0$, then we let $G'_i$ be cleared by OO, and place one searcher on $v^1_c$.

(b) $\chi_5 = 1$. If $\chi_1 \geq 3$, then we let $G'_i$ be cleared by IO; further, if $\chi_1 + \chi_2 + \chi_3 + \chi_5$ is even, then we place one searcher on $v^1_c$. If $1 \leq \chi_1 \leq 2$, then we let $G'_i$ be cleared by OO. If $\chi_1 = 0$, then we let $G'_i$ be cleared by OO and place one searcher on $v^1_c$.

(c) $\chi_6 = 1$. Let $G'_i$ be cleared by OO. If $\chi_1 = 0$, then we place one searcher on $v^1_c$.

4. If $\chi_4 + \chi_5 + \chi_6 = 0$, then we have three cases:

(a) $\chi_1 = 2$ or $\chi_1 + \chi_2 + \chi_3$ is even. Place a searcher on $v^1_c$.

(b) $\chi_1 = 1$. Place a searcher on $v^1_c$ and $v^2_c$ respectively.

(c) $\chi_1 = 0$. Place two searchers on $v^1_c$, and place one searcher on $v^2_c$.

In the next, we give a general strategy for clearing $T \Box P_2$.

1. Assign strategies to all the subgraphs according to the rules given above.

2. Clear all the subgraphs that are supposed to be cleared by OO. After each of the subgraphs is cleared, a searcher will reside on $v^1_c$ and $v^2_c$ respectively.
3. Clear all the subgraphs that are supposed to be cleared by IO and OI. In the meantime, use one searcher to clear the edge $v_1^1v_2^2$.

4. Slide searchers on $v_1^1$ and $v_2^2$ to clear all the remaining subgraphs, which are supposed to be cleared by II.

The fast search number of $T\square P_2$ can be computed by using a recursive method. 

Let $G_j$ be a connected subgraph, where $1 \leq j \leq k$. Let $G' = G \setminus G_j$.

**Lemma 5.3.6** For each connected subgraph $G_i'$ of $G'$ with respect to $v_1^1$ and $v_2^2$, where $1 \leq i \neq j \leq k$, if we know $s_1(G'_i)$, $s_2(G'_i)$, $s_3(G'_i)$ and $s_4(G'_i)$, then we can compute $s_1(G')$, $s_2(G')$, $s_3(G')$ and $s_4(G')$.

**Proof.** Similar to the proof of Lemma 5.3.5, we can show that there must exist an optimal fast search strategy for $G'$, which satisfies:

1. subgraphs in category 1 are cleared by OI, subgraphs in categories 2 and 3 are cleared by IO.

2. there are $\left\lfloor \frac{\chi_4 + \chi_5 + \chi_6}{2} \right\rfloor$ subgraphs in categories 4, 5 and 6 are cleared by II and another $\left\lfloor \frac{\chi_4 + \chi_5 + \chi_6}{2} \right\rfloor$ subgraphs in categories 4, 5 and 6 are cleared by OO.

For convenience, if $\chi_4 + \chi_5 + \chi_6 \geq 1$, then we let $G'_i$ be a subgraph in category $j$, where $4 \leq j \leq 6$ and $j$ is the minimal. In addition, if $\chi_4 + \chi_5 + \chi_6$ is even, then let half of all the subgraphs in categories 4, 5 and 6 cleared by II.
and let the other half be cleared by OO. If $\chi_4 + \chi_5 + \chi_6$ is odd, then let half of all the subgraphs in categories 4, 5 and 6 except $G'_l$ be cleared by II, and let the other half be cleared by OO.

To compute $s_1(G')$, we need to know the number of searchers that are placed on $v^1_c$ and $v^2_c$, as well as the number of searchers required for clearing all adjacent connected subgraphs. Let the two pendant edges of $G'$ be cleared by II. There are three cases:

Case 1. $\chi_4 + \chi_5 + \chi_6 = 0$. If $\chi_1 = 0$, then place a searcher on $v^1_c$.

Case 2. $\chi_4 + \chi_5 + \chi_6 = 1$. Let $G'_l$ be cleared by II. If $\chi_1 = 0$, then place a searcher on $v^1_c$.

Case 3. $\chi_4 + \chi_5 + \chi_6 \geq 2$. If $\chi_4 + \chi_5 + \chi_6$ is odd, then let $G'_l$ be cleared by II; Further, if $\chi_1 + \chi_2 + \chi_3 \geq 2$ is even, then (1) place a searcher on $v^1_c$.

To compute $s_2(G')$, let the two pendant edges of $G'$ be cleared by IO. There are three cases.

Case 1. $\chi_4 + \chi_5 + \chi_6 = 0$. If $v^1_c$ is odd or $\chi_1 = 1$, then place a searcher on $v^1_c$; else, if $\chi_1 = 0$, then place one searcher on $v^1_c$ and $v^2_c$ respectively.

Case 2. $\chi_4 + \chi_5 + \chi_6 = 1$. We have three subcases.

Case 2.1. If $\chi_4 = 1$, then let $G'_l$ be cleared by OI. If $v^1_c$ is odd or $\chi_1 = 0$, then place a searcher on $v^1_c$.

Case 2.2. If $\chi_5 = 1$, then let $G'_l$ be cleared by IO. If $v^1_c$ is odd or $\chi_1 = 1$, then place a searcher on $v^1_c$; else, if $\chi_1 = 0$, then place one searcher on $v^1_c$ and $v^2_c$ respectively.
Case 2.3. If $\chi_6 = 1$, then let $G'_l$ be cleared by OO.

Case 3. $\chi_4 + \chi_5 + \chi_6 \geq 2$. If $\chi_4 + \chi_5 + \chi_6$ is odd, then let $G'_l$ be cleared by OI if $G'_l$ is in category 4, or let $G'_l$ be cleared by IO if $G'_l$ is in category 5, or let $G'_l$ be cleared by OO if $G'_l$ is in category 6. If $v^1_c$ is odd, then place one searcher on it.

To compute $s_3(G')$, let the two pendant edges of $G'$ be cleared by OI. There are three cases.

Case 1. $\chi_4 + \chi_5 + \chi_6 = 0$. If $v^1_c$ is odd and $\chi_1 \geq 3$ or $\chi_1 = 2$, then place a searcher on $v^1_c$; otherwise, if $\chi_1 = 1$, then place one searcher on $v^1_c$ and $v^2_c$ respectively; otherwise, if $\chi_1 = 0$, then place two searchers on $v^1_c$ and place one searcher on $v^2_c$.

Case 2. $\chi_4 + \chi_5 + \chi_6 = 1$. There are three subcases.

Case 2.1. $\chi_4 = 1$. Let $G'_l$ be cleared by OI. If $v^1_c$ is odd and $\chi_1 \geq 2$ or $\chi_1 = 1$, then place a searcher on $v^1_c$; otherwise, if $\chi_1 = 0$, then place a searcher on $v^1_c$ and $v^2_c$ respectively.

Case 2.2. $\chi_5 = 1$. Let $G'_l$ be cleared by IO. This case is similar to Case 1.

Case 2.3. $\chi_6 = 1$. Let $G'_l$ be cleared by OO. If $\chi_1 = 0$, then place one searcher on $v^1_c$.

Case 3. $\chi_4 + \chi_5 + \chi_6 \geq 2$. If (1) $\chi_4 + \chi_5 + \chi_6 = 2$ and $\chi_1 = 0$, or (2) $v^1_c$ is odd, then place a searcher on $v^1_c$. If $\chi_4 + \chi_5 + \chi_6$ is odd, then there are three subcases.
Case 3.1. $G'_t$ is in category 4. Let $G'_t$ be cleared by OI. If $v^1_c$ is odd, then place a searcher on $v^1_c$.

Case 3.2. $G'_t$ is in category 5. Let $G'_t$ be cleared by IO. If $v^1_c$ is odd, then place a searcher on $v^1_c$.

Case 3.3. $G'_t$ is in category 6. Let $G'_t$ be cleared by OO.

To compute $s_4(G')$, let the two pendant edges of $G'$ be cleared by OO.

There are three cases.

Case 1. $\chi_4 + \chi_5 + \chi_6 = 0$. If $v^1_c$ is odd, then place a searcher on $v^1_c$ and $v^2_c$ respectively; if $v^1_c$ is even, then place two searchers on $v^1_c$. If $\chi_1 = 0$, then place another searcher on $v^1_c$.

Case 2. $\chi_4 + \chi_5 + \chi_6$ is odd. Let $G'_t$ be cleared by OO. If (1) $\chi_4 + \chi_5 + \chi_6 = 1$, and (2) $\chi_1 = 0$ or $v^1_c$ is odd, then place one searcher on $v^1_c$.

Case 3. $\chi_4 + \chi_5 + \chi_6 \geq 2$ and $\chi_4 + \chi_5 + \chi_6$ is even. If $v^1_c$ is odd, then place a searcher on $v^1_c$ and $v^2_c$ respectively; if $v^1_c$ is even, then place two searchers on $v^1_c$.

**Theorem 5.3.7** An optimal fast search strategy for $T\Box P_2$ can be found in polynomial time.

**Proof.** We give a strategy below for finding an optimal fast search strategy for $T\Box P_2$.

1. Arbitrarily select a pair of vertices $v$ and $v'$, where $v'$ is the corresponding vertex of $v$ in $T\Box P_2$. 

126
Figure 5.14: The above are three connected subgraphs of $T \square P_2$. Let $G'_1$, $G'_c$ and $G'_2$ denote the three connected subgraphs from left to right. Let $G' = G'_1 \cup G'_c \cup G'_2$. It is easy to see that $s_1(G'_1) = s_1(G'_2) = 1$, $s_2(G'_1) = s_2(G'_2) = 2$, $s_3(G'_1) = s_3(G'_2) = 3$, and $s_4(G'_1) = s_4(G'_2) = 3$. By using our strategies given above, we have $s_1(G') = 4$, $s_2(G') = 4$, $s_3(G') = 5$ and $s_4(G') = 6$.

2. For each of the connected subgraphs that are neighbors of $v$ and $v'$, iteratively use the strategies in Lemma 5.3.6 to find its profile.

3. Compute the optimal fast search number of $T \square P_2$ based on the profiles of all its neighbor connected subgraphs. An optimal fast search strategy for $T \square P_2$ is produced by merging all the placing and sliding actions based on the profiles used above.

Clearly, the above strategy can find an optimal fast search strategy for $T \square P_2$ in polynomial time. □
Chapter 6

A Partition Method for the Zero-Visibility Cops and Robber Game

In this chapter, we proposed a new partition method that can be used to prove lower bounds for the zero-visibility cops and robber game. We first study a partition problem. Using such partitions, we then investigate the cop number and monotonic cop number of graph products.

6.1 Partitions and Boundaries of Vertex Sets

Let $G = (V, E)$ be a graph with vertex set $V$ and edge set $E$. A pair of subsets $(V_1, V_2)$ is a partition of $V$ if $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. For a partition $(V_1, V_2)$ of $V$, the boundary of $V_1$, denoted $\partial V_1$, is the largest subset of $V_1$ such that each vertex in the subset is adjacent to a vertex in $V_2$ on $G$. 
Theorem 6.1.1 Let $P_m \square P_n$, $n \geq m \geq 2$, be a grid with $m$ rows and $n$ columns. Let $(V_1, V_2)$ be a partition of the vertex set of $P_m \square P_n$. If $|\partial V_1| < m$ and there exists a row whose vertices all belong to $V_2$, then

(i) for any $\ell \in \{1, \ldots, m-1\}$, the number of rows that contain at least $m-\ell$ vertices in $V_1$ is at most $\ell$;

(ii) $|V_1| \leq \frac{m^2-m}{2}$.

Proof. (i) For the sake of contradiction, assume that there exists $\ell'$, $1 \leq \ell' \leq m-1$, such that the number of rows that contain at least $m-\ell'$ vertices in $V_1$ is at least $\ell' + 1$. Let $i_1$ be the index of a row in which all vertices are in $V_2$. Let $i_2$ be the index of a row that contains at least $m-\ell'$ vertices in $V_1$. Without loss of generality, suppose that there is no row between the $i_1$-th row and the $i_2$-th row which contains at least $m-\ell'$ vertices in $V_1$. Let $v_{i_2,j}$ be the vertex of $V_1$ on the $i_2$-th row and $j$-th column, and let $P' = v_{i_2,j} \ldots v_{i_1,j}$ be a subcolumn of the $j$-th column. Note that all the vertices in the $i_1$-th row are in $V_2$. Hence, there must exist a vertex $v'$ of $P'$ such that $v' \in \partial V_1$. Thus, there are at least $m-\ell'$ vertices in all the subcolumns from the $i_1$-th row to the $i_2$-th row, which belong to $\partial V_1$. Similar to the above, we can also show that there is no row containing at least $m$ vertices in $V_1$; otherwise, $|\partial V_1|$ is at least $m$ and this contradicts the condition that $|\partial V_1| < m$. So each row contains at most $m-1$ vertices in $V_1$; and furthermore, if a row contains vertices in $V_1$, it also contains a vertex in $\partial V_1$. Note that on the $i_2$-th row, the rightmost vertex which belongs to $\partial V_1$ is counted twice, and this is the only doubly counted vertex in the above counting. Hence, $|\partial V_1| \geq (m-\ell') + (\ell' + 1) - 1 = m$, which contradicts the
condition that $|\partial V_1| < m$. Therefore, for any $\ell \in \{1, \ldots, m-1\}$, there are at most $\ell$ rows that contain $m-\ell$ or more vertices in $V_1$ each.

(ii) From (i), each row contains at most $m-1$ vertices in $V_1$, and moreover, the number of rows that contain at least $m-\ell$ vertices in $V_1$ is at most $\ell$, for each $\ell \in \{1, \ldots, m-1\}$. Therefore, $|V_1| \leq \sum_{i=1}^{m-1} i = \frac{m^2-m}{2}$.

Theorem 6.1.2 Let $C_m \square P_n$ be a cylinder grid, where $m \geq 2n \geq 4$. Let $(V_1, V_2)$ be a partition of the vertex set of $C_m \square P_n$. If $|\partial V_1| < 2n$ and there exists a copy of $P_n$ whose vertices are all in $V_2$, then

(i) for any $\ell \in \{1, \ldots, n\}$, the number of copies of $C_m$ that contain at least $2n-2\ell+1$ vertices in $V_1$ is at most $\ell$;

(ii) for any $\ell \in \{2, \ldots, n\}$, the number of copies of $C_m$ that contain at least $2n-2\ell+2$ vertices in $V_1$ is at most $\ell-1$;

(iii) each copy of $C_m$ contains at most $2n-1$ vertices in $V_1$;

(iv) $|V_1| \leq n^2$.

Proof. Let $P_n^z$ be a copy of $P_n$ whose vertices all belong to $V_2$. If each copy of $C_m$ contains at least two vertices in $V_1$, then $|\partial V_1| \geq 2n$, which is a contradiction. Hence, there must exist a copy of $C_m$, say $C_m^{i_1}$, which contains at most one vertex in $V_1$.

(i) The claim is trivial when $\ell = n$. So we only consider the case where $1 \leq \ell \leq n-1$. Let $C_m^{i_2}$ be a copy of $C_m$ that contains at least $2n-2\ell+1$ vertices in $V_1$. Without loss of generality, we assume that there is no copy of
$C_m$ between $C_{i_1}^m$ and $C_{i_2}^m$ that contains at least $2n - 2\ell + 1$ vertices in $V_1$. Let $v_{i_2,j}$ be a vertex in $C_{i_2}^m$ such that $v_{i_2,j} \in V_1$. Let $P' = v_{i_1,j} \ldots v_{i_2,j}$ be a subpath of $P_n^j$. Note that there are at least $m - 1$ vertices in $C_{i_1}^m$ that belong to $V_2$. There must exist a vertex on $P'$ that is in $\partial V_1$. Hence, at least $2n - 2\ell + 1$ vertices in the subpaths from $C_{i_1}^m$ to $C_{i_2}^m$ belong to $\partial V_1$ (see Figure 6.1)

If there exist at least $\ell + 1$ copies of $C_m$ containing at least $2n - 2\ell + 1$ vertices in $V_1$, then each of the $\ell + 1$ copies of $C_m$ contains at least $2n - 2\ell + 1 \geq 2n - 2(n - 1) + 1 = 3$ vertices in $V_1$, and at least two of them are in $\partial V_1$. Hence, $|\partial V_1| \geq (2n - 2\ell + 1) + (2\ell) = 2n + 1$, which contradicts the condition that $|\partial V_1| < 2n$. Therefore, we know that there are at most $\ell$ copies of $C_m$ containing at least $2n - 2\ell + 1$ vertices in $V_1$.

![Figure 6.1](image)

Figure 6.1: Let $\ell = 3$, $i_1 = 2$ and $i_2 = 4$. $C_m^4$ contains three vertices $v_{4,2}$, $v_{4,3}$ and $v_{4,4}$ in $V_1$. Each of the paths $v_{2,2}v_{3,2}v_{4,2}$, $v_{2,3}v_{3,3}v_{4,3}$ and $v_{2,4}v_{3,4}v_{4,4}$ contains a vertex in $\partial V_1$.

(ii) For the sake of contradiction, assume that there are at least $\ell'$, $2 \leq \ell' \leq n$, copies of $C_m$ that contain $2n - 2\ell' + 2$ or more vertices of $V_1$ each. If $\ell' = n$, then there are $n$ copies of $C_m$ that contain two or more vertices of $V_1$ each. Note that all vertices of $P_n^i$ are in $V_2$. Hence, each copy of $C_m$ contains at least two vertices that are in $\partial V_1$. So $|\partial V_1| \geq 2n$ and this contradicts the condition
that $|\partial V_1| < 2n$. Thus, there are at most $n - 1$ copies of $C_m$ that contain at least two vertices in $V_1$. We now consider the case where $2 \leq \ell' \leq n - 1$.

Suppose that $C_m^{i_3}$ contains at least $2n - 2\ell' + 2$ vertices in $V_1$ satisfying that there is no copy of $C_m$ between $C_m^{i_1}$ and $C_m^{i_3}$ that contains at least $2n - 2\ell' + 2$ vertices in $V_1$. Note that at least $2n - 2\ell' + 2$ vertices in all subpaths from $C_m^{i_1}$ to $C_m^{i_3}$ are in $\partial V_1$. On the other hand, there are at least $\ell'$ copies of $C_m$ that contain at least $2n - 2\ell' + 2 \geq 2n - 2(n-1) + 2 = 4$ vertices of $V_1$ each. It follows that each of these cycles contains at least two vertices in $\partial V_1$ because $P_n^z$ only contains vertices that are in $V_2$.

Note that on $C_m^{i_3}$, two vertices which belong to $\partial V_1$ are counted twice, and they are the only doubly counted vertices. Thus, $|\partial V_1| \geq (2n - 2\ell' + 2) + (2\ell') - 2 = 2n$, which is a contradiction. So the claim holds.

(iii) Assume some $C_m^{i_1}$ contains at least $2n$ vertices in $V_1$. Since $C_m^{i_1}$ contains at most one vertex in $V_1$, there are at least $2n$ vertices in all subpaths from $C_m$ to $C_m^{i_1}$, which are in $\partial V_1$. This is a contradiction.

(iv) From (i) – (iii), we have $|V_1| \leq \sum_{\ell=1}^{n}(2n - 2\ell + 1) = n^2$.

**Theorem 6.1.3** Let $C_m \square P_n$ be a cylinder grid, where $3 \leq m < 2n$. Let $(V_1, V_2)$ be a partition of the vertex set of $C_m \square P_n$. If $|\partial V_1| < m$ and there exists a copy of $P_n$ whose vertices all belong to $V_2$, then

(i) when $m$ is even, for any $\ell \in \{0, \ldots, \frac{m-2}{2}\}$, the number of copies of $P_n$ that contain at least $\frac{m}{2} - \ell$ vertices in $V_1$ is at most $2\ell + 1$;

(ii) when $m$ is odd, for any $\ell \in \{1, \ldots, \frac{m-1}{2}\}$, the number of copies of $P_n$ that contain at least $\frac{m+1}{2} - \ell$ vertices in $V_1$ is at most $2\ell$;
(iii) $|V_1| \leq \frac{m^2}{4}$ when $m$ is even; and $|V_1| \leq \frac{m^2-1}{4}$ when $m$ is odd.

**Proof.** Let $P_{n}^{j_1}$ be a path whose vertices all belong to $V_2$. If there are two or more copies of $P_n$ containing $n$ vertices in $V_1$, then $|\partial V_1| \geq 2n > m$, which is a contradiction. Hence, there is at most one copy of $P_n$ that contains $n$ vertices in $V_1$. If there exists a copy of $P_n$ containing $n$ vertices in $V_1$, since there is at most one copy of $P_n$ that contains $n$ vertices in $V_1$, this path must contain at least one vertex in $\partial V_1$. For any copy of $P_n$ containing at most $k$ vertices in $V_1$, where $1 \leq k \leq n - 1$, since it contains vertices in both $V_1$ and $V_2$, it must contain vertices in $\partial V_1$. Therefore, if a copy of $P_n$ contains vertices in $V_1$, then it must contain a vertex in $\partial V_1$.

(i) For the sake of contradiction, assume that there exists $\ell' \in \{0, \ldots, \frac{m-2}{2}\}$, such that the number of copies of $P_n$ that contain at least $\frac{m}{2} - \ell'$ vertices in $V_1$ is at least $2\ell' + 2$. If $\ell' = 0$, then there exist at least two copies of $P_n$ containing at least $\frac{m}{2}$ vertices of $V_1$, and thus, $|\partial V_1| \geq m$. This is a contradiction. So we only consider the case where $\ell' \in \{1, \ldots, \frac{m-2}{2}\}$. Let $P_{n}^{j_2}$ and $P_{n}^{j_3}$ be two paths, each of which contains at least $\frac{m}{2} - \ell'$ vertices in $V_1$ (see Figure 6.2), such that $P_{n}^{j_2}$ and $P_{n}^{j_3}$ partition all copies of $P_n$ into two sets $\mathcal{P}_1$ and $\mathcal{P}_2$, satisfying that (1) each copy of $P_n$ in $\mathcal{P}_1$ contains at most $\frac{m}{2} - \ell' - 1$ vertices in $V_1$, (2) $\mathcal{P}_1$ contains all consecutive paths between $P_{n}^{j_2}$ and $P_{n}^{j_3}$ which include $P_{n}^{j_1}$, and (3) $\mathcal{P}_2$ contains $P_{n}^{j_2}$ and $P_{n}^{j_3}$ and all consecutive paths between them which do not include $P_{n}^{j_1}$. Recall that if a copy of $P_n$ contains vertices in $V_1$, then it must contain a vertex in $\partial V_1$. Since there are $2\ell' + 2$ copies of $P_n$ in $\mathcal{P}_2$ that contain at least $\frac{m}{2} - \ell'$ vertices in $V_1$, we know $\mathcal{P}_2$ contains at least $2\ell' + 2$ vertices in $\partial V_1$. Further, consider the set $\mathcal{P}_1 \cup \{P_{n}^{j_2}, P_{n}^{j_3}\}$. All paths on copies of $C_m$ from
$P_{n}^{j_2}$ to $P_{n}^{j_1}$ contain at least $\frac{m}{2} - \ell'$ vertices in $\partial V_1$, and all paths on copies of $C_m$ from $P_{n}^{j_3}$ to $P_{n}^{j_1}$ also contain at least $\frac{m}{2} - \ell'$ vertices in $\partial V_1$. Since at most two vertices of $\partial V_1$ are counted twice, we have $|\partial V_1| \geq (2\ell' + 2) + 2(\frac{m}{2} - \ell') - 2 = m$, which is a contradiction.

Figure 6.2: Let $\ell' = 1$, $j_1 = 3$, $j_2 = 1$ and $j_3 = 5$. $P_{n}^{1}$, $P_{n}^{5}$ and $P_{n}^{6}$ contain two vertices in $V_1$ respectively. $P_{n}^{1}$ and $P_{n}^{5}$ partition all the copies of $P_n$ into two sets $\mathcal{P}_1$ and $\mathcal{P}_2$: $\mathcal{P}_1$ contains $P_{n}^{2}$, $P_{n}^{3}$ and $P_{n}^{4}$, and $\mathcal{P}_2$ contains $P_{n}^{1}$, $P_{n}^{5}$ and $P_{n}^{6}$. Note that $P_{n}^{1}$, $P_{n}^{2}$ and $P_{n}^{5}$ contain five vertices in $V_1$ and four of them are in $\partial V_1$.

(ii) When $m$ is odd, the proof is similar to the proof of (i).

(iii) We first show that there are exactly $m - 1$ copies of $P_n$ that contain vertices of $V_1$ when $|V_1|$ is maximal. From the partition $(V_1, V_2)$, we can construct a new partition $(V'_1, V'_2)$ of $V(C_m \Box P_n)$ such that for each $P_{n}^{i}$ $(1 \leq i \leq m)$, $|V(P_{n}^{i}) \cap V'_1| = |V(P_{n}^{i}) \cap V_1|$ and all vertices in $V(P_{n}^{i}) \cap V'_1$ form a subpath of $P_{n}^{i}$ with one end on $C_m$. It is easy to see that $|V'_1| \leq |V_1|$ and $|\partial V'_1| \leq |\partial V_1|$. If there are at most $m - 2$ copies of $P_n$ that contain vertices of $V_1$, we can construct another partition $(V''_1, V''_2)$ of $V(C_m \Box P_n)$ such that $|\partial V''_1| = m - 1$ and $|V''_1| \geq |V_1| + 1$. This contradicts the maximality of $|V_1|$. So, when $|V_1|$ is maximal, $m - 1$ copies of $P_n$ contain vertices of $V_1$, and each of them has exactly one vertex in $\partial V_1$. We now have two cases.
Case 1: $m$ is even. If there is some $P_i^n$ that contains at least $\frac{m^2}{2} + 1$ vertices of $V_1$, then $P_i^n$ must contain two or more vertices of $\partial V_1$. This is a contradiction because when $|V_1|$ is maximal, each copy of $P_n$ except $P_{i_1}^n$ contains exactly one vertex of $\partial V_1$. Thus, each copy of $P_n$ contains at most $\frac{m^2}{2}$ vertices in $V_1$. From (i) we have $|V_1| \leq \frac{m^2}{2} + 2 \sum_{\ell=1}^{\frac{m^2}{2}} \left( \frac{m^2}{2} - \ell \right) = \frac{m^2}{4}$.

Case 2: $m$ is odd. If there exist two or more copies of $P_n$ containing $n$ vertices from $V_1$, then $|\partial V_1| \geq 2n > m$ which is a contradiction. From (ii), there are at most two copies of $P_n$ that contain at least $\frac{m+1}{2} - 1$ vertices in $V_1$. If there exists exactly one path $P_i^n$ that contains at least $\frac{m+1}{2}$ vertices of $V_1$, then since there are at most two copies of $P_n$ that contain at least $\frac{m+1}{2} - 1$ vertices in $V_1$, there is a neighboring path of $P_i^n$ that contains at most $\frac{m+1}{2} - 2$ vertices in $V_1$. Hence, $P_i^n$ must contain at least two vertices in $\partial V_1$. This is a contradiction. Therefore, each copy of $P_n$ contains at most $\frac{m-1}{2}$ vertices in $V_1$. From (ii) we have $|V_1| \leq 2 \sum_{\ell=1}^{\frac{m-1}{2}} \left( \frac{m+1}{2} - \ell \right) = \frac{m^2-1}{4}$.  

6.2 Lower Bounds

Consider the zero-visibility cops and robber game on a graph $G = (V, E)$. After any turn, the set of cleared vertices $V_1$ and the set of contaminated vertices $V_2$ form a partition of $V$. So the boundary of $V_1$ is the largest subset of $V_1$ such that each cleared vertex in the subset is adjacent to a contaminated vertex in $V_2$. Let $V_B^i, i \geq 1$, denote the boundary of $V_1$ after the cops’ turn in round $i$, and let $V_C^i$ be the set of cleared vertices after the robber’s turn in round $i$. Note that $V_B^i$ might not be a subset of $V_C^i$. After the cops’ turn in
round $i$, a vertex is called *non-boundary* if it is not in $V^i_B$.

**Lemma 6.2.1** Let $G$ be a graph with $c_0(G) = k$. For any round $i$ ($i \geq 1$) of an optimal cop-win strategy for $G$, if $|V^i_B| \geq 2k$, then $|V^i_C| \leq |V^{i-1}_C|$.

**Proof.** Since there are $k$ cops, at most $k$ contaminated vertices can be cleared by cops in each round. Note that at most $k$ vertices in $V^i_B$ are occupied by cops. Hence, after the cops’ turn in round $i$, there are at least $|V^i_B| - k \geq k$ cleared vertices that are unoccupied and are adjacent to contaminated vertices, and so, these vertices get recontaminated after the robber’s turn in round $i$. Therefore $|V^i_C| \leq |V^{i-1}_C|$.

**Lemma 6.2.2** Let $P_m \square P_n$, $n \geq m \geq 1$, be a grid with $m$ rows and $n$ columns. For a cop-win strategy for $P_m \square P_n$, let $i$ be a round in which $|V^i_B| < m$. Then either $|V^i_C| \leq \frac{m^2-m}{2}$ or $|V^i_C| \geq mn - \frac{m^2-m}{2}$.

**Proof.** After the cops’ turn in round $i$, if every row contains both cleared vertices and contaminated vertices, then $|V^i_B| \geq m$, which is a contradiction. So there is at least one row that contains only cleared vertices or only contaminated vertices. Similarly, there is at least one column that contains only cleared vertices or only contaminated vertices. Thus, we have two cases.

Case 1. There exist a row and a column that both contain only contaminated vertices. It follows from Theorem 6.1.1(ii) that $|V^i_C| \leq \frac{m^2-m}{2}$.

Case 2. There exist a row and a column that both contain only cleared vertices. After the cops’ turn in round $i$, let $(U_1, U_2)$ be a partition of $V(P_m \square P_n)$, where $U_1$ is the set of cleared vertices, $U_2$ is the set of contaminated vertices,
and $V^i_B$ is the boundary of $U_1$. It is easy to see that every row and every column must contain a vertex in $U_1$. Note that $|V^i_B| < m$. Hence, there must exist a row that contains only non-boundary vertices. Since every row contains a vertex in $U_1$, there must exist a row that contains only cleared non-boundary vertices. Similarly, we can show that there must exist a column that contains only cleared non-boundary vertices. From Theorem 6.1.1(ii), we have $|U_2 \cup V^i_B| \leq \frac{m^2 - m}{2}$ (where $U_2 \cup V^i_B$ is considered as $V_1$ and $U_1 \setminus V^i_B$ is considered as $V_2$ in Theorem 6.1.1). Thus, $|U_1 \setminus V^i_B| \geq mn - \frac{m^2 - m}{2}$. Notice that the vertices in $U_1 \setminus V^i_B$ are all cleared non-boundary vertices. Hence, $|V^i_C| \geq |U_1 \setminus V^i_B| \geq mn - \frac{m^2 - m}{2}$.}

**Theorem 6.2.3** For $n \geq m \geq 1$, $c_0(P_m \Box P_n) \geq \left\lceil \frac{m+1}{2} \right\rceil$.

**Proof.** The claim is trivial when $m = 1$. Suppose that $c_0(P_m \Box P_n) \leq \left\lfloor \frac{m-1}{2} \right\rfloor$, $m \geq 2$. Consider a cop-win strategy for $P_m \Box P_n$ that uses at most $\left\lfloor \frac{m-1}{2} \right\rfloor$ cops. We will use mathematical induction to show that $|V^i_C| \leq \frac{m^2 - m}{2}$ for all $i \geq 0$. When $i = 0$, it is easy to see that $|V^0_C| \leq \left\lfloor \frac{m-1}{2} \right\rfloor \leq \frac{m^2 - m}{2}$. Assume that $|V^{i-1}_C| \leq \frac{m^2 - m}{2}$ holds for round $i - 1$, where $i \geq 1$. There are two cases.

**Case 1:** $|V^i_B| < m$. Since $|V^{i-1}_C| \leq \frac{m^2 - m}{2}$, there are at most $\frac{m^2 - m}{2} + \left\lfloor \frac{m-1}{2} \right\rfloor \leq \frac{m^2}{2}$ cleared vertices after the cops’ turn in round $i$. Since $\frac{m^2}{2} < mn - \frac{m^2 - m}{2}$, it follows from Lemma 6.2.2 that $|V^i_C| \leq \frac{m^2 - m}{2}$.

**Case 2:** $|V^i_B| \geq m$. It follows from Lemma 6.2.1 that $|V^i_C| \leq |V^{i-1}_C|$, and thus, $|V^i_C| \leq \frac{m^2 - m}{2}$.
From the above, we have $|V^i_C| \leq \frac{m^2 - m}{2}$ for all $i \geq 0$, which is a contradiction. Hence, $c_0(P_m \square P_n) \geq \lceil \frac{m+1}{2} \rceil$. \hfill \blacksquare

**Lemma 6.2.4** For a cop-win strategy for $C_m \square P_n$ with $m \geq 2n \geq 4$, let $i$ be a round in which $|V^i_B| < 2n$. Then either $|V^i_C| \leq n^2$ or $|V^i_C| \geq mn - n^2 + 2n - 1$.

**Proof.** After the cops’ turn in round $i$, if every copy of $P_n$ contains both cleared and contaminated vertices, then $|V^i_B| \geq m \geq 2n$, which is a contradiction. So there must exist a path $P^i_n$, $1 \leq j \leq m$, which contains only cleared or only contaminated vertices. Hence, we have two cases.

**Case 1.** $P^i_n$ contains only contaminated vertices. After the cops’ turn in round $i$, let $V_1$ be the set of cleared vertices and $V_2$ be the set of contaminated vertices. It follows from Theorem 6.1.2(iv) that $|V_1| \leq n^2$. Since $|V^i_C| \leq |V_1|$, we have $|V^i_C| \leq n^2$.

**Case 2.** $P^i_n$ contains only cleared vertices. Similar to the proof of Lemma 6.2.2, we know that the maximum number of contaminated vertices in Case 2 is equal to the maximum number of cleared non-boundary vertices in Case 1. So we need to find this number in Case 1.

Consider Case 1 again, that is, $P^i_n$ contains only contaminated vertices. Let $n_c$ be the maximum number of cleared non-boundary vertices over all possible cop-win strategies for $C_m \square P_n$ satisfying that $|V^i_B| \leq 2n - 1$. Let $k \leq 2n - 1$ be the largest size of the boundary such that the number of cleared non-boundary vertices is $n_c$. We will prove that $k = 2n - 1$. Assume that $k \leq 2n - 2$. If every copy of $C_m$ contains at least two cleared vertices, then $|V^i_B| \geq 2n$, which

138
is a contradiction. Thus, there exists a copy of $C_m$ that contains at most one cleared vertex. Let $u$ be a contaminated vertex on this cycle but not on $P_n^j$. We can obtain a new partition on $V(C_m \square P_n)$ by letting $u$ become cleared. Notice that $P_n^j$ still contains only contaminated vertices after changing $u$ from contaminated to cleared. Further, the new partition must satisfy: (a) the size of the boundary is $k + 1$ and the number of cleared non-boundary vertices is $n_c$, or (b) the size of the boundary is at most $k$ and the number of cleared non-boundary vertices is at least $n_c + 1$. Note that case (a) contradicts that $k$ is the largest size of the boundary when the number of cleared non-boundary vertices is $n_c$; and case (b) contradicts that $n_c$ is the maximum number of cleared non-boundary vertices when the boundary has size at most $2n - 1$. Thus, we know that $k$ cannot be less than $2n - 1$. Hence, $k = 2n - 1$. Since there are at most $n^2$ cleared vertices when the size of the boundary is $2n - 1$, we have $n_c + (2n - 1) \leq n^2$, and thus $n_c \leq n^2 - 2n + 1$.

So in Case 2, the number of contaminated vertices is at most $n^2 - 2n + 1$, and therefore, the number of cleared vertices is at least $mn - n^2 + 2n - 1$.  

**Lemma 6.2.5** For $m \geq 2n \geq 4$, $c_0(C_m \square P_n) \geq n + 1$.

**Proof.** Assume that $c_0(C_m \square P_n) \leq n$. Consider an optimal cop-win strategy for $C_m \square P_n$ that uses $n$ cops. We will show that $|V_C^i| \leq n^2$ for all $i \geq 0$. When $i = 0$, $|V_C^0| \leq n < n^2$. Assume that $|V_C^{i-1}| \leq n^2$, $i \geq 1$. There are two cases when we consider $|V_C^i|$.

Case 1: $|V_B^i| < 2n$. Since $|V_C^{i-1}| \leq n^2$, there are at most $n^2 + n$ cleared vertices after the cops’ turn in round $i$. Since $n^2 + n < mn - n^2 + 2n - 1$, it
follows from Lemma 6.2.4 that $|V^i_C| \leq n^2$.

Case 2: $|V^i_B| \geq 2n$. Similar to the proof of Lemma 6.2.1, we have $|V^i_C| \leq |V^{i-1}_C| \leq n^2$.

Thus, $|V^i_C| \leq n^2$ for all $i \geq 0$, which contradicts the assumption that $n$ cops can clear $C_m \square P_n$. Hence, $c_0(C_m \square P_n) \geq n + 1$.

Lemma 6.2.6 For a cop-win strategy for $C_m \square P_n$ with $3 \leq m < 2n$, let $i$ be a round in which $|V^i_B| < m$. Then either $|V^i_C| \leq \frac{m^2}{4}$ or $|V^i_C| \geq mn - \frac{m^2}{4} + m - 1$.

Proof. After the cops’ turn in round $i$, there is a path $P^i_n$, $1 \leq j \leq m$, which contains only cleared or only contaminated vertices; otherwise, $|V^i_B| \geq m \geq 2n$, which is a contradiction. So there are two cases for $P^i_n$.

Case 1. $P^i_n$ contains only contaminated vertices. Let $n_c$ be the maximum number of cleared non-boundary vertices over all partitions on $V(C_m \square P_n)$ when $|V^i_B| \leq m - 1$. Let $k \leq m - 1$ be the largest size of the boundary such that the number of all cleared non-boundary vertices is $n_c$. We will prove that $k = m - 1$. Assume that $k \leq m - 2$. Consider a partition $(V_1, V_2)$ of $V(C_m \square P_n)$, where $V_1$ is the set of cleared vertices and $V_2$ is the set of contaminated vertices, such that $|\partial V_1| = k$ and $|V_1 \setminus \partial V_1| = n_c$. From the proof of Theorem 6.1.3, there must exist a copy of $P_n$ containing both cleared and contaminated vertices. Let $u$ be a contaminated vertex on this copy of $P_n$, and let $V'_1 = V_1 \cup \{u\}$ and $V'_2 = V_2 \setminus \{u\}$. Since $P^i_n$ contains only contaminated vertices, on the new partition $(V'_1, V'_2)$, either the size of the boundary is $k + 1$ and the number of cleared non-boundary vertices is $n_c$, which is a contradiction, or the size of the
boundary is at most \( k \) and the number of cleared non-boundary vertices is at least \( n_c + 1 \), which is also a contradiction. Hence, \( k = m - 1 \). It follows from Theorem 6.1.3(iii) that \( |V_i| \leq \frac{m^2}{4} \) when \( m \) is even, and \( |V_i| \leq \frac{m^2-1}{4} \) when \( m \) is odd. Since \( |V_C^i| \leq |V_i| \), we have \( |V_C^i| \leq \max\{\frac{m^2}{4}, \frac{m^2-1}{4}\} = \frac{m^2}{4} \).

Case 2. \( P_n^i \) contains only cleared vertices. In this case, the number of contaminated vertices can be considered as the number of cleared non-boundary vertices in Case 1. From Case 1, we know that \( n_c + (m - 1) \leq \frac{m^2}{4} \), that is, \( n_c \leq \frac{m^2}{4} - m + 1 \). So the maximum number of contaminated vertices in Case 2 is at most \( \frac{m^2}{4} - m + 1 \). Hence, \( |V_C^i| \geq mn - \frac{m^2}{4} + m - 1 \).

Lemma 6.2.7 For \( 3 \leq m < 2n \), \( c_0(C_m \Box P_n) \geq \lceil \frac{m+1}{2} \rceil \).

Proof. Assume that \( c_0(C_m \Box P_n) \leq \lceil \frac{m-1}{2} \rceil \). Consider an optimal cop-win strategy that uses \( \lceil \frac{m-1}{2} \rceil \) cops. We will show that \( |V_C^i| \leq \frac{m^2}{4} \) for all \( i \geq 0 \). When \( i = 0 \), \( |V_C^0| \leq \lceil \frac{m-1}{2} \rceil < \frac{m^2}{4} \). Assume that \( |V_C^{i-1}| \leq \frac{m^2}{4} \), \( i \geq 1 \). There are two cases.

Case 1. \( |V_C^i| \leq m - 1 \). Since \( |V_C^{i-1}| \leq \frac{m^2}{4} \), there are at most \( \lceil \frac{m-1}{2} \rceil + \frac{m^2}{4} \) cleared vertices after the cops’ turn in round \( i \). Since \( \lceil \frac{m-1}{2} \rceil + \frac{m^2}{4} < mn - \frac{m^2}{4} + m - 1 \), it follows from Lemma 6.2.4 that \( |V_C^i| \leq \frac{m^2}{4} \).

Case 2. \( |V_C^i| \geq m \). Note that in round \( i \), at most \( \lceil \frac{m-1}{2} \rceil \) vertices are cleared after the cops’ turn, but at least \( \lceil \frac{m+1}{2} \rceil \) cleared vertices get recontaminated after the robber’s turn. Thus, \( |V_C^i| \leq |V_C^{i-1}| \leq \frac{m^2}{4} \).

From the above cases, \( |V_C^i| \leq \frac{m^2}{4} \) for all \( i \geq 0 \), which is a contradiction.

Theorem 6.2.8 For \( m \geq 3 \) and \( n \geq 2 \), \( c_0(C_m \Box P_n) \geq \min\{\lceil \frac{m+1}{2} \rceil, n + 1\} \).

141
Lemma 6.2.9 For \( n \geq m \geq 3 \), \( c_0(C_m \Box C_n) \geq \lceil \frac{m+1}{2} \rceil \).

Let \( \text{pw}(G) \) denote the pathwidth of a graph \( G \). The following result appears in [21], which gives a lower bound for \( mc_0(G) \).

Lemma 6.2.10 ([21]) For any connected graph \( G \), \( mc_0(G) \geq \frac{1}{2}(\text{pw}(G) + 1) \).

Lemma 6.2.11 For \( n > m \geq 3 \), both \( mc_0(C_m \Box C_n) \) and \( mc_0(C_m \boxtimes C_n) \) are at least \( m + 1 \), and both \( mc_0(C_m \Box C_m) \) and \( mc_0(C_m \boxtimes C_m) \) are at least \( m \).

**Proof.** From Theorem 7.1 in [23], we have

\[
\text{pw}(C_m \Box C_n) = \begin{cases} 
2m & \text{if } n > m \geq 3, \\
2m - 1 & \text{if } n = m \geq 3.
\end{cases}
\]

From Lemma 6.2.10, we have

\[
mc_0(C_m \boxtimes C_n) \geq \frac{1}{2}(\text{pw}(C_m \boxtimes C_n) + 1) \geq \frac{1}{2}(\text{pw}(C_m \Box C_n) + 1).
\]

Hence, the claim holds for \( mc_0(C_m \boxtimes C_n) \). Similarly, the claim also holds for \( mc_0(C_m \Box C_n) \).  

6.3 Cartesian Products

**Theorem 6.3.1** For \( n \geq m \geq 1 \), \( c_0(P_m \Box P_n) = mc_0(P_m \Box P_n) = \lceil \frac{m+1}{2} \rceil \).

**Proof.** The claim is trivial when \( m = 1 \). When \( m \) is odd and \( m \geq 3 \), we have the following monotonic strategy for \( \frac{m+1}{2} \) cops to clear \( P_m \Box P_n \).
1. Place cop $\lambda_i$ on $v_{2i-1,1}$ where $1 \leq i \leq \frac{m+1}{2}$. Hence, we use $\frac{m+1}{2}$ cops in total. Let $j = 1$ and $k = \frac{m+1}{2}$.

2. For $i = 1$ to $\frac{m+1}{2}$, take one of the following actions for the cop $\lambda_i$:
   
   (1) if $i = k$, then move $\lambda_i$ to its right neighbor.
   
   (2) if $i \neq k$, then move $\lambda_i$ to its lower neighbor.

3. If $k > 1$, then for $i = 1, \ldots, \frac{m+1}{2}$, move the cop $\lambda_i$ to its upper neighbor.
   
   Set $k \leftarrow k - 1$. If $k \geq 1$, go to Step 2.

4. If $j < n - 1$, set $j \leftarrow j + 1$, $k \leftarrow \frac{m+1}{2}$, and go to Step 2. If $j = n - 1$, then all vertices of $P_m \Box P_n$ are cleared.

When $m$ is even and $m \geq 2$, we can easily modify the above strategy so that $\frac{m}{2} + 1$ cops can clear $P_m \Box P_n$. Thus, from Theorem 6.2.3, the claim holds.

Theorem 6.3.2 For $m \geq 3$ and $n \geq 2$,

\[ c_0(C_m \Box P_n) = mc_0(C_m \Box P_n) = \min\{\lceil \frac{m+1}{2} \rceil, n + 1\}. \]

Proof. Applying a strategy similar to the one described in the proof of Theorem 6.3.1, we can use $\lceil \frac{m+1}{2} \rceil$ cops to clear $C_m \Box P_n$. We now give a monotonic cop-win strategy that clears $C_m \Box P_n$ with $n + 1$ cops when $n$ is odd (the strategy is similar if $n$ is even).

1. For each $i \in \{1, \ldots, \frac{n+1}{2}\}$, place one cop on $v_{2i-1,1}$ and $v_{2i-1,2}$ respectively.
   
   Hence, we use $n + 1$ cops in total.
2. For each \( i \in \{1, \ldots, \frac{n-1}{2}\} \), the cop on \( v_{2i-1,1} \) vibrates between \( v_{2i-1,1} \) and \( v_{2i,1} \) throughout the strategy while the cop on \( v_{n,1} \) stays still.

3. Using a strategy similar to the one in the proof of Theorem 6.3.1, the cops on \( v_{2i-1,2}, 1 \leq i \leq \frac{n+1}{2} \), can clear all paths from \( P^2_n \) to \( P^m_n \).

So the claim follows from Theorem 6.2.8.

**Theorem 6.3.3**

(i) For \( n \geq m \geq 3 \), \( \lceil \frac{m+1}{2} \rceil \leq c_0(C_m \square C_n) \leq m + 1 \);

(ii) for \( n > m \geq 3 \), \( mc_0(C_m \square C_n) = m + 1 \); and

(iii) for \( m \geq 3 \), \( m \leq mc_0(C_m \square C_m) \leq m + 1 \).

### 6.4 Strong Products

**Lemma 6.4.1** Consider two products \( P_m \square P_n \) and \( P_m \boxtimes P_n \). Let \( (V_1, V_2) \) be a partition of \( V(P_m \square P_n) \) and let the same partition of \( V(P_m \boxtimes P_n) \) be denoted by \( (V'_1, V'_2) \). Then \( |\partial V_1| \leq |\partial V'_1| \).

**Proof.** Let \( v \in V_1 \) and let \( v' \in V'_1 \) be the corresponding vertex of \( v \). If \( v \) has a neighbor in \( V_2 \), then \( v' \) also has a neighbor in \( V'_2 \). Thus \( |\partial V_1| \leq |\partial V'_1| \).

**Theorem 6.4.2** For \( n \geq m \geq 2 \),

(i) \( c_0(P_m \boxtimes P_n) = mc_0(P_m \boxtimes P_n) = \frac{m}{2} + 1 \) when \( m \) is even; and

(ii) \( \frac{m+1}{2} \leq c_0(P_m \boxtimes P_n) \leq mc_0(P_m \boxtimes P_n) \leq \frac{m+1}{2} + 1 \) when \( m \) is odd.
Proof. Suppose that $m$ is even. Let $v_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq n$, be the vertex on the $i$-th row and $j$-th column of $P_m \boxtimes P_n$. It is easy to see that $mc_0(P_2 \boxtimes P_n) = 2$; when $m \geq 4$, the following monotonic strategy can clear $P_m \boxtimes P_n$ with $\frac{m}{2} + 1$ cops.

1. For each $i \in \{1, \ldots, \frac{m}{2}\}$, place a cop on $v_{2i-1,1}$. Place a cop on $v_{1,2}$.
   Hence, we use $\frac{m}{2} + 1$ cops in total. Let $j = 1$ and $k = 1$.

2. Slide all cops to their lower neighbors.

3. If $k < m - 1$, slide the cop on $v_{k+1,j}$ to $v_{k+2,j+1}$ and slide all other cops to their upper neighbors; otherwise, slide all cops to their upper neighbors.
   Set $k \leftarrow k + 2$. If $k \leq m - 1$, go to Step 2.

4. If $j = n - 1$, then stop and all vertices of $P_m \boxtimes P_n$ are cleared. If $j < n - 1$, then
   
   (a) slide the cop on $v_{m-1,j}$ to $v_{m,j+1}$;
   (b) slide the cop on $v_{1,j+1}$ to $v_{1,j+2}$; and
   (c) slide all other cops to their upper neighbors.

5. Slide all the cops to their upper neighbors, except the one on $v_{1,j+2}$. Set $j \leftarrow j + 1$, $k \leftarrow 1$, and go to Step 2.

When $m$ is odd, we can clear $P_m \boxtimes P_n$ with $\frac{m+1}{2} + 1$ cops in a similar way.
Hence, $mc_0(P_m \boxtimes P_n) \leq \left\lceil \frac{m+1}{2} \right\rceil + 1$.  

145
From Lemma 6.4.1, we can use a method similar to the one in the proof of Theorem 6.2.3 to prove that \( c_0(P_m \boxtimes P_n) \geq \lceil \frac{m+1}{2} \rceil \). This completes the proof.

**Theorem 6.4.3** For \( m \geq 3 \) and \( n \geq 2 \),

\[
\min\{\lceil \frac{m+1}{2} \rceil, n+1\} \leq c_0(C_m \boxtimes P_n) \leq mc_0(C_m \boxtimes P_n) \leq \min\{\lceil \frac{m+1}{2} \rceil, 2\lceil \frac{n+1}{2} \rceil\}+1.
\]

**Proof.** We first show that \( C_m \boxtimes P_n \) can be cleared monotonically with \( \lceil \frac{m+1}{2} \rceil + 1 \) cops.

Suppose that \( m \) is odd. Let \( v_{i,j} \) be the vertex of \( C_m \boxtimes P_n \) that is on the \( i \)-th copy of \( C_m \) and the \( j \)-th copy of \( P_n \). We place a cop on vertex \( v_{1,m} \) and this cop will vibrate between the vertices \( v_{1,m} \) and \( v_{2,m} \) until all vertices of \( C^1_m \) and \( C^2_m \) are cleared. We use \( \frac{m+1}{2} \) cops to clear vertices of \( C^4_m \) and \( C^2_m \) except \( v_{1,m} \) and \( v_{2,m} \) with the strategy described in the proof of Theorem 6.4.2. We then use \( \frac{m+1}{2} + 1 \) cops to clear vertices of \( C^2_m \) and \( C^3_m \). We continue like this until all vertices of \( C^m_m \) are cleared. If \( m \) is even, we can easily extend this strategy to clear \( C_m \boxtimes P_n \) using \( \frac{m}{2} + 2 \) cops.

We now give a monotonic strategy to clear \( C_m \boxtimes P_n \) with \( 2\lceil \frac{n+1}{2} \rceil + 1 \) cops. Suppose that \( n \) is even (the strategy will be similar if \( n \) is odd). For each \( i \in \{1, \ldots, \frac{n}{2}\} \), place a cop on \( v_{2i-1,m} \) and let these \( \frac{n}{2} \) cops vibrate between \( v_{2i-1,m} \) and \( v_{2i,m} \) until all vertices of the graph are cleared. We use \( \frac{n}{2} + 1 \) cops to clear \( P^1_n, \ldots, P^{m-1}_n \) by the strategy in Theorem 6.4.2.

From Lemma 6.4.1 and the proofs of Lemmas 6.2.5 and 6.2.7, we have
\[
c_0(C_m \boxtimes P_n) \geq \begin{cases} 
\frac{m+1}{2} & \text{if } m \geq 2n, \\
\frac{m+1}{2} & \text{if } m \leq 2n - 1.
\end{cases}
\]

Thus the claim holds. \[\square\]

**Theorem 6.4.4**  
(i) For \( n \geq m \geq 3 \), \( \left\lceil \frac{m+1}{2} \right\rceil \leq c_0(C_m \boxtimes C_n) \leq m + 2; \)

(ii) for \( n > m \geq 3 \), \( m + 1 \leq mc_0(C_m \boxtimes C_n) \leq m + 2; \) and

(iii) for \( m \geq 3 \), \( m \leq mc_0(C_m \boxtimes C_m) \leq m + 2. \)

**Proof.**  
(i) We have shown in Theorem 6.4.3 that \( C_m \boxtimes P_n \) can be cleared with \( \left\lceil \frac{m+1}{2} \right\rceil + 1 \) cops. Suppose that \( m \) is odd. To clear \( C_m \boxtimes C_n \), we extend the strategy in Theorem 6.4.3 by adding the following actions: in the beginning, for each \( i \in \{1, \ldots, \frac{m-1}{2}\} \), place a cop on \( v_{n,2i} \) that will only vibrate between \( v_{n,2i-1} \) and \( v_{n,2i} \), and place a cop on \( v_{n,m} \) and this cop will stay still throughout the strategy. Clearly, \( C_m^m \) is cleared and protected by those \( \frac{m-1}{2} + 1 \) cops. We can use \( \frac{m+1}{2} + 1 \) cops to clear \( C_m^1, \ldots, C_m^{m-1} \) sequentially. Hence, we can use \( m + 2 \) cops to clear \( C_m \boxtimes P_n \) monotonically. The strategy is similar if \( m \) is even.

Similar to the proofs in Section 6.2, we can prove that \( c_0(C_m \boxtimes C_n) \geq \left\lceil \frac{m+1}{2} \right\rceil. \)

(ii) and (iii) follow from Lemma 6.2.11 and the monotonic strategy in the proof of (i). \[\square\]
6.5 Hypercubes

Theorem 6.5.1 For a hypercube $Q_n$, $n \geq 2$,

$$\frac{1}{2} \sum_{k=0}^{n-1} \binom{k}{\lfloor \frac{k}{2} \rfloor} + \frac{1}{2} \leq mc_0(Q_n) \leq \sum_{k=0}^{n-2} \binom{k}{\lfloor \frac{k}{2} \rfloor} + 1.$$

Proof. From Lemma 6.2.10 and [17], we have

$$mc_0(Q_n) \geq \frac{1}{2} (\text{pw}(Q_n) + 1) = \frac{1}{2} \sum_{k=0}^{n-1} \binom{k}{\lfloor \frac{k}{2} \rfloor} + \frac{1}{2}.$$

We now consider the upper bound. Note that $Q_n = Q_{n-1} \Box P_2$. For convenience, let $Q_{n-1}^1$ and $Q_{n-1}^2$ be the two copies of $Q_{n-1}$ in $Q_n$. Let $(B_1, \ldots, B_m)$ be an optimal path decomposition of $Q_{n-1}^1$, where $B_i \subseteq V(Q_{n-1}^1)$, $1 \leq i \leq m$. We give a monotonic cop-win strategy for clearing $Q_n$ with $\text{pw}(Q_{n-1}) + 1$ cops.

1. In round 1, place $\text{pw}(Q_{n-1}) + 1$ cops on vertices in $B_1$, such that every vertex in $B_1$ contains at least one cop. Let $i = 1$.

2. In round 2, slide all cops to their neighbors in $Q_{n-1}^2$, and in round 3, slide all cops back to $Q_{n-1}^1$.

3. If all vertices are cleared, then stop.

4. If there exists a vertex $v$ in $B_i \setminus B_{i+1}$ such that $v$ is occupied by a cop and it is cleared in the last two rounds, then
(a) select a vertex \( v' \) in \( B_{i+1} \) that contains the smallest number of cops among all other vertices in \( B_{i+1} \setminus B_i \),
(b) find a path between \( v \) and \( v' \), and
(c) slide a cop on \( v \) along the path to \( v' \), during which this cop does not move to \( Q_{n-1}^2 \), but all other cops move to their neighbors in \( Q_{n-1}^2 \) in the even rounds and move back to \( Q_{n-1}^1 \) in the odd rounds.

If the cop arrives at \( v' \) in an even round, then this cop stays on \( v' \) until all other cops move back to \( Q_{n-1}^1 \) in the next round; after that all cops move to \( Q_{n-1}^2 \) and then move back to \( Q_{n-1}^1 \) in the next two rounds. If the cop arrives at \( v' \) in an odd round, then all cops move to \( Q_{n-1}^2 \) and then move back to \( Q_{n-1}^1 \) in the next two rounds.

5. If \( B_i \setminus B_{i+1} \) contains no cops, then set \( i \leftarrow i + 1 \) and go to Step 3; otherwise, go to Step 4.

From [17], we have \( mc_0(Q_n) \leq pw(Q_{n-1}) + 1 = \sum_{k=0}^{n-2} \binom{k}{\frac{k}{2}} + 1 \).

Since the strategy described in the proof of Theorem 6.5.1 can be easily extended to clear \( G \square P_2 \), where \( G \) is connected, we have the following result.

**Corollary 6.5.2** Let \( G \) be a connected graph and \( P_2 \) be a path with two vertices. Then \( mc_0(G \square P_2) \leq pw(G) + 1 \).

For \( c_0(Q_n) \), we conjecture that \( c_0(Q_n) = mc_0(Q_n) \). From the results in Sections 6.3 and 6.4, we may think that \( c_0(G \square H) = mc_0(G \square H) \) for any graphs \( G \) and \( H \). But this is not always true. The following result implies that their difference can be arbitrarily large for product graphs.
Lemma 6.5.3 For any positive integer $k$, there exist graphs $G$ and $H$ such that $c_0(G\square H) \leq 4$ and $mc_0(G\square H) \geq k$.

Proof. For any positive integer $\ell$, we can construct a graph $G$ such that $c_0(G) = 2$ and $pw(G) \geq \ell$ (see Theorem 4.1 in [21]). It is easy to see that we can use four cops to clear the two copies of $G$ in $G\square P_2$ synchronously. So $c_0(G\square P_2) \leq 4$. On the other hand, from Lemma 6.2.10, we have

$$mc_0(G\square P_2) \geq \frac{1}{2}(pw(G\square P_2) + 1) \geq \frac{1}{2}(\ell + 1).$$

Remarks: The hypercube result given by Tošić [57] is incorrect. We gave new results on hypercubes with rigorous proofs in this chapter.
Chapter 7

Lower Bounds on the Cop Number

In this chapter, we provide lower bounds and matching upper bounds on the cop number of a number of classes of graphs, including graph joins, Lexicographic products of graphs, complete multipartite graphs and split graphs.

7.1 Lower Bounds

Let $u$ and $v$ be two vertices of $G$. We use $d_G(u,v)$ to denote the distance between $u$ and $v$, which is the number of edges in a shortest path connecting $u$ and $v$ on $G$. Let $H$ be a subgraph of $G$. We say $H$ is an isometric subgraph of $G$ if $d_H(u,v) = d_G(u,v)$, for any two vertices $u,v \in V(H)$. Lemma 7.1.1 appears in [56].
Lemma 7.1.1 If $H$ is an isometric subgraph of $G$, then $c_0(H) \leq c_0(G)$.

Corollary 7.1.2 If $G$ contains a clique with $m$ vertices, then $c_0(G) \geq \lceil \frac{m}{2} \rceil$.

Lemma 7.1.3 In the zero-visibility cops and robber game on $G$, for any cop strategy, the number of cops required to visit all vertices of $G$ within a single round of the game is at least $\mu(G)$.

Proof. Consider a round of the game on $G$ in which all vertices in $V(G)$ are visited by cops. Without loss of generality, assume that each vertex can only contain at most one cop at any moment in the round. Let $m_1$ be the number of cops that visit two vertices of $G$ and let $m_2$ be the number of cops that visit exactly one vertex of $G$. It is easy to see that $|V(G)| = 2m_1 + m_2$. Since $\mathcal{M}(G)$ is a maximum matching in $G$, hence, $m_1 \leq |\mathcal{M}(G)|$. Note that $\mu(G) = |V(G)| - |\mathcal{M}(G)|$. Hence, we have $\mu(G) \leq |V(G)| - m_1 = (2m_1 + m_2) - m_1 = m_1 + m_2$. \[\qed\]

In the following, we establish specific conditions on strategies, and show that these conditions must be met in every round of a strategy when insufficient cops are used for capturing the robber. This method is later used for finding lower bounds on the cop number of several types of graphs.

Definition 7.1.4 For a strategy $S$ of a graph $G$, let $\mathcal{P}_{G,S}(i)$ be a propositional function such that at the end of round $i$, every cleared vertex is occupied by at least one cop and the number of cleared vertices is less than $|V(G)|$. When $G$ and $S$ are clear from the context, we drop the subscript and simply use $\mathcal{P}(i)$.

The following proposition is straightforward.
Proposition 7.1.5 For a cop-win strategy for $G$, there must exist a round satisfying that:

1. all vertices of $G$ are occupied by cops after the cop’s turn, or

2. at least one unoccupied vertex of $G$ is cleared after the robber’s turn.

7.1.1 Graph Joins

Let $G$ and $H$ be two graphs. The graph join, denoted as $G + H$, is the graph whose vertex set is $V(G) \cup V(H)$, and two vertices $u$ and $v$ are adjacent in $G + H$ if and only if $uv \in E(G)$, or $uv \in E(H)$, or $u \in V(G)$ and $v \in V(H)$.

Theorem 7.1.6 $c_0(G + H) \geq \min\{\mu(G), \mu(H)\}$.

Proof. Suppose that $\mu(G) \leq \mu(H)$. We will use Definition 7.1.4 to show that $\mu(G) - 1$ cops are insufficient for clearing $G + H$. Assume that $c_0(G + H) < \mu(G)$. Consider a strategy for $G + H$ that uses at most $\mu(G) - 1$ cops. We will show that at the end of each round, only occupied vertices are cleared.

At round 0, since cops are placed on vertices in $V(G + H)$, we know cleared vertices are those occupied ones. Hence, $\mathcal{P}(0)$ holds. Suppose that $\mathcal{P}(i)$ holds for some $i \geq 0$. From Lemma 7.1.3, we know that $\mu(G) - 1$ cops are insufficient for visiting all the vertices in $V(G)$, not to speak of all vertices in $V(H)$. It is very easy to see that both $G$ and $H$ contain contaminated vertices when the cop’s turn is finished at round $i + 1$. Hence, all unoccupied cleared vertices will get recontaminated during the robber’s turn at round $i + 1$. We know
\( \mathcal{P}(i+1) \) holds. Thus, from Proposition 7.1.5, we have \( c_0(G + H) \geq \mu(G) \). $\blacksquare$

The lower bound in Theorem 7.1.6 is tight. For example, if \( G \) and \( H \) are both independent vertex sets, then \( G + H \) is a complete bipartite graph. It is easy to see that \( c_0(G + H) = \min\{|V(G)|, |V(H)|\} \). Since \( \mu(G) = |V(G)| \) and \( \mu(H) = |V(H)| \), we have \( c_0(G + H) = \min\{\mu(G), \mu(H)\} \).

Let \( K_n \) denote a complete graph of \( n \) vertices, and let \( K_n \) denote an independent vertex set of \( n \) vertices.

**Corollary 7.1.7** If \( |V(G)| \leq n \), then \( c_0(G + K_n) \geq \mu(G) \).

In the following, we give some further discussions about Theorem 7.1.6. There exist a great number of graphs whose cop number is equal to the lower bound given in Theorem 7.1.6. Consider an example of graph join in Figure 7.1. From Theorem 7.1.6, we have \( c_0(G + H) \geq \min\{\mu(G), \mu(H)\} = 3 \). We now describe a strategy that clears \( G + H \) using exactly 3 cops.

1. Place cops \( \lambda_1 \), \( \lambda_2 \) and \( \lambda_3 \) on \( u_1 \), \( u_4 \) and \( u_5 \) respectively.

2. From round 1 to round 6, let \( \lambda_1 \) vibrate on \( u_1 \) and \( u_2 \), let \( \lambda_2 \) vibrate on \( u_3 \) and \( u_4 \); in the meantime, let \( \lambda_3 \) move between \( u_5 \) and \( V(H) \) to clear \( v_1 \), \( v_2 \) and \( v_3 \) and stay on \( u_5 \) at the end of round 6.

3. Slide \( \lambda_1 \) from \( u_1 \) to \( u_2 \), slide \( \lambda_2 \) from \( u_4 \) to \( v_4 \) and slide \( \lambda_3 \) from \( u_5 \) to \( v_5 \).

4. Slide \( \lambda_1 \) from \( u_2 \) to \( u_3 \), slide \( \lambda_2 \) from \( v_4 \) to \( u_4 \) and slide \( \lambda_3 \) from \( v_5 \) to \( u_5 \).

5. In the following rounds, let \( \lambda_1 \) vibrate on \( u_3 \) and \( u_2 \) and let \( \lambda_2 \) vibrate on \( u_4 \) and \( u_1 \); in the meantime, let \( \lambda_3 \) clear \( v_1 \), \( v_2 \) and \( v_3 \).
Theorem 7.1.8 Let $G$ and $H$ be two graphs such that $2 \leq |V(G)| \leq |V(H)|$. If $G$ has a perfect matching, then $c_0(G + H) \geq \mu(G) + 1$.

Proof. Since $G$ has a perfect matching, we have $\mu(G) = \frac{|V(G)|}{2}$. We will use Definition 7.1.4 to show that $\mu(G)$ cops are insufficient. Suppose that $\mu(G)$ cops can clear $G + H$. At the end of round 0, it is easy to see that the cleared vertices are those occupied ones. Hence, $P(0)$ holds. Suppose that $P(i)$ holds for some $i \geq 0$. Obviously, there are at most $2\mu(G) = |V(G)|$ cleared vertices after the cops’ turn in round $i + 1$. Consider the beginning moment of round $i + 1$.

Case 1. All cleared vertices are contained either in $V(G)$ or in $V(H)$. Without loss of generality, assume that all cops stay on $G$ at the beginning moment of round $i + 1$.

Case 1.1. If all cops are still on $G$ at the end of round $i + 1$, then all vertices in $V(H)$ must be contaminated throughout round $i + 1$. Hence, all unoccupied vertices in $V(G)$ must be contaminated at the end of round $i + 1$. 

Figure 7.1: An example of graph join for which 3 cops are sufficient.
Thus, $P(i + 1)$ holds.

Case 1.2. If there is any cop sliding from $G$ to $H$ in round $i + 1$, then both $G$ and $H$ must contain a vertex that remains contaminated throughout round $i + 1$. Hence, all unoccupied vertices in $V(G)$ and $V(H)$ must be contaminated at the end of round $i + 1$. Thus, $P(i + 1)$ holds.

Case 2. Both $G$ and $H$ contain cleared vertices. It is easy to see that both $G$ and $H$ contain a vertex that remains contaminated throughout round $i + 1$. Hence, all cleared vertices at the end of round $i + 1$ are those occupied ones. We know $P(i + 1)$ also holds.

Therefore, all cleared vertices in $V(G + H)$ at the end of each round are those occupied ones. From Proposition 7.1.5, we know that $\mu(G)$ cops are insufficient for clearing $G + H$. This completes the proof.

We give some further discussions about Theorem 7.1.8. There exist a great number of graphs whose cop number is equal to the lower bound given in Theorem 7.1.8. Consider the example of graph join in Figure 7.2. From Theorem 7.1.8, we have $c_0(G + H) \geq \mu(G) + 1 = 4$. We now describe a strategy that clears $G + H$ using exactly 4 cops.

1. Place two cops $\lambda_1$ and $\lambda_2$ on $u_1$, place cop $\lambda_3$ on $u_4$ and place cop $\lambda_4$ on $u_5$.

2. From round 1 to round 6, let $\lambda_1$ vibrate on $u_1$ and $u_2$, let $\lambda_3$ vibrate on $u_3$ and $u_4$ and let $\lambda_4$ vibrate on $u_5$ and $u_6$; in the meantime, let $\lambda_2$ clear $v_1$, $v_2$ and $v_3$ and stay on $u_1$ at the end of round 6.
3. In round 7, slide $\lambda_1$ from $u_1$ to $u_2$, slide $\lambda_2$ from $u_1$ to $v_4$, slide $\lambda_3$ from $u_4$ to $u_3$, and slide $\lambda_4$ from $u_5$ to $u_6$.

4. In round 8, slide $\lambda_1$ from $u_2$ to $u_1$, slide $\lambda_2$ from $v_4$ to $v_5$, slide $\lambda_3$ from $u_3$ to $u_4$, and slide $\lambda_4$ from $u_6$ to $u_5$.

5. In round 9, slide $\lambda_1$ from $u_1$ to $u_2$, slide $\lambda_2$ from $v_5$ to $v_4$, slide $\lambda_3$ from $u_4$ to $v_6$, and slide $\lambda_4$ from $u_5$ to $u_6$.

6. In round 10, slide $\lambda_1$ from $u_2$ to $u_3$, slide $\lambda_2$ from $v_5$ to $v_4$, slide $\lambda_3$ from $v_6$ to $u_4$, and slide $\lambda_4$ from $u_6$ to $u_5$.

7. In the following rounds until all vertices are cleared, let $\lambda_1$ vibrate on $u_3$ and $u_2$, let $\lambda_3$ vibrate on $u_4$ and $u_1$, and let $\lambda_4$ vibrate on $u_5$ and $u_6$; in the meantime, slide $\lambda_2$ to $u_1$ and use it to clear $v_1$, $v_2$ and $v_3$.

![Figure 7.2: An example of graph join for which 4 cops are sufficient.](image)

*Cone graph* is the graph join of a cycle $C_m$ and an independent vertex set $\overline{K_n}$, where $m \geq 3$ and $n \geq 1$. Let $V(C_m) = \{u_1, \ldots, u_m\}$ and $V(\overline{K_n}) = \{v_1, \ldots, v_n\}$.
Theorem 7.1.9 If $n \leq 2$, then $c_0(C_m + \overline{K}_n) \geq \min\{\left\lceil \frac{m+1}{2} \right\rceil, n+1\}$. If $n \geq 3$, then $c_0(C_m + \overline{K}_n) \geq \min\{\left\lceil \frac{m+1}{2} \right\rceil, n\}$.

Proof. If $n = 1$ or $n = 2$ and $m = 3$, then $\min\{\left\lceil \frac{m+1}{2} \right\rceil, n+1\} = 2$. Since $C_m + \overline{K}_n$ contains a cycle of length 3, it follows from Lemma 7.1.1 that $c_0(C_m + \overline{K}_n) \geq 2$.

If $n = 2$ and $m \geq 4$, then $\min\{\left\lceil \frac{m+1}{2} \right\rceil, n+1\} = 3$. We will use Definition 7.1.4 to show that 2 cops are insufficient for clearing $C_m + \overline{K}_n$. Assume that $c_0(C_m + \overline{K}_2) \leq 2$. Consider a strategy for $C_m + \overline{K}_2$ that uses at most 2 cops. Note that $c_0(C_m + \overline{K}_2) \leq 2 < \frac{m+2}{2}$. Obviously, $\mathcal{P}(0)$ holds. Suppose that $\mathcal{P}(i)$ holds for some $i \geq 0$. Note that there are at most 2 cleared vertices at the beginning of round $i+1$. After the cops’ turn at round $i+1$, we know there are at most 4 cleared vertices. It is easy to see that all those cleared vertices must have a contaminated neighbor. Thus, after the robber’s turn at round $i+1$, all unoccupied cleared vertices become recontaminated. Hence, $\mathcal{P}(i+1)$ also holds. From Proposition 7.1.5, we know that 2 cops are insufficient for clearing $C_m + \overline{K}_2$. Therefore, $c_0(C_m + \overline{K}_2) \geq 3$.

If $n \geq 3$, it follows from Theorem 7.1.6 that $c_0(C_m + \overline{K}_n) \geq \min\{\left\lceil \frac{m+1}{2} \right\rceil, n\}$. 

Theorem 7.1.10 For the graph join of $C_m$ and $C_n$, where $m \leq n$, if $m$ is odd, then $c_0(C_m + C_n) \geq \frac{m+1}{2} + 1$; if $m$ is even, then $c_0(C_m + C_n) \geq \frac{m}{2} + 2$.

Proof. We first consider the case when $m$ is odd. We will use Definition 7.1.4 to show that $\frac{m+1}{2}$ cops are insufficient for clearing $C_m + C_n$. Assume that
Consider a strategy for $C_m + C_n$ that uses at most $\frac{m+1}{2}$ cops. Note that $c_0(C_m + C_n) \leq \frac{m+1}{2} < \frac{m+n}{2}$. So $\mathcal{P}(0)$ holds. Suppose that $\mathcal{P}(i)$ holds for some $i \geq 0$. Consider the moment when the cops’ turn is finished at round $i + 1$. It is easy to see that all cleared vertices have a contaminated neighbor. After the robber’s turn at round $i + 1$, all unoccupied cleared vertices will become recontaminated. Thus, $\mathcal{P}(i+1)$ also holds. Hence, $c_0(C_m + C_n) \geq \frac{m+1}{2} + 1$ when $m$ is odd.

Similarly, we can prove that $c_0(C_m + C_n) \geq \frac{m}{2} + 2$ when $m$ is even. This completes the proof.

7.1.2 Lexicographic Products of Graphs

Recall that $G \cdot H$ denotes the lexicographic product of $G$ and $H$. We now consider the cop number of lexicographic products of graphs.

**Lemma 7.1.11** Let $P_m$ be a path with $m$ vertices. For $m \geq 4$, $c_0(P_m \cdot G) \geq |V(G)|$.

**Proof.** For convenience, we use $G^1, G^2, \ldots, G^m$ to denote the $m$ copies of $G$ in $P_m \cdot G$. Assume that $c_0(P_m \cdot G) < |V(G)|$. Consider a strategy for $P_m \cdot G$ that uses at most $|V(G)| - 1$ cops. Let $i$ be the index of the round satisfying the following two conditions: (i) when the cops’ turn is finished at round $i$, there are at most two copies of $G$ that are free of the robber; (ii) when the cops’ turn is finished at round $i + 1$, there are at least three copies of $G$ that are free of the robber.
Consider the moment when the cops’ turn is done at round $i$. If there is only one copy of $G$ that is free of the robber, say $G^k$, then we know each vertex in $P_m \cdot G$ must be adjacent to at least one contaminated vertex. Hence, at the end of round $i$, all unoccupied vertices are contaminated. Note that $c_0(G) \leq |V(G)| - 1$. We know there are at most $2|V(G)| - 2$ cleared vertices after the cop’s turn at round $i + 1$. Since $2|V(G)| - 2 < 3|V(G)|$, there do not exist three copies of $G$ that are free of the robber after the cop’s turn at round $i + 1$. This contradicts condition (ii). If there are exactly two copies of $G$ that are free of the robber, then there are two cases:

Case 1. The two copies of $G$ are not consecutive. Obviously, each vertex in $V(P_m \cdot G)$ must be adjacent to at least one contaminated vertex. Hence, only occupied vertices are cleared at the end of round $i$. Thus, the total number of cleared vertices when the cops’ turn is finished at round $i + 1$ is at most $2|V(G)| - 2$. This contradicts condition (ii).

Case 2. The two copies of $G$ are consecutive. Let $G^k$ and $G^{k+1}$, where $k \geq 0$, denote the two consecutive copies of $G$. There are two subcases.

Case 2.1. $k = 1$ or $k = m - 1$. We consider the case when $k = 1$ in the next, and the case when $k = m - 1$ can be proved in a similar way. Note that $G^j$ contains at least one vertex that is contaminated after the cops’ turn at round $i$, for every $j$ in $\{3, \ldots, m\}$. All unoccupied vertices in $V(P_m \cdot G) \setminus V(G^1)$ are contaminated at the end of round $i$. Since there are at most $2|V(G)| - 1$ cleared vertices at the end of round $i$, we know the total number of cleared vertices in $P_m \cdot G$ is at most $3|V(G)| - 2 < 3|V(G)|$ when the cop’s turn is
finished at round $i + 1$. This contradicts condition (ii).

Case 2.2. $2 \leq k \leq m - 2$. It is easy to see that each vertex in $V(P_m \cdot G)$ is adjacent to at least one contaminated vertex. Hence, all unoccupied vertices are contaminated at the end of round $i$. Further, the total number of cleared vertices when the cop’s turn is finished at round $i + 1$ is at most $2|V(G)| - 2$ and this contradicts condition (ii).

Hence, round $i$ does not exist and this contradicts that $|V(G)| - 1$ cops are sufficient for clearing $P_m \cdot G$. This completes the proof.

Consider $P_2 \cdot G$. If $G$ is a cycle of $n$ vertices where $n$ is even, then we can clear $P_2 \cdot G$ with at most $\frac{n}{2} + 2$ cops. Let $G^1$ and $G^2$ denote the two copies of $G$ in $P_2 \cdot G$. The following briefly describes a strategy that clears $P_2 \cdot G$ utilizing $\frac{n}{2} + 2$ cops: (1) let $\frac{n}{2}$ cops vibrate on all vertices of $G^1$ such that every vertex of $G^1$ is visited by a cop in each round; (2) let two additional cops clear all vertices of $G^2$. Consider $P_3 \cdot G$. If $G$ is a cycle of $n$ vertices where $n$ is even, we can also clear $P_3 \cdot G$ with at most $\frac{n}{2} + 2$ cops. Let $G^1$, $G^2$ and $G^3$ denote the three copies of $G$ in $P_3 \cdot G$. Similar to the above, let $\frac{n}{2}$ cops vibrate on all vertices of $G^2$, and let two additional cops clear all vertices of $G^1$ and $G^2$ respectively. Hence, Lemma 7.1.11 does not always hold when $m = 2$ or 3.

**Corollary 7.1.12** Let $W_m$ be a graph obtained from $P_m \cdot G$ ($m \geq 4$) by replacing each copy of $G$ by an arbitrary graph with $|V(G)|$ vertices. Then $c_0(W_m) \geq |V(G)|$.

**Theorem 7.1.13** Let $P_m$ be a path with $m$ vertices. For $m \geq 4$, $c_0(P_2 \cdot P_m) \geq \left\lceil \frac{m}{2} \right\rceil + 1$. 

161
Proof. We will use Definition 7.1.4 to show that $\left\lceil \frac{m}{2} \right\rceil$ cops cannot clear $P_2 \cdot P_m$. Without loss of generality, suppose $m$ is odd. The other case when $m$ is even can be proved in a similar way. Assume that $c_0(P_2 \cdot P_m) \leq \left\lceil \frac{m}{2} \right\rceil$. Consider a strategy that clears $P_2 \cdot P_m$ using at most $\left\lceil \frac{m}{2} \right\rceil$ cops. Note that $c_0(P_2 \cdot P_m) \leq \frac{m+1}{2} < 2m$. Obviously, $\mathcal{P}(0)$ holds. Suppose that $\mathcal{P}(i)$ holds for some $i \geq 0$. Consider the moment when the cops’ turn is finished at round $i+1$. Since $c_0(P_2 \cdot P_m) \leq \frac{m+1}{2}$, there are at most $m+1$ cleared vertices in $V(P_2 \cdot P_m)$. Further, if there exists a copy of $P_m$ that is free of the robber, then the other copy of $P_m$ contains at most one cleared vertex. Hence, every cleared vertex must be adjacent to a contaminated vertex. Thus, all unoccupied cleared vertices become recontaminated immediately after the robber’s turn at round $i+1$. Hence, $\mathcal{P}(i+1)$ holds. From Proposition 7.1.5, we know that $\left\lceil \frac{m}{2} \right\rceil$ cops are insufficient for clearing $P_2 \cdot P_m$. This completes the proof. 

7.1.3 Complete Multipartite Graphs

Let $K_{n_1,\ldots,n_k} = (V_1,\ldots,V_k,E)$ denote a complete $k$-partite graph, where $n_1 \leq \cdots \leq n_k$. Clearly, complete multipartite graphs can be defined recursively using graph join operations. For example, $K_{n_1,\ldots,n_k}$ can also be defined as $K_{n_1,\ldots,n_{k-1}} + \overline{K_{n_k}}$. Since the problem of determining the cop number of complete bipartite graphs has been solved, we only consider complete $k$-partite graphs with $k \geq 3$. Let $H_j = K_{n_1,\ldots,n_k} - V_j$, where $1 \leq j \leq k$. Let $\mu_{\text{min}} = \min\{\mu(H_j)|1 \leq j \leq k\}$. The next corollary is from Lemma 7.1.3.

Corollary 7.1.14 For $1 \leq i \leq k$, the minimum number of cops that can visit
all vertices of \( H_i \) in a round is \( \mu(H_i) \).

From Corollary 7.1.14, we know that all vertices of \( H_x - V(\mathcal{M}(H_x)) \) must be contained in some vertex set \( V_i \) of \( K'_{n_1,\ldots,n_k} \), where \( 1 \leq i \neq x \leq k \).

**Lemma 7.1.15** \( \frac{1}{2} \sum_{i=1}^{k-1} n_i \leq \mu_{\text{min}} \leq \frac{1}{2} \sum_{i=1}^{k} n_i \), and both upper bound and lower bound are tight.

**Proof.** We first consider the lower bound. Note that \( n_i \leq n_{i+1} \) for all \( 1 \leq i \leq k - 1 \). We have \( \sum_{i=1}^{k-1} n_i \leq \sum_{i=1}^{k} n_i - n_x = |V(H_x)| \), where \( 1 \leq x \leq k \). Let \( j \) be the index such that \( \mu_{\text{min}} = \mu(H_j) \). Hence, it follows from Corollary 7.1.14 that \( 2\mu_{\text{min}} \geq |V(H_j)| \geq \sum_{i=1}^{k-1} |V_i| \). Further, the equality holds if there is a perfect matching in \( H_k \).

We now consider the upper bound. Since \( n_k \geq n_i \) for all \( 1 \leq i \leq k \), we know \( n_k \geq |V(H_k)\setminus V(\mathcal{M}(H_k))| \). Hence, \( 2\mu(H_k) = |V(H_k)| + |V(H_k)\setminus V(\mathcal{M}(H_k))| \leq |V(H_k)| + n_k = \sum_{i=1}^{k} n_i \). Therefore, we have \( \mu_{\text{min}} \leq \mu(H_k) \leq \frac{1}{2} \sum_{i=1}^{k} n_i \). Further, if \( |V_k| = 1 \) and \( k \) is even, then \( \mu_{\text{min}} = \frac{1}{2} \sum_{i=1}^{k} n_i \).

**Lemma 7.1.16** \( c_0(K_{n_1,\ldots,n_k}) \geq \mu_{\text{min}} \).

**Proof.** We will use Definition 7.1.4 to show that \( \mu_{\text{min}} - 1 \) cops cannot clear \( K_{n_1,\ldots,n_k} \). Assume that \( c_0(K_{n_1,\ldots,n_k}) \leq \mu_{\text{min}} - 1 \). Consider a strategy for \( K_{n_1,\ldots,n_k} \) that uses at most \( \mu_{\text{min}} - 1 \) cops. From Lemma 7.1.15, we have \( c_0(K_{n_1,\ldots,n_k}) < \frac{1}{2} \sum_{i=1}^{k} n_i \). It is easy to see that \( \mathcal{P}(0) \) holds. Suppose that \( \mathcal{P}(i) \) holds for some \( i \geq 0 \). Consider round \( i + 1 \). Since \( c_0(K_{n_1,\ldots,n_k}) < \mu_{\text{min}} \), it follows from Lemma 7.1.3 that \( \mu_{\text{min}} - 1 \) cops are insufficient for visiting all vertices of any
\(k - 1\) vertex sets of \(K_{n_1,\ldots,n_k}\) within one round. Hence, there must exist two vertex sets \(V_p\) and \(V_q\), where \(1 \leq p \neq q \leq k\), such that both of which contain vertices that remain contaminated throughout round \(i+1\). In the robber’s turn at round \(i + 1\), all unoccupied cleared vertices thus become recontaminated. Hence, \(P(i + 1)\) holds. From Proposition 7.1.5, we know that \(\mu_{\text{min}} - 1\) cops are insufficient for clearing \(K_{n_1,\ldots,n_k}\). Hence, \(c_0(K_{n_1,\ldots,n_k}) \geq \mu_{\text{min}}\). □

Lemma 7.1.17 If there is a perfect matching in \(H_k\), then \(c_0(K_{n_1,\ldots,n_k}) \geq \frac{1}{2} \sum_{i=1}^{k-1} n_i + 1\).

Proof. Since there is a perfect matching in \(H_k\), then \(|V(H_k)|\) is even and \(\mu(H_k) = \frac{1}{2} \sum_{i=1}^{k-1} n_i\). From Lemma 7.1.15, we know \(\mu(H_k) = \frac{1}{2} \sum_{i=1}^{k-1} n_i = \mu_{\text{min}}\).

From Lemma 7.1.16, we have \(c_0(K_{n_1,\ldots,n_k}) \geq \mu_{\text{min}} = \frac{1}{2} \sum_{i=1}^{k-1} n_i\).

We will use Definition 7.1.4 to show that \(\frac{1}{2} \sum_{i=1}^{k-1} n_i\) cops cannot clear \(K_{n_1,\ldots,n_k}\). Assume that \(c_0(K_{n_1,\ldots,n_k}) \leq \frac{1}{2} \sum_{i=1}^{k-1} n_i\). Consider a strategy for \(K_{n_1,\ldots,n_k}\) that uses at most \(\frac{1}{2} \sum_{i=1}^{k-1} n_i\) cops. Note that \(c_0(K_{n_1,\ldots,n_k}) \leq \frac{1}{2} \sum_{i=1}^{k-1} n_i < \frac{1}{2} \sum_{i=1}^{k} n_i\). Obviously, \(P(0)\) holds. Suppose that \(P(i)\) holds for some \(i \geq 0\). Consider round \(i + 1\). Note that there are at least \(n_k\) contaminated vertices that remain unoccupied throughout round \(i + 1\). Hence, there exist at least \(n_k\) vertices that remain contaminated throughout round \(i + 1\). These contaminated vertices must be contained in at least one vertex set of \(K_{n_1,\ldots,n_k}\). Note that \(n_k \geq n_i\) for \(1 \leq i \leq k\). If these contaminated vertices are contained in one vertex set, then the vertex set must contain \(n_k\) vertices.

Case 1. There exists a vertex set \(V_p\), where \(n_p = n_k\), which contains all the vertices that remain contaminated throughout round \(i + 1\). Then all cops are
on $V(K_{n_1,\ldots,n_k}) \setminus V_p$ throughout round $i + 1$. It is easy to see that after the robber’s turn at round $i + 1$, all unoccupied cleared vertices get recontaminated. Hence, $P(i + 1)$ holds.

Case 2. There exist two or more vertex sets containing vertices that remain contaminated throughout round $i + 1$. Then we can also show that all unoccupied cleared vertices get recontaminated in the robber’s turn at round $i + 1$. Hence, $P(i + 1)$ holds.

Note that $P(i + 1)$ holds for both cases. From Proposition 7.1.5, we know that $\frac{1}{2} \sum_{i=1}^{k-1} n_i$ cops are insufficient. Thus, $c_0(K_{n_1,\ldots,n_k}) \geq \frac{1}{2} \sum_{i=1}^{k-1} n_i + 1$.

**Lemma 7.1.18** $|M(H_i)| \leq |M(H_k)| + n_k - n_i$, where $1 \leq i \leq k - 1$.

**Lemma 7.1.19** If there is no perfect matching in $H_k$, then $c_0(K_{n_1,\ldots,n_k}) \geq \mu(H_k)$.

**Proof.** We first prove that $\mu(H_k) = \mu_{\text{min}}$. Consider a maximum matching in $H_k$. Since there is no perfect matching in $H_k$, we have $\mu(H_k) > |M(H_k)|$. Note that $\mu(H_k) = |M(H_k)| + \sum_{i=1}^{k-1} n_i - 2|M(H_k)|$ and $\mu(H_i) = |M(H_i)| + \sum_{j=1}^{k-n_i-n_i-2|M(H_i)|}$. Then, $\mu(H_k) - \mu(H_i) = |M(H_i)| - n_k + n_i - |M(H_k)|$.

It follows from Lemma 7.1.18 that $\mu(H_k) - \mu(H_i) \leq 0$ for all $1 \leq i \leq k$. Hence, $\mu(H_k) = \min\{\mu(H_i)|1 \leq i \leq k\}$. It follows from Lemma 7.1.16 that $c_0(K_{n_1,\ldots,n_k}) \geq \mu_{\text{min}} = \mu(H_k)$.

By combining Lemmas 7.1.1 and 7.1.19, we obtain the next Theorem.

**Theorem 7.1.20** Let $G$ be a graph such that $K_{n_1,\ldots,n_k}$ is an induced subgraph of $G$, where $k \geq 3$. We have $c_0(G) \geq \mu(H_k)$. 

165
7.1.4 Split Graphs

A graph is a split graph if its vertex set can be partitioned into a clique and an independent set. Let $S_{m,n} = (C, I)$ denote a split graph, where $C$ is a clique of $m$ vertices and $I$ is an independent vertex set of $n$ vertices.

**Theorem 7.1.21** Let $S_{m,n} = (C, I)$ be a split graph with $|V(C)| = m$ and $|V(I)| = n$. If $m$ is odd or $V(C)$ has a pair of vertices that share no common neighbor in $V(I)$, then $c_0(S_{m,n}) \geq \lceil \frac{m^2}{2} \rceil$. If $m$ is even and every pair of vertices in $V(C)$ have a common neighbor in $V(I)$, then $c_0(S_{m,n}) \geq \frac{m^2}{2} + 1$.

**Proof.** (i) Since $S_{m,n}$ has a clique of size $m$, it follows from Corollary 7.1.2 that $c_0(S_{m,n}) \geq \lceil \frac{m}{2} \rceil$.

(ii) Assume that $c_0(S_{m,n}) \leq \frac{m}{2}$. Consider a strategy for $S_{m,n}$ that uses at most $\frac{m}{2}$ cops. Let $\mathcal{P}'(i)$ be the proposition that there exists a contaminated vertex in $V(C)$ whose unoccupied neighbors are all contaminated at the end of round $i$. We will use induction to show that the proposition holds for one of any two consecutive rounds.

Base case: $\mathcal{P}'(0)$ obviously holds.

Induction step: Suppose that $\mathcal{P}'(i)$ holds for some $i \geq 0$. Consider the beginning moment of round $i + 1$. There are two cases.

Case 1. Some cops stay on vertices in $V_2$. Note that $c_0(S_{m,n}) \leq \frac{m}{2}$. Clearly, there must be a vertex $u \in V(C)$ that remains contaminated throughout round $i + 1$. After the robber’s turn at round $i + 1$, it is easy to see that all unoccupied
neighbors of $u$ become contaminated. Therefore, $P'(i + 1)$ holds.

Case 2. All cops stay on vertices in $V_1$. Let $u \in V(C)$ be a contaminated vertex whose unoccupied neighbors are all contaminated. If some cop slides to $V(I)$ at round $i + 1$, similar to Case 1, we can show that $P'(i + 1)$ holds. Assume that all cops are on vertices in $V(C)$ throughout round $i + 1$. Then all neighbors of $u$ in $V(I)$ remain contaminated throughout round $i + 1$. Note that every pair of vertices in $V(C)$ have a common neighbor in $V(I)$. Hence, all unoccupied cleared vertices in $V(C)$ get recontaminated at the end of round $i + 1$. If $u$ is unoccupied after the cops’ turn at round $i + 1$, then $P'(i + 1)$ holds; if $u$ is occupied by a cop after the cops’ turn at round $i + 1$, then we consider the moment when the cops’ turn is finished at round $i + 2$.

Case 2.1. All the cops are still on vertices in $V(C)$. If $u$ is unoccupied after the cops’ turn at round $i + 2$, then $u$ gets recontaminated in the following robber’s turn and $P'(i + 2)$ holds. If $u$ is occupied by a cop, then there must exist a vertex in $V(C)$ which remains contaminated throughout round $i + 2$. Similar to Case 1, we can show that $P'(i + 2)$ holds.

Case 2.2. Some cops slide to $V(I)$ at round $i + 2$. Similar to Case 1, we can show that $P'(i + 2)$ holds.

Above all, we have shown that the proposition holds for one of any two consecutive rounds. This contradicts the assumption that $\frac{m}{2}$ cops are sufficient for clearing $S_{m,n}$. Therefore, $c_0(S_{m,n}) \geq \frac{m}{2} + 1$. ■
7.2 Matching Upper Bounds

Using the lower bounds presented in Section 7.1, we now investigate the cop number of graph joins, lexicographic products of graphs, complete multipartite graphs and split graphs in the following subsections.

7.2.1 Graph Joins

Let $G$ and $H$ be two graphs. Recall that $G + H$ is the graph join of $G$ and $H$. Let $G_1, \ldots, G_m$ be the maximal connected components of $G$ and let $H_1, \ldots, H_n$ be the maximal connected components of $H$. Let $\gamma_1 = \max\{\max\{pw(H_i) | 1 \leq i \leq n\} + 1 - (\mu(G) - |\mathcal{M}(G)|), 0\}$ and let $\gamma_2 = \max\{\max\{pw(G_i) | 1 \leq i \leq m\} + 1 - (\mu(H) - |\mathcal{M}(H)|), 0\}$.

**Lemma 7.2.1** $c_0(G + H) \leq \min\{\mu(G) + \gamma_1, \mu(H) + \gamma_2\}$.

**Proof.** In the following, we show that $G + H$ can be cleared with $\mu(G) + \gamma_1$ cops. Let $H'$ be a connected component of $H$. Let $(B_1, \ldots, B_k)$ be an optimal path decomposition of $H'$, where $B_i$ is a subset of $V(H')$ and $1 \leq i \leq k$. For each edge in $\mathcal{M}(G)$, place a cop on one of its endpoints and let it vibrate on the two endpoints. For each vertex in $V(G) \setminus V(\mathcal{M}(G))$, place a cop on it and let it vibrate between $V(G) \setminus V(\mathcal{M}(G))$ and $V(H')$ in the following rounds. If $|B_1| < \gamma_1$, then place $\gamma_1$ cops on $B_1$ such that each vertex in $B_1$ contains a searcher; if $|B_1| \geq \gamma_1$, then select $\gamma_1$ vertices from $B_1$ and place a searcher on each of them. Using the cops that are initially placed on $B_1$ and
on $V(G) \setminus V(\mathcal{M}(G))$, we can easily clear all vertices in $B_1$. If $k \geq 2$, then we can keep using these cops to clear all vertices in $B_2, \ldots, B_k$ respectively. After all vertices of $H'$ are cleared, we can clear all other components of $H$ in a similar way.

Similarly, we can show that $\mu(H) + \gamma_2$ cops are also sufficient for clearing $G + H$. This completes the proof.

In combination of Theorem 7.1.6 and Lemma 7.2.1, we give the next theorem.

**Theorem 7.2.2** \[ \min\{\mu(G), \mu(H)\} \leq c_0(G+H) \leq \min\{\mu(G) + \gamma_1, \mu(H) + \gamma_2\}. \]

**Theorem 7.2.3** If $n \leq 2$, then \[ c_0(C_m + K_n) = \min\left\lceil \frac{m+1}{2} \right\rceil, n + 1 \] if $n \geq 3$, then \[ c_0(C_m + K_n) = \min\left\lceil \frac{m+1}{2} \right\rceil, n \].

**Proof.** Let $C_m = \{u_1, \ldots, u_m\}$ and $K_n = \{v_1, \ldots, v_n\}$. We first show that \[ c_0(C_m + K_n) \leq \left\lceil \frac{m+1}{2} \right\rceil. \] Consider the case when $m$ is odd. To clear $C_m + K_n$, place cop $\lambda_i$ on $u_{2i-1}$, where $1 \leq i \leq \frac{m+1}{2}$. Hence, we use \( \frac{m+1}{2} \) cops in total. Let cop $\lambda_i$, where $1 \leq i \leq \frac{m-1}{2}$, vibrate on $u_{2i-1}$ and $u_{2i}$ throughout the strategy until all vertices are cleared. Let cop $\lambda_{\frac{m+1}{2}}$ vibrate on $u_m$ and each vertex of $K_n$ to clear all the vertices of $K_n$. When $m$ is even, we can use $\frac{m}{2} + 1$ cops to clear $C_m + K_n$ in a similar way. Therefore, \[ c_0(C_m + K_n) \leq \left\lceil \frac{m+1}{2} \right\rceil. \]

We then show that \[ c_0(C_m + K_n) \leq n + 1 \] if $n \leq 2$. If $n = 1$, then we place two cops on $u_1$ initially. Let one cop vibrate on $u_1$ and $v_1$ until all vertices are cleared. Let the second cop slide around $C_m$ to clear all its vertices. If $n = 2$, then we place cop $\lambda_i$ on $u_i$, where $1 \leq i \leq 3$. Let cop $\lambda_i$, where $1 \leq i \leq 2$,
vibrate on \( u_i \) and \( v_i \) until all vertices are cleared. Let cop \( \lambda_3 \) slide around \( C_m \) to clear all its vertices.

In the last, we show that \( c_0(C_m + K_n) \leq n \) if \( n \geq 3 \). Place cop \( \lambda_i \) on \( v_i \), where \( 1 \leq i \leq n \). Let cop \( \lambda_1 \) vibrate on \( v_i \) and \( u_i \) until all vertices are cleared. Slide \( \lambda_{n-1} \) to \( u_2 \) and slide \( \lambda_n \) to \( u_3 \). If \( m \leq 3 \), then all vertices are cleared. If \( m \geq 4 \), then (1) slide \( \lambda_{n-1} \) back to \( v_{n-1} \) and \( \lambda_n \) back to \( v_n \), and (2) slide \( \lambda_{n-1} \) to \( u_3 \) and slide \( \lambda_n \) to \( u_4 \). All vertices of \( C_m \) can be cleared in a similar way.

In combination of the above and Theorem 7.1.9, we have: (1) if \( n \leq 2 \), then \( c_0(C_m + K_n) = \min\{\lceil \frac{m+1}{2} \rceil, n + 1 \} \), and (2) if \( n \geq 3 \), then \( c_0(C_m + K_n) = \min\{\lceil \frac{m+1}{2} \rceil, n \} \).

**Theorem 7.2.4** For \( 3 \leq m \leq n \), if \( m \) is odd, then \( c_0(C_m + C_n) = \frac{m+1}{2} + 1 \); if \( m \) is even, then \( c_0(C_m + C_n) = \frac{m}{2} + 2 \).

**Proof.** Let \( v^1_i \) denote the \( i \)-th vertex on \( C_m \), where \( 1 \leq i \leq m \). Let \( v^2_j \) denote the \( j \)-th vertex on \( C_n \), where \( 1 \leq j \leq n \).

When \( m \) is odd, we clear \( C_m + C_n \) in the following way. Place cop \( \lambda_i \) on \( v^1_{2i-1} \), for \( i = 1, 2, \ldots, \frac{m+1}{2} \); place cop \( \lambda_{\frac{m+3}{2}} \) on \( v^2_{\frac{m+1}{2}} \). Let cop \( \lambda_i \), where \( 1 \leq i \leq \frac{m-1}{2} \), vibrate on \( v^1_{2i-1} \) and \( v^1_{2i} \) throughout the strategy until all vertices are cleared. Let cop \( \lambda_{\frac{m+1}{2}} \) vibrate on \( v^1_{m} \) and \( v^2_{\frac{m+1}{2}} \) throughout the strategy until all vertices are cleared. Slide cop \( \lambda_{\frac{m+3}{2}} \) along \( C_n \) to clear all its vertices.

When \( m \) is even, in a similar way, we can use \( \frac{m}{2} \) cops to clear all the vertices of \( C_m \), and use two extra cops to clear all the vertices of \( C_n \). Hence \( \frac{m}{2} + 2 \) cops are sufficient for clearing \( C_m + C_n \) when \( m \) is even.
In combination of Theorem 7.1.10, we have that if $m$ is odd, then $c_0(C_m + C_n) = \frac{m+1}{2} + 1$; if $m$ is even, then $c_0(C_m + C_n) = \frac{m}{2} + 2$. □

**Theorem 7.2.5** If $n = 1$, then $c_0(K_m + P_n) = \lceil \frac{m+1}{2} \rceil$; if $n \geq 2$, then $c_0(K_m + P_n) = \lceil \frac{m}{2} \rceil + 1$.

**Proof.** Let $K_m = \{u_1, \ldots, u_m\}$ and $P_n = \{v_1, \ldots, v_n\}$.

If $n = 1$, $K_m + P_1$ is a clique of size $m + 1$. Hence, we can use $\lceil \frac{m+1}{2} \rceil$ cops to clear the graph in one round. In combination of Corollary 7.1.2, $c_0(K_m + P_1) = \lceil \frac{m+1}{2} \rceil$.

If $n \geq 2$, we first consider the case when $m$ is even. The following describes a cop-win strategy that uses $\frac{m}{2} + 1$ cops to clear $K_m + P_n$. Place cop $\lambda_i$ on $u_i$, where $1 \leq i \leq \frac{m}{2}$, and let $\lambda_i$ vibrate on $u_i$ and $u_{i+\frac{m}{2}}$ until all vertices are cleared. Place another cop on $v_1$, and slide it along the path $P_n$ to clear all its vertices. Therefore, $\frac{m}{2} + 1$ cops are sufficient for clearing $K_m + P_n$. Similarly, when $m$ is odd, we can use $\frac{m+1}{2} + 1$ cops to clear $K_m + P_n$. Therefore, we have $c_0(K_m + P_n) \leq \lceil \frac{m}{2} \rceil + 1$. Further, $K_m + P_n$ contains a clique of size $m + 2$ as subgraph. In combination of Corollary 7.1.2, $c_0(K_m + P_1) \geq \lceil \frac{m+2}{2} \rceil = \lceil \frac{m}{2} \rceil + 1$.

□

### 7.2.2 Lexicographic Products of Graphs

**Theorem 7.2.6** For $m \geq 4$, $c_0(P_m \cdot G) = |V(G)|$. 

171
Proof. From Lemma 7.1.11, we have $c_0(P_m \cdot G) \geq |V(G)|$. To complete the proof, we only need to give a strategy that utilizes $|V(G)|$ cops to clear $P_m \cdot G$.

Let $G^1, \ldots, G^m$ denote the $m$ copies of $G$ in $P_m \cdot G$. To clear $P_m \cdot G$, place a cop on each vertex of $G^1$; slide all cops simultaneously from $G^i$ to $G^{i+1}$ for $i \in \{1, \ldots, m-1\}$.

Lemma 7.2.7 For $m \geq 4$, $c_0(P_2 \cdot P_m) = \lceil \frac{m}{2} \rceil + 1$.

Proof. From Lemma 7.1.13, we have $c_0(P_2 \cdot P_m) \geq \lceil \frac{m}{2} \rceil + 1$. So we only need to give a strategy that clears $P_2 \cdot P_m$ with $\lceil \frac{m}{2} \rceil + 1$ cops. Let $P_m^1$ and $P_m^2$ be the two copies of $P_m$ in $P_2 \cdot P_m$. To clear $P_2 \cdot P_m$, we can use $\lceil \frac{m}{2} \rceil$ cops and let them vibrate on vertices in $V(P_m^1)$ to ensure that every vertex in $V(P_m^1)$ remains cleared in at least one of any two consecutive rounds. Then we can use one additional cop to clear all vertices in $V(P_m^2)$.

7.2.3 Complete Multipartite Graphs

Theorem 7.2.8 If there is a perfect matching in $K_{n_1,\ldots,n_k}-V_k$, then $c_0(K_{n_1,\ldots,n_k}) = \frac{1}{2} \sum_{i=1}^{k-1} |V_i| + 1$.

Proof. From Lemma 7.1.17, we have $c_0(K_{n_1,\ldots,n_k}) \geq \frac{1}{2} \sum_{i=1}^{k-1} |V_i| + 1$. To complete the proof, we describe a cop-win strategy for $K_{n_1,\ldots,n_k}$ that uses $\frac{1}{2} \sum_{i=1}^{k-1} |V_i| + 1$ cops.

1. For each edge in $M(H_k)$, place a cop on one endpoint of the edge, and let it vibrate between two endpoints of the edge until all vertices in
2. Select a vertex $v \in V_1$ and place a cop $\lambda$ on it. Then, slide $\lambda$ between $v$ and $V_k$ to clear all vertices in $V_k$.

\textbf{Theorem 7.2.9} If there is no perfect matching in $K_{n_1, \ldots, n_k} - V_k$, then $c_0(K_{n_1, \ldots, n_k}) = \mu(H_k)$.

\textbf{Proof.} From Lemma 7.1.19, we have $c_0(K_{n_1, \ldots, n_k}) \geq \mu(H_k)$. Hence, we only need to describe a cop-win strategy that uses $\mu(H_k)$ cops to capture the robber. Let $U$ denote the vertex set that includes all the vertices in $V(H_k)$ which are not matched by any edge in $M(H_k)$.

1. For each edge in $M(H_k)$, place a cop on one endpoint of the edge, and let it vibrate between two endpoints of the edge until all vertices in $V(K_{n_1, \ldots, n_k})$ are cleared.

2. Place a cop on each vertex in $U$, and use one of the cops to clear all vertices in $V_k$.

Consider the $K_{3,4,5}$ in Figure 7.3. Let $V_1 = \{v_{1,1}, v_{1,2}, v_{1,3}\}$, $V_2 = \{v_{2,1}, v_{2,2}, v_{2,3}, v_{2,4}\}$ and $V_3 = \{v_{3,1}, v_{3,2}, v_{3,3}, v_{3,4}, v_{3,5}\}$. The following gives a cop-win strategy for $K_{3,4,5}$ which uses four cops. Place a cop on each vertex in $V_2$. Then, let the cop, which occupies $v_{2,i}$, $1 \leq i \leq 3$, vibrate between two endpoints of
the edge $v_{2,i}v_{1,i}$. Also, let the cop, which occupies $v_{2,4}$, move along the path
$v_{2,4}v_{3,1}v_{2,4}v_{3,2}v_{2,4}v_{3,3}v_{2,4}v_{3,4}v_{2,4}v_{3,5}$. Clearly, the above strategy uses four cops
to capture the robber.

![Diagram of cop placement]

Figure 7.3: Four cops are sufficient for capturing the robber on $K_{3,4,5}$.

### 7.2.4 Split Graphs

**Lemma 7.2.10** For a split graph $S_{m,n} = (C, I)$, if $m$ is odd, or there is a pair of vertices in $V(C)$ which share no common neighbor in $V(I)$, then $c_0(S_{m,n}) \leq \lceil \frac{m}{2} \rceil$.

**Proof.** The following describes a cop-win strategy that uses $\frac{m+1}{2}$ cops to
clear $S_{m,n}$ when $m$ is odd.

1. Place cop $\lambda_i$ on $u_{2i-1}$, where $1 \leq i \leq \frac{m+1}{2}$. Hence, we totally use $\frac{m+1}{2}$
cops. Let $u' = u_m$, and let $j = m$.  

174
2. Let cop $\lambda_i$, where $1 \leq i \leq \frac{m-1}{2}$, vibrate on $u_{2i-1}$ and $u_{2i}$ throughout this strategy until the robber is captured.

3. For each contaminated vertex $v \in V_2$ that is adjacent to $u'$, slide cop $\frac{\lambda_{m+1}}{2}$ from $u'$ to $v$, then slide it back to $u'$.

4. If $j = 1$, then stop; else, set $j \leftarrow j - 1$. Slide cop $\frac{\lambda_{m+1}}{2}$ from $u'$ to $u_j$, and let $u' = u_j$. Go to Step 3.

In the following, we consider the case when $m$ is even and there is a pair of vertices in $V_1$ which share no common neighbor in $V_2$. Without loss of generality, let $u_1$ and $u_2$ denote a pair of vertices in $V_1$ which share no common neighbor in $V_2$.

1. Place cop $\lambda$ on $u_1$. Place a cop on $u_{2i-1}$, where $2 \leq i \leq \frac{m}{2}$, and let it vibrate on $v_{2i-1}$ and $v_{2i}$ throughout this strategy until all vertices are cleared.

2. Slide $\lambda$ between $u_1$ and each of its neighbors in $V_2$. Then slide $\lambda$ to $u_2$ and clear all its neighbors in $V_2$.

3. Similar to the strategy for odd $m$ given above, slide $\lambda$ to each of its neighbors in $V_1$, and clear its contaminated neighbors in $V_2$.

Lemma 7.2.11  For a split graph $S_{m,n}$, if $m$ is even and every pair of vertices in $V_1$ have a common neighbor in $V_2$, then $c_0(S_{m,n}) \leq \frac{m}{2} + 1$.
Proof. To clear $S_{m,n}$, we place cop $\lambda_i$ on $u_i$, where $1 \leq i \leq \frac{m}{2}$, and let the cop vibrate on $u_i$ and $u_{i+\frac{m}{2}}$ until the end of the strategy; then we place another cop $\lambda'$ on $u_1$, and use it to clear all $u_1$’s neighbors in $V_2$; in the next, we slide $\lambda'$ to $u_i$, for $i = 2, \ldots, m$, to clear all its contaminated neighbors in $V_2$. Obviously, we clear $S_{m,n}$ using a total of $\frac{m}{2} + 1$ cops. ■

By combining Theorem 7.1.21, Lemmas 7.2.10 and 7.2.11, we give the cop number of split graph in the next theorem.

**Theorem 7.2.12** Let $S_{m,n} = (C, I)$ be a split graph with $|V(C)| = m$ and $|V(I)| = n$. (i) $c_0(S_{m,n}) = \left\lceil \frac{m}{2} \right\rceil$, if $m$ is odd or $V(C)$ has a pair of vertices that share no common neighbor in $V(I)$; (ii) $c_0(S_{m,n}) = \frac{m}{2} + 1$, if $m$ is even and every pair of vertices in $V(C)$ have a common neighbor in $V(I)$. 

176
Chapter 8

Conclusions and Open Problems

This chapter will summarize contributions of the thesis and outline some open problems that are worth investigating in the future.

8.1 Summary

This thesis focuses on the fast searching and zero-visibility cops and robber games. We proposed a few new lower bounds on the fast search number. We then applied the new lower bounds to the investigation of several classes of graphs, including Cartesian products of graphs, hypercubes, toroidal grids, etc. We also examined the complete $k$-partite graphs and solved a problem that was open for eight years.

Further, we introduced a partition method which we then used to prove lower bounds for the zero-visibility cops and robber game. We applied this
method to show lower bounds of graph products. We believe that this idea can be used for other classes of graphs. We also considered the zero-visibility cops and robber game on graph joins, as well as a few more generalized graphs and products. We developed a new method for finding lower bounds on the cop number by establishing conditions on search strategies. We found propositions that are always true in a strategy for $G$ if less than $c_0(G)$ cops are used.

The challenge in establishing lower bounds on various graph search parameters typically lie in finding non-trivial lower bounds that apply to more than just a few special graphs. The techniques that we have developed in this thesis can be seen as general methods for finding lower bounds on the fast search number and zero-visibility cop number for a large class of graphs. We believe that these techniques can inspire more lower bound results for other classes of graphs, and potentially also for other graph search parameters.

8.2 Open Problems

We conclude this thesis by listing some open problems that we consider worth investigating.

1. For toroidal grids $C_m \square C_n$ ($n \geq m \geq 4$), we proved that $fs(C_m \square C_n) \leq 2m + n - 3$. We conjecture that $fs(B_m \square C_n) \geq 2m + n - 3$, $n \geq m \geq 4$.

2. We proved that $fs(Q_k) \leq 2^{k-1} + 1$ when $k$ is even and $fs(Q_k) \geq \frac{3}{25} 2^{k+2-\log\sqrt{k}}$ for large $k$. We conjecture that $fs(Q_k) = 2^{k-1} + 1$ when $k$ is even.
(3) In [52], Stanley and Yang showed that $fs(P_m \square P_n) = m + n - 2 \ (m \geq 2, n \geq 2)$. We also believe that it would be interesting to consider algorithms for computing $fs(T_m \square P_n)$, where $T_m$ is a tree with $m$ vertices and $P_n$ is a path with $n$ vertices.

For the zero-visibility cops and robber game, we present the following conjectures.

1. $c_0(C_m \square C_n) = mc_0(C_m \square C_n) = m + 1$, for $n \geq m \geq 3$.

2. $c_0(P_m \boxtimes P_n) = mc_0(P_m \boxtimes P_n) = \frac{m+1}{2} + 1$, where $m$ is odd and $n \geq m \geq 3$.

3. $c_0(C_m \boxtimes P_n) = mc_0(C_m \boxtimes P_n) = \min\{\lceil \frac{m+1}{2} \rceil, 2\lceil \frac{n+1}{2} \rceil\} + 1$, for $m \geq 3$ and $n \geq 2$.

4. $c_0(C_m \boxtimes C_n) = mc_0(C_m \boxtimes C_n) = m + 2$, for $n \geq m \geq 3$.

5. For $n \geq 2$,

\[
c_0(Q_n) = mc_0(Q_n) = \sum_{k=0}^{n-2} \left( \binom{k}{\lfloor \frac{k}{2} \rfloor} \right) + 1.
\]
References


