Extended Exclusive Graph Search

by

Youjiao Sun

B.Sc., Simon Fraser University, 2014
B.Eng., Zhejiang University, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Science

Approval

Name:	**Youjiao Sun**
Degree:	**Master of Science**
Title:	***Extended Exclusive Graph Search***
Examining Committee:	**Chair:** Dr. Ramesh Krishnamurti Professor

Dr. Qianping Gu
Senior Supervisor

Dr. Andrei Bulatov
Supervisor

Dr. Jiangchuan Liu
Internal Examiner

Date Defended:	May 12, 2017

Abstract

Graph search is an important research area in both mathematics and computer science with many practical applications such as eliminating a malicious software in a computer network. The graph search problem can be intuitively described as follows: given a set of searchers and a fugitive in a graph, the searchers and fugitive move from vertices to vertices in the graph alternatively and the searchers try to capture the fugitive which tries to escape from the searchers. A major optimization problem in graph search is to find the minimum number of searchers (called search number) to capture the fugitive. There are several well known graph models: node-search, edge-search, mixed-search and exclusive search. In this thesis, we propose a new search model which is an extension of the exclusive search. We prove the extended exclusive search number for trees. We give the search numbers for the well known graph search models and the extended exclusive search on trees of rings. We also propose heuristic search algorithms for power law graphs based on these models.

Keywords: Graph search, trees, tree of rings, power law graphs

Acknowledgements

I would like to express my deepest gratitude to my senior supervisor Dr. Qianping Gu, without whom the completion of this thesis would not have been possible. Dr. Gu introduced me into the field of algorithmic graph theory and the world of research. His encouraging guidance, thoughtful ideas, meticulous inspections and easy-going character enabled me to explore the unknown and discover the possibilities. I would like to thank Dr. Andrei Bulatov for being my supervisor during my graduate study. I would also like to thank Dr. Jiangchuan Liu for being my examiner and Dr. Ramesh Krishnamurti for taking the time to chair my thesis defense.

I am grateful to Leo liang for the discussions on both my research and courses. Thanks to all my lab mates for their help in my graduate study and the helpful discussions in seminars. Thanks are also due to my friends, who accompanied me, supported me and laughed with me through the three years. Last but not least, I would like to give the most special thanks to my family: my parents. They gave me unconditional support and continuous love throughout my life. Without them, neither my graduate study nor this thesis could be possible.

Table of Contents

List of Figures

Chapter 1

Introduction

The problem of Graph Searching was first introduced by Breisch [6, 7], for solving the problem of rescuing a lost speleologist in a network of caves. Alternatively, graph searching can be defined as a particular type of searchers-and-fugitive game, as follows. Given a graph G, modeling any kind of network, develop a strategy for a team of searchers moving in G resulting in capturing a fugitive. There are no limitations on the capabilities of the fugitive, who can be arbitrary fast, be aware of the whole structure of the network, and be aware of the current positions of the searchers. The objective is to compute the minimum number of searchers required to capture the fugitive in G.

To be more formal regarding the behavior of the fugitive above, it is more convenient to rephrase the problem in terms of clearing a network of pipes contaminated by some gas [35]. In this framework, a team of searchers aims at clearing the edges of a initially contaminated graph. Searchers stand on the nodes of the graph, and can slide along its edges. Moreover, a searcher can be removed from one node and then placed to any other node, that is, a searcher can "jump" from one node to another. Sliding of a searcher along an edge, as well as positioning one searcher at each endpoint of an edge, results in clearing that edge. Nevertheless, if there is a path free of searchers between a clear edge and a contaminated edge, then the former is instantaneously recontaminated. Thus, to actually keep an edge clear, searchers must occupy appropriate nodes for avoiding recontamination to occur.

A search strategy is a sequence of operations that will clear an initially contaminated graph. The search number for a graph G is the minimum number of searchers for which a search strategy exists. For example, one searcher is sufficient to clear a path graph, while two searchers are necessary (and sufficient) in a cycle: the search number of any path is 1, while the search number of any cycle is 2.

The above variant of graph searching is actually called mixed-search. Other classical variants of graph searching are node-search, edge-search, cops-and-robbers problem.

When consider the three basic searching games: edge searching, node searching and mixed searching, the fugitive is invisible and active, which involves placing some restrictions on searchers, but placing no restrictions on the fugitive. In these search games, the discrete time intervals (or time-steps) are introduced. Initially, G contains a fugitive who is located at a vertex in G, and G does not contain any searchers. Each searcher has no information on the whereabouts of the fugitive (i.e., fugitive is invisible), but the fugitive has complete knowledge of the location of all searchers. The goal of the searchers is to capture the fugitive, and the goal of the fugitive is to avoid being captured. The fugitive always chooses the best strategy so that he evades capture. Suppose the game starts at time t_0 and the fugitive is captured at time t_N and the search time is divided into N intervals $(t_0, t_1], (t_1, t_2], ..., (t_{N-1}, t_N]$ such that in each interval $(t_i, t_{i+1}]$ (also called step), exactly one searcher performs one action: placing, removing, or sliding. The fugitive can move from a vertex x to a vertex y in G at any time in the interval $(t_0, t_N]$ if there exists a path between x and y which contains no searcher (i.e., fugitive is active).

In the edge search game introduced by Megiddo et al. [28], 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 end to the other. The fugitive is captured if a searcher and the fugitive occupy the same vertex on G.

In the node search game introduced by Kirousis and Papadimitriou [24], there are two actions for searchers: placing a searcher on a vertex and removing a searcher from a vertex. The fugitive is captured if a searcher and the fugitive occupy the same vertex of G or the fugitive is on an edge whose endpoints are both occupied by searchers.

In the mixed search game introduced by Bienstock and Seymour [9], searchers have the same actions as those in the edge search game. The fugitive is captured if a searcher and the fugitive occupy the same vertex on G or the fugitive is on an edge whose endpoints are both occupied by searchers.

The edge search number of G, denoted by $es(G)$, is the smallest positive integer k such that k searchers can capture the fugitive in the edge search model. Analogously, define the node search number of G (written $ns(G)$) and mixed search number of G (written $ms(G)$).

The node searching problem is equivalent to the gate matrix layout problem and interval graph augmentation problem [27]. The problem of finding the node-search number is equivalent to the pathwidth problem [27, 32], the interval thickness problem [26], the narrowness problem [1], and the vertex separation problem [25, 21]. From the equivalent of the above problems, the node searching problem is NP-complete on planar graphs with vertex degree at most 3 [3], starlike graphs (a proper subclass of chordal graphs) [14], bipartite graphs [22], cobipartite graphs (i.e., complement of bipartite graphs) [37], and bipartite distance-hereditary graphs (a proper subclass of the chordal bipartite graphs and distance-hereditary graphs) [42]. For some special classes of graphs, it can be solved in polynomial time, as e.g., trees [20, 27, 39], cographs [18], permutation graphs [15], trapezoid graphs [16], split graphs [14, 22], partial k-trees [19], and k-starlike graphs for a fixed k [14, 41].

The edge searching problem is equivalent to the min-cut linear arrangement problem for any graph with the maximum degree 3 [12]. The edge searching problem is NP-complete on general graphs [29], planar graphs with the maximum vertex degree 3 [3] and starlike graphs [41]. However, it can be solved in polynomial time on complete graphs [34], trees [29], interval graphs, split graphs, and k-starlike graphs for a fixed $k \geq 2$ [41].

Moreover, the search numbers for all these three models have the following relationship with each other.

Theorem 1.0.1. *[10] If G is a connected graph, then the following inequalities hold*

1. $ns(G) - 1 \leq es(G) \leq ns(G) + 1$
2. $ms(G) \leq es(G) \leq ms(G) + 1$
3. $ms(G) \leq ns(G) \leq ms(G) + 1$

Though the seach numbers for these searching problems appear to be similar, the time complexities to solve them are different. For example, there are linear time algorithms on a tree to find both its node-search number and an optimal node-search strategy [39, 40] (also mentioned in [27], Theorem 4.7). However, the previous best algorithm [29] takes $O(nlogn)$ time to find an optimal edge-search strategy on a tree of n vertices, while its edge-search number can be found in linear time [29].

Cops and Robbers is another interesting model where the fugitive is visible by the searchers. In this two-player game of perfect information, a set of cops tries to capture a robber by moving at unit speed from vertex to vertex. More precisely, Cops and Robbers is a game played on a reflexive graph (i.e., there is a loop at each vertex). There are two players consisting of a set of cops and a single robber. The game is played over a sequence of discrete time-steps or rounds, with the cops going first in round 0 and then playing alternate time-steps. The set of cops is referred to as C and the robber as R: When a player is ready to move in a round, they must move to a neighboring vertex. Because of the loops, players can pass or remain on their own vertex. The cops win if after some finite number of rounds, one of them can occupy the same vertex as the robber (in a reflexive graph, this is equivalent to the cop landing on the robber). This is called a capture. The robber wins if he can evade capture indefinitely. A winning strategy for the cops is a set of rules that if followed, result in a win for the cops. A winning strategy for the robber is defined analogously. If a cop is placed at each vertex, then the cops are guaranteed to win. Therefore, the minimum number of cops required to win in a graph G is a well-defined positive integer, named the cop number (or copnumber) of the graph G: The notation $c(G)$ is used for the cop number of a graph G. If $c(G) = k$, then G is k-cop-win. In the special case $k = 1$, G is cop-win (or copwin).

J. Burman[5] introduced exclusive graph searching model. This model is motivated to address some assumptions used by the node searching, edge searching and mixed searching models that may have limitations in practice.

First, all of node, edge and mixed searching models assume that any node can be simultaneously occupied by several searchers. This assumption may be unrealistic in several contexts. Typically, placing several searchers at the same node may simply be impossible in a physical environment in which, e.g., the searchers are modeling physical robots moving in a network of pipes. In the case of software agents deployed in a computer network, maintaining several searchers at the same node may consume local resources (e.g., memory, computation cycles, etc.). To be more realistic on these situations, it is better to investigate exclusive graph searching, i.e., graph searching bounded to satisfy the exclusivity constraint stating that no two or more searchers can occupy the same node at the same time.

Second, those classic graph searching model also suffer from another unrealistic assumption: searchers are enabled to “jump” from one node of the graph, to another, potentially far away, node. If searcher jump is allowed during the search process, the first restriction would be meaningless. Thus in exclusive searching model, searchers are limited to move along the edges of the graph, that is, restricted to satisfy the internality constraint.

To sum up, exclusive search is mixed searching model by applying the constrain that any node cannot be simultaneously occupied by more than one searcher.

In the exclusive search model, a vertex gets recontaminated when it is exposed to a single dirty path. This may have limitations in practice. We extend the exclusive search model by considering a more general recontamination scenario. In this thesis, we generalize the exclusive search model to better fit the current computer network. We design a polynomial-time algorithm which, given any tree T, computes the search number of T for the extended exclusive search. Moreover, for any integer k, we provide a characterization of the trees T with extended exclusive search number at most k. This characterization allows us to describe a special type of extended exclusive search strategy.

Then we present exclusive search number and extended exclusive search number for graphs of tree of rings. We also give the search numbers for the node-search, edge-search and mixed-search models on tree of rings. Finally, we propose heuristic search algorithms for power law graphs based on the exclusive search, extended exclusive search, node-search, edge-search and mixed search models.

Chapter 2

Preliminaries

In this chapter, we first present important properties of the three classic graph searching model, then review the definition of original exclusive graph searching, and present some basic general properties of exclusive search.

2.1 Graph Search Basics

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. Let $\{u, v\}$ denote an edge of G. For an edge $e = \{u, v\}$, vertices u and v are called the endpoints of e. For a vertex $v \in V$, the node degree of v is the number of edges incident to v. G is connected if for any two vertices of G, there is a path connecting the two vertices. A connected component of G is a maximum connected subgraph of G (if G is connected, G has only one component, G itself). A vertex v is a cut vertex of G if removing v from G gives a graph with at least two connected components.

Theorem 2.1.1. *[10, 24] Given an arbitrary graph G and an integer k, the problem of determining whether $es(G) \leq k$ ($ns(G) \leq k$ or $ms(G) \leq k$) is NP-complete.*

Definition 2.1 A search strategy is monotonic if the set of cleared edges before any step is always a subset of the set of cleared edges after the step.

Megiddo et al. [28] showed that the edge search problem is NP-hard. This problem belongs to NP owing to the monotonicity result of [23], in which LaPaugh showed that recontamination of edges cannot reduce the number of searchers needed to clear a graph.

Monotonicity is an important issue in graph search problems. Bienstock and Seymour [10] proposed a method that gives a succinct proof for the monotonicity of the mixed search problem, which implies the monotonicity of the edge search problem and the node search problem. Fomin and Thilikos [13] provided a general framework that can unify monotonicity results in a unique min-max theorem.

Theorem 2.1.2. *[10, 23] The edge search, node search, and mixed search problems are monotonic.*

Search numbers have close relationships with pathwidth and treewidth [30, 31]. Given a graph G, a tree decomposition of G is a pair (T, W) with a tree $T = (I, F)$, $I = 1, 2, ..., m$ and a family of nonempty subsets $W = W_i \subseteq V : i = 1, 2, ..., m$, such that

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

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

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 a tree decomposition (T, W) is $max|W_i| - 1 : 1 \leq i \leq m$

The treewidth of G, denoted by $tw(G)$, 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 pathwidth of a graph G, denoted by $pw(G)$, is the minimum width over all path decompositions of G. More information on treewidth and related problems can be found in the survey papers [12, 36].

2.2 The Results for Trees

It follows from results of Parsons in [11] that the mixed graph-searching problem, when restricted to trees, can be solved in polynomial time.

Claim 2.2.1. *[35] For any tree T and integer $k \geq 1$, $ms(T) \geq k + 1$ if and only if T has a vertex v at which there are three or more branches that have search number k or more.*

Notice that this result implies that a tree with mixed search number k must have at least 3^{k-1} edges, hence we have the following property:

Property 2.2.1. *The mixed search number of an n node tree always satisfies $ms(T) \leq 1 + \log_3(n - 1)$*

This result leads us to the key concept also needed for other graph searching theorem, the concept of the "avenue" of a tree. Intuitively, an avenue of a tree T is a path $v_l, v_2, ..., v_r$ of two or more vertices such that T can be cleared using $ms(T)$ searchers by placing a searcher on v_l and subsequently moving it along the avenue to $v_2, v_3, ..., v_r$, pausing long enough at each vertex v_i along the path so that the nonavenue branches at v_i can be cleared using the remaining $ms(T) - 1$ searchers. Formally, a path $v_l, v_2, ..., v_r$ of two or more vertices is an avenue for T if v_l and v_r, each have exactly one branch with search number $ms(T) = s$ (containing v_2 and v_{r-1} , respectively) and for every j, $2 \leq j \leq r-1$, v_j has exactly two branches with search number s (containing v_{j-1} and v_{j+1}, respectively). It is not hard to see that this definition implies that the avenue can be used inductively to search T with $ms(T)$ searchers in the manner indicated above.

2.3 Exclusive Search and Properties

J. Burman[5] first introduced the exclusive search model. Here we formally review the problem of exclusive graph searching and its unique properties.

Given a connected n-node graph G, an exclusive search strategy in G, using $k \leq n$ searchers consists: 1) placing the k searchers at k different nodes of G and 2) performing a sequence of moves, A move is to slide one searcher from one endpoint u of an edge $e = \{u, v\}$ to its other endpoint v. Such a move can be performed only if v is free of searchers. That is, exclusive search limits the strategy to place at most one searcher at each node, at any point in time. The edges of graph G are supposed to be initially contaminated. An edge becomes clear whenever either a searcher slides along it, or one searcher is placed at each of its endpoints. An edge becomes recontaminated whenever there is a path free of searchers from that edge to a contaminated edge. A search strategy is winning if its execution results in all edges of the graph G being simultaneously clear. The exclusive search number of G, denoted by $xs(G)$ is the smallest k for which there exists a winning search strategy for searchers in G.

Now, we formally state and explain the main differences between exclusive search and all classical variants of graph searching. These differences are mainly due to the combination of the two restrictions introduced in exclusive search: two searchers cannot occupy the same node (exclusivity) and a searcher cannot “jump” (internality). Therefore, intuitively, the difficulty occurs when a searcher has to go from one node u to a far away node v, and all paths from u to v contain a node with a searcher.

Consider a simple example of a star with central node c and n leaves. In the classical graph searching, we can use one searcher to occupy c, while a second searcher will sequentially clear all leaves, either by jumping from one leaf to another, or by sliding from one leaf to another, and therefore occupying several times the already occupied node c. In exclusive graph searching, such strategies are not allowed. Intuitively, if a searcher r_1 has to cross a node v that is already occupied by another searcher r_2, the latter should step aside for letting r_1 pass. However, r_2 may occupy v to preserve the graph from recontamination, and moving away from v could lead to recontaminate the whole graph. To avoid this, it may be necessary to use extra searchers (compared to the classical graph searching) that will guard several neighbors of v to prevent from recontamination when r_2 gives way to r_1. It follows that, as opposed to all classical search numbers, which differ by at most some constant factor, the exclusive search number may be arbitrary large compared to the mixed-search number, even in trees. For instance, it is easy to check that $xs(S_n) = n - 2$ for any n-node star $S_n, n \geq 3$. A more general result is stated below:

Property 2.3.1. *[5] For any tree T with maximum degree $\Delta \geq 2, xs(T) > \Delta - 2$*

Proof. Since later proofs use similar techniques, we include the prove details of [5] here.

A more general result of the claim is that : Let G be any connected graph with a cut-vertex v and let $cc(v) \geq 2$ be the number of the connected components in $G \backslash \{v\}$. Then $xs(G) > cc(v) - 2$(here we can imagine replacing each edge of the star by a connected component). We prove the general result by induction. The result clearly holds for $cc(v) = 2$ since any strategy must use at least one searcher to clear any graph with at least one edge. Therefore, we may assume that $cc(v) > 2$.

Let v be a cut-vertex of a graph G and consider any strategy using at most $cc(v)-2$ searchers. Initially, at least two components U and W of $G \backslash \{v\}$ are unoccupied and so all edges in these components are contaminated. Let us, consider the first step when a searcher occupies a node in one of these components, say U. That is, let us consider the first step when a searcher slides from v to a node in U(in order to move a searcher into U, the searcher must be moved to v first). But after this step, no searchers are occupying a node in W which is still contaminated, no searcher is occupying v and there is a connected component C of $G \backslash (v \cup U \cup W)$ that contains no searchers. Hence, at any step of the strategy, at least two connected components of $G \backslash \{v\}$ remain contaminated. Property 2.3.1 is implied by the above general result. □

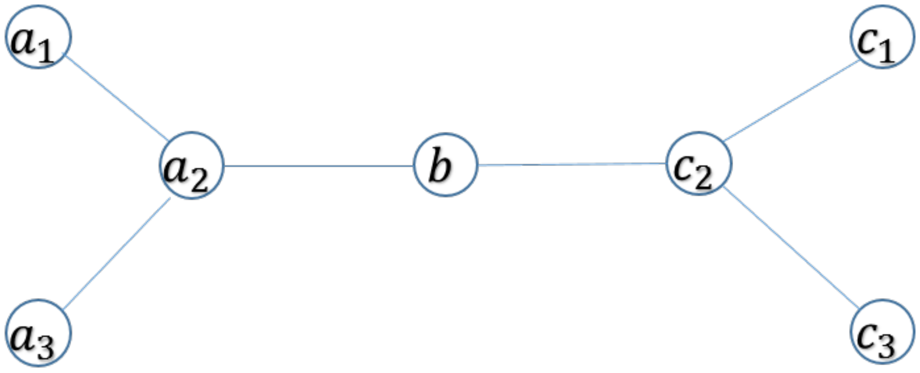
Property 2.3.2. *[5] For any connected graph G with maximum degree Δ, $ms(G) \leq xs(G) \leq (\Delta - 1)(ms(G) + 1)$*

A proof for this property is that given a search strategy S for edge, node or mixed search on G, S can be mimicked by a team of $\Delta - 1$ searchers for each searcher in S.

Property 2.3.3. *[5] Exclusive graph searching is not monotonic*

Proof. Since the model we proposed will use a similar technique, we include the prove of [5] here.

Figure 2.1: A possible tree structure where optimal exclusive search strategy is not monotonic



It can be easily seen that the given tree above can not be cleared by only using one searcher. Here we demonstrate a winning strategy of exclusive search with two searchers in the graph in Figure 2.1.

The strategy works like the following : choose a_1 and a_3 as initial position and place our two searchers X, Y on the initial position respectively. Then slide Y along the edges of the path from a_3 to c_2, X slides along the edges from a_1 to b, Y goes to c_3 and finally X goes to c_1. As we can see, during our strategy when Y goes to c_3, edge (b, c_2) becomes recontaminated.

We just showed that we have a winning strategy for two searchers. Now consider any winning exclusive search strategy with two searchers, there has to be a step with recontamination. There are two cases to be considered. Either the set of initial positions contains no nodes in $C = \{c_1, c_2, c_3\}$(or symmetrically in $A = \{a_1, a_2, a_3\}$), or one searcher initially occupies a node in C and the other searcher occupied a node in A.

In the first case, no nodes in C are occupied initially, consider the first step when c_1 is occupied. This can be done only by moving a searcher along the edge(c_2, c_1).

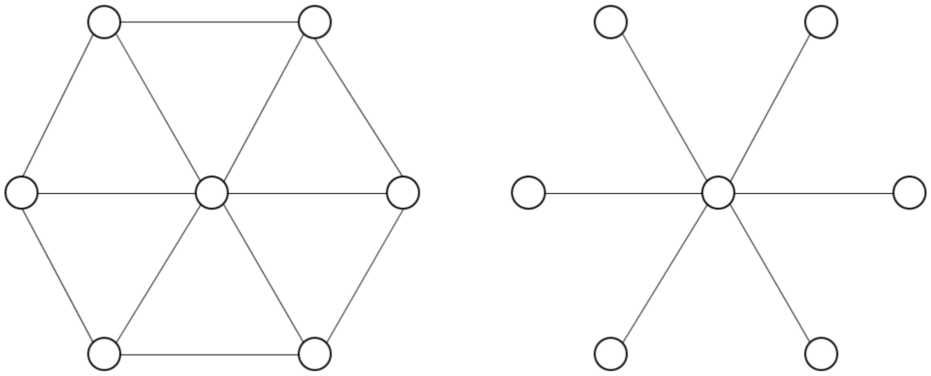
But then the edge(b, c_2) is recontaminated by (c_2, c_3)(since c_3 has never been occupied yet).

In the second case, consider the first searcher to reach b, suppose it comes from a_2. Then it is easy to verify that (a_2, b) is recontaminated because a single searcher cannot have cleaned (a_1, a_2) and (a_3, a_2) simultaneously. $\square$

Property 2.3.4. *[5] For any tree T and any subtree T' of T, $xs(T') \leq xs(T)$.*

Property 2.3.4 is not true for general graphs. Taking a subgraph can decrease the connectivity which may not help reduce the used of searchers (due to the exclusivity constraint). That is, there exist a graph G and a subgraph H of G such that $xs(H) > xs(G)$. Consider the following example in Figure 2.2:

Figure 2.2: An example shows that cleaning subgraph may need more searchers in exclusive search



In the figure, the graph on the left can be cleared by placing one searcher on the center and another two searchers walking along the boundary edges. While the graph on the right needs five searchers as we already discussed in Property 2.3.1.

Contrary to classical graph searching, the proof of Property 2.3.4 is not trivial because of the exclusivity property. To prove it, Janna Burman[5] transformed an exclusive strategy S for T into a strategy S' for T' using the same number of searchers, and without violating the exclusivity property. Details are omitted here.

Exclusive graph searching is different from classic graph searching, for at least two reasons. First, it does not satisfy the monotonicity property. That is, there are graphs (even trees) in which every exclusive search strategy using the minimum number of searchers requires to let recontamination occurring at some step of the strategy. Second, exclusive graph searching is not closed under taking minor (not even under subgraph). That is, there are graphs G and H such that H is a subgraph of G, and $xs(H) > xs(G)$. The absence of these two properties makes exclusive-search considerably different from classical search, and its analysis requires introducing new techniques. The following is the main result for exclusive search model on trees.

Theorem 2.3.1. *[5] Let $k \geq 1$. For any tree T, $xs(T) \leq k$ if and only if, for any node v, the following three properties hold:*

1. *v has degree at most $k+1$;*
2. *for any branch B at v, $xs(B) \leq k$;*
3. *for any even $i > 1$, at most i branches B at v have $xs(B) \leq k - i/2 + 1$.*

Chapter 3

Extended Exclusive Search

In this chapter, we propose extended exclusive search model. In the previous exclusive search, when a node is exposed to a single contaminated link, the node becomes recontaminated. In a real network, however, each node may have a certain capability to prevent contamination and may not become recontaminated when the number of contaminated edges the node is exposed to is below some threshold value. To address a more general exclusive search as stated above, we propose a new search model which is an extension of the previous exclusive search.

3.1 Extended Exclusive Search Basics

First we formally state extended exclusive search model.

Given a connected graph G of n nodes, an extended exclusive search strategy with recontamination constraint m in G, using $k \leq n$ searchers satisfying:

1. Placing the k searchers at k different nodes of G

2. Performing a sequence of moves. A move consists of sliding one searcher from one end node u of an edge $e = \{u, v\}$ to the other end node v;

Such a move can be performed only if v is free of searchers. Here we still limit the strategy to place at most 1 searcher at each node, at any time. The edges of graph G are supposed to be initially contaminated. An edge becomes clear whenever either a searcher slides along it, or one searcher is placed at each of its extremities. An edge is contaminated if one end node of the edge is contaminated. A node can be recontaminated if at least m edges incident to it are contaminated. An edge becomes recontaminated whenever one of its end node is recontaminated. A search strategy is winning if its execution results in all edges of the graph G being simultaneously clear.The extended exclusive-search number of G, denoted by $xs_m(G)$ is the smallest k for which there exists a winning search strategy in G.

For the extended exclusive search on trees, we have the following properties:

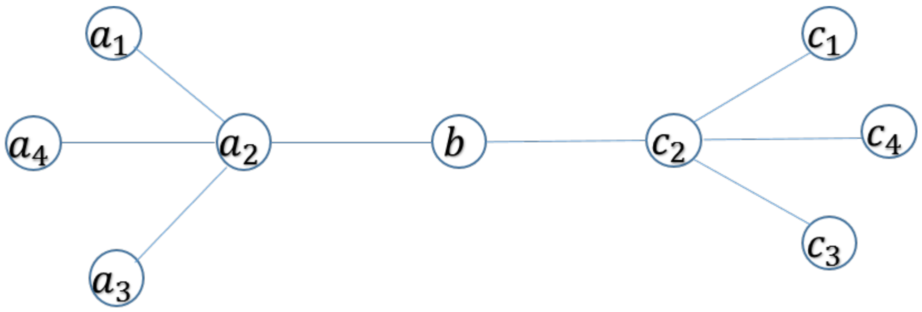
Property 3.1.1. *For any tree T with maximum degree $\Delta \geq 2$ and recontimination constrain m, $xs_m(T) > \Delta - m - 1$*

Proof. We consider a connected graph G with a cut-vertex v and let $cc(v) \geq 2$ be the number of the connected components in $G \backslash \{v\}$. The result clearly holds for $cc(v) = 2$ since m is at least 1 and any strategy must use at least one searcher to clear any graph with at least one edge. Therefore, we assume that $cc(v) > 2$.

Let v be a cut-vertex of a graph G. We consider any strategy using at most $cc(v) - m - 1$ searchers. Initially, at least $\Delta - (\Delta - m - 1) = m + 1$ components $\{U_0, .., U_m\}$ of $G \backslash \{v\}$ are unoccupied and so all edges in these components are contaminated. Assume that t_i is the first step that a searcher slides from v to one of these components, say U_0. After t_i, no searchers are occupying a node in m connected components which are still contaminated, no searcher is occupying v and there is a connected component C of $G \backslash (v \cup \cup_{i=0}^{m} U_i)$ that contains no searchers. Hence, C is fully recontaminated. Hence, at any step of the strategy, at least $m + 1$ connected components of $G \backslash \{v\}$ remain contaminated. Thus in extended exclusive search model, for any tree T with maximum degree $\Delta \geq 2$, $xs_m(T) > \Delta - m - 1$ □

Property 3.1.2. *The extended exclusive graph search model is not monotonic.*

Figure 3.1: A possible tree structure where extended exclusive search strategy is not monotonic



Proof. Here we first utilize the proof idea of Property 2.3.3. Let recontamination constrain m to be 2. By Property 3.1.1, the number of searchers to clear this graph is at least 2. Here we use Figure 3.1 to demonstrate a winning strategy of extended exclusive search with two searchers that works like the following : choose a_1 and a_3 as initial position and place our two searchers X, Y on the initial position respectively. Then slide Y along the edges of the path from a_3 to c_2, X slides along the edges from a_1 to a_4 then to b, Y goes to c_3 and finally X goes to c_1, back to c_2, goes to c_4. As we can see, during our strategy when Y goes to c_3, edge (b, c_2) becomes recontaminated.

The above shows that we have a winning strategy with two searchers. Now consider any winning exclusive search strategy with two searchers, there has to be a step with recontamination. There are two cases to be considered. Either the set of initial positions contains no nodes in $C = \{c_1, c_2, c_3, c_4\}$(or symmetrically in $A = \{a_1, a_2, a_3, a_4\}$), or one searcher initially occupies a node in C and the other searcher occupied a node in A.

In the first case, no nodes in C are occupied initially. Consider the first step when c_1 is occupied. This can be done only by moving a searcher along the edge(c_2, c_1). But then the edge(b, c_2) is recontaminated by (c_2, c_3) and (c_2, c_4)(since two dirty edges connected to c_2 has never been occupied yet).

In the second case, consider the first searcher to reach b, suppose it comes from a_2. Then it is easy to verify that (a_2, b) is recontaminated because a single searcher cannot have cleaned (a_1, a_2), (a_4, a_2) and (a_3, a_2) simultaneously.

A more general proof idea is to consider a graph that two symmetric star structure connected by a single path of more than one vertex. The analysis is similar to the description above. □

Now we present a polynomial-time algorithm which, given any tree T, computes the extended exclusive search number $xs_m(T)$ of T as well as an exclusive search strategy enabling $xs_m(T)$ searchers to clear T.

This algorithm is based on a characteristics of trees with exclusive search number at most k. Given a node v in a tree T, a connected component of $T \backslash \{v\}$ is called a branch at v. Our characterization establishes a relationship between the extended exclusive-search number of T and the extended exclusive-search number of some of the branches adjacent to any node in T. More precisely, we prove that:

Theorem 3.1.1. *Let $k \geq 1$. For any tree T, $xs_m(T) \leq k - m + 1$ if and only if, for any node v, the following three properties hold:*

1. *v has degree at most $k + 1$;*

2. *for any branch B at v, $xs_m(B) \leq k - m + 1$;*

3. *for choosing $k - m + 2$ branches, for any even $i > 1$, at most i branches B at v have $xs_m(B) \geq k - i/2 - (m - 1) + 1$.*

To prove the theorem, we first prove that, for any tree T and $k \geq 1$, $xs_m(T) \leq k - m + 1$, only if the conditions of Theorem 3.1.1 are satisfied. Then, we show that any tree satisfying these conditions can be decomposed in a particular way, depending on k. Next, we describe an extended exclusive search strategy using at most $k-m+1$ searchers, that clears any tree decomposed in such a way.

3.2 Necessary Conditions for Theorem 3.1.1

We first prove that the conditions of Theorem 3.1.1 are necessary. The fact that the first property is necessary directly follows from the following claim:

Claim 3.2.1. *For any tree T with maximum degree Δ and recontaminating constrain m, $xs_m(T) > \Delta - m - 1$.*

The claim can be directly derived by Property 3.1.1.

The second property is necessary by the following claim.

Claim 3.2.2. *For any tree T and any subtree T' of T, $xs_m(T') \leq xs_m(T)$.*

Proof. Contrary to classical graph searching, the proof of this result is not trivial because of the constraint that no vertex can have more than one searcher on it. To prove it, we have to transform an exclusive strategy S for T into a strategy S' for T' using the same number of searchers, and without violating the exclusivity property. The fact that S may be not monotone (i.e., some recontamination may occur during S) makes the proof technical, because one has to "control" the recontamination of T' in S'.

An easy way to get the idea is like this: Given a graph G and its subgraph H, if $xs_m(H) > xs_m(G)$, then for any extended exclusive search strategy that clears H, there has to be some moves of searchers resulting in clearing the edges in $G \backslash H$. However, any tree T and its subtree T' is connected by a single vertex. Thus the moves of searchers within H would never help clearing $T \backslash T'$, which makes clearing T' a sub problem of clearing T. □

The idea to show the third property in the theorem is necessary is to reduce the problem to a smaller scope and use the strategy from the exclusive graph search with

$m = 1$. Suppose that we are able to clean $k-m+2$ branches that are connected to node v. Then there are only $m-1$ branches left to be cleaned. Consider the recontamination constrain stated in the definition, node v will never get recontaminated again. Then the problem becomes easy, we can simply clean remaining branches one by one and finally finish cleaning the whole tree. Thus the key point is to clean a sub-graph with $k-m+2$ branches. We have reduced the original problem to a problem that exclusive graph search a sub-graph with $k - m + 2$ branches.

We prove the following claim to show the third condition

Claim 3.2.3. *Let $k \geq 1$. For any tree T, if there exists $v \in T$ and an even integer $i > 1$ such that any branch B at v, $xs_m(B) \leq k - m + 1$ and there is a set $B = \{T_j : xs_m(T_j) \geq k-i/2-(m-1)+1\}$ of branches at v and $|B| > i$, then $xs_m(T) > k-m+1$.*

Proof. Let S be any extended exclusive strategy that clears T. For the sake of contradiction, suppose S uses at most $k - m + 1$ searchers. Then, at some step, at least $k - i/2 - (m - 1) + 1$ searchers are in T_j of B in order to clear the branch according to the definition of set B. Let s_j be the last such step of S that occurs in T_j . Without loss of generality, assume that $s_i < s_j$ for any $1 < i < j \leq |B|$. Then we may assume before step s_j , T_j is not completely clear. Then at step $s_{i/2+1}$, at least $k-i/2-(m-1)+1$ searches are in $T_{i/2+1}$, some edges have been cleared in T_j for any $j \leq i/2$ and T_j cannot become fully contaminated again.(Otherwise there would be another step after s_j that has $k-i/2-(m-1)+1$ searches in T_j). Now consider step $s_{i/2+1}$, at least $k-i/2-(m-1)+1$ searches are in $T_{i/2+1}$ and there are at most $i/2-1$ searches outside $T_{i/2+1}$. What we have here is that there is at least one branch at v in $T_{i/2+2}, ...T_{|B|}$ with still contaminated edges(at least $m+i-i/2-1-(i/2-1) = m$ branch at v in $T_{i/2+2}, ...T_{|B|}$ is not guarded),and at least one branch at v in $T_1, ...T_{i/2}$ with (at least) some clear edges that must not be recontaminated and no searchers occupy nodes in both these branches. $\square$

By Claim 3.2.1, 3.2.2, 3.2.3 together, we have proved the necessary conditions of Theorem 3.1.1.

3.3 Sufficient Proof

Claim 3.3.1. *Given a Tree T, for any node v that satisfies*

1. *v has degree at most $k+1$;*
2. *for any branch B at v, $xs_m(B) \leq k-m+1$;*
3. *for choosing $k-m+2$ branches, for any even $i > 1$, at most i branches B at v have $xs_m(B) \geq k - i/2 - (m-1) + 1$.*

then T can be cleaned by using at most $k-m+1$ searchers in the extended exclusive search model.

Proof. Here we still take advantage of the avenue structure but show the differences with classic search strategy when it comes to the extended exclusive search procedure.

Recall the definition of avenue. First we use $k-m+1$ searchers to clean the branch that is connected to v_l(the left most vertex of the avenue), since for any branch B at v, $xs_m(B) \leq k-m+1$, we have enough searchers to clean the branch. Then we move the searchers out through v_l to v_2 one by one, during this movement, we clean the avenue edge (v_l, v_2) and leave a guard searcher on v_2. Then in order to minimize the use of searchers, we first choose $k-m+2$ branches that are connected to v_2 that satisfy the condition 3. Sort all the chosen branches in non-increasing order of xs_m, suppose the order is $\{B_1, B_2, ..., B_{k-m+2}\}$. We partition all the branches into two sets $S_1 = \{B_1, B_3, ...\}$ and $S_2 = \{B_2, B_4, ...\}$. After the partition is done, the clean process is divided into two steps. Set S_1 is cleaned first in non-increasing order of their xs_m, during this process, we leave one searcher to guard the previous cleaned branch as we proceed. Since the third property restricts the structure of the tree, the searchers we need to clean the next branch is always decreasing.

After clearing the branches in S_1, there are searchers currently blocked in the roots of the cleared branches. In order to reuse these searchers to clear the remaining branches, we would like the roots of the contaminated branches occupied to prevent recontamination of the cleared subtrees. That is, we would like to switch the searchers from guarding the cleared branches to guard the contaminated branches.

Consider the last time step when we finish cleaning the last branch of S_1, since the size of S_1 is at most $\frac{k-m+1}{2}$ and we have $k-m+1$ searchers, so we have at least $\frac{k-m+1}{2}$ searchers that are in the last branch of S_1. Since we know the size of S_2 is at most the size of S_1, this means the searchers in the last branch is enough to guard all the branches in S_2. After it is done, there are only $m-1$ dirty edges connected to v_2, we can safely move the guard searchers on the branches of S_1 to S_2 without worry about recontamination.

Then S_2 is cleaned in non-decreasing order of xs_m. The procedure is similar to step 1, we gain new searchers as we proceed since we can reuse the previous guard searcher. By the first condition, v has degree at most $k+1$. By the second condition, the searchers needed for each branch is at most $k-m+1$. And the third condition guarantees that during each step the searchers needed is only increased by one. After we finish clean all the chosen dirty branches, there are only $m-1$ dirty edges connected to v_2 which would never cause recontamination, so we can use the remaining searchers to clean each branch one by one. Then we move on to each vertex on the avenue, follow these steps and eventually clean the whole graph. □

By Claim 3.3.1, the sufficient conditions of Theorem 3.1.1 are proved.

The time complexity to check the properties in Theorem 3.1.1 is $O(n)$ where n is the number of vertices in T. Basically for each vertex v in T, we need $xs_m(B)$ for every branch B which is connected to v. We can do the check starting from the leaf vertices, the extended exclusive search number for a single vertex is 1, then by following a bottom up merging strategy, we can finally get the extended exclusive search number for every branch in T.

Chapter 4

Graph Search on Tree Of Rings

In this chapter, we show the search numbers for several well studied graph search models on networks, called tree of rings, which are supergraphs of trees and have important applications in computer networks and parallel computing. Especially, we generalize the exclusive and extended exclusive search strategies for trees to trees of rings. The motivation is that, we would like to extend the search algorithms for trees to be viable on graphs with cycles.

4.1 Tree Of Rings Basics

A ring network is a cycle with at least three vertices. A tree of rings can be defined as follows:

Definition 4.1.1 [11] A single ring is a tree of rings, and the graph obtained by adding a vertex-disjoint ring R to an existing tree of rings G and then merging one vertex of R and one vertex of G into one is also a tree of rings.

In a tree of rings, any two rings have at most one vertex in common, and for any pair (u, v) of vertices in the tree of rings there are exactly two edge-disjoint paths between u and v. A tree of rings is a practical topology, with several sub-rings connected to a main ring, and sub-subrings to the sub-rings, and so on. It remains connected even if an arbitrary link fails in each ring, and thus provides a better fault tolerance than a tree network.

A tree of rings network is denoted by a graph TR with vertex set $V(TR)$ and edge set $E(TR)$. We use a path for a simple path in TR (i.e. repetition of nodes is not allowed). Two paths in TR intersect if they have a common edge. Two paths in

TR are edge-disjoint if they do not intersect. A set of paths in TR is edge-disjoint if any two paths in the set do not intersect. We say a path is on an edge if the path contains the edge.

Property 4.1.1. *[43] For any node $u \in V(TR)$, a path on u can be on at most two rings which contain u.*

Definition 4.1.2 Given a vertex v in TR, v has ring degree k if v is the common vertex of k rings.

It can be easily seen ring degree of vertex v is half of the degree of vertex v. Also, we give the definition of a branch in tree of rings to make it consistent with the graph theory of tree.

Definition 4.1.3 For any vertex in a tree of rings, consider two edges a and b that are connected to v, if those two edges are in the same ring, then a branch of v is the connected component that is reachable from v and pass through a or b.

4.2 Exclusive Search on Tree of Rings

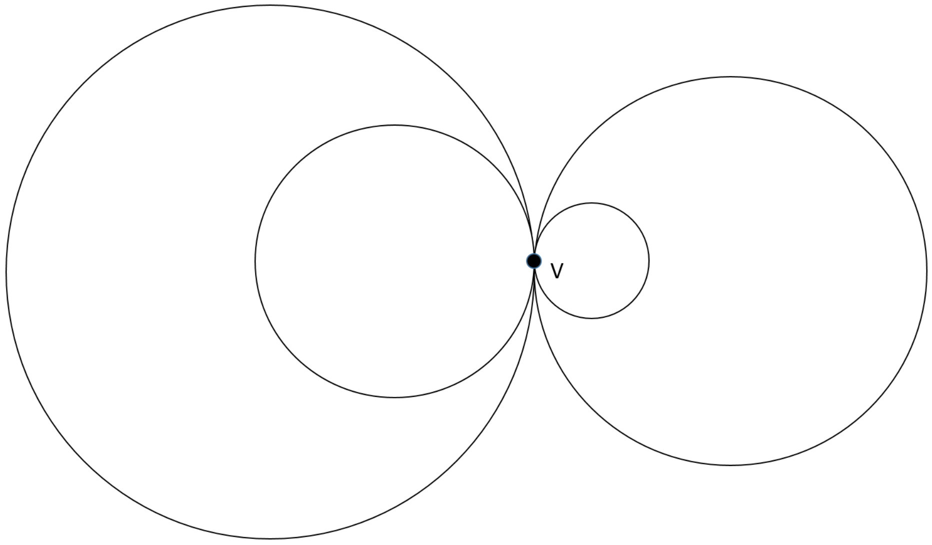
An interesting observation is that instead of only need one searcher to guard a single branch in the case of trees, in tree of rings we need two searchers to guard a single branch from recontamination. This makes guarding and the reuse of searchers relatively expensive compared to the case of trees.

Consider the following graph where multiple single rings are merged to a single vertex.

Suppose v is the common vertex. It is easy to see a possible exclusive search strategy is to place a searcher on the common vertex v and assign each ring with a unique searcher to clean the rings. If we do not allow the guard searcher to be removed from v, we are always using $\Delta + 1$ searchers. For the rest of this chapter, we define this kind of search strategy to be lazy search.

Then we consider the search strategy that the guard searcher steps aside from v in order to reuse searchers. If we choose to guard the rings that are already cleaned, for each ring we need two searchers. On the other hand if we choose to guard the rings that haven't been cleaned, we also need two searchers for each ring. If the degree of v is even, as the search strategy proceed, there is always a time when the number of cleaned rings is equal to the number of dirty rings. In this case we need at least

Figure 4.1: An example of Tree of Rings where all rings are merged to a single vertex



Δ guard searchers and there has to be extra searchers in order to make the search proceed. When the degree of v is odd, analysis is similar. Similarly, define this type of search strategy to be searcher reuse strategy.

The analysis above indicates that for some tree of rings graphs, the optimal solution does not require reusing searchers. Thus our result consists of two parts.

Given a tree of rings TR, let v be a vertex in TR, denote $xs(v, B)$ to be the exclusive search number of B which is a branch of v, $\sum xs(v, B)$ to be the sum of exclusive search number of every branch of v.

We first present our result for exclusive search on tree of rings graphs.

Theorem 4.2.1. *Let $k \geq 1$. For any tree of rings TR, $xs(TR) \leq k$ if and only if*

I $\exists v \in TR, \sum xs(v) < k$ or

II for every vertex v, the following three properties hold:

1 v has ring degree at most $\left\lfloor \frac{k+1}{2} \right\rfloor$;

2 for any branch B at v, $xs(B) \leq k$;

3 for any even $i > 1$, at most i branches B at v have $xs(B) \geq k - i + 1$

We first show that the conditions of Theorem 4.2.1 are necessary. If $xs(TR) \leq k$, we need to consider lazy search strategy or searcher reuse strategy. Consider using lazy search strategy, then there exists a vertex v in TR such that $\sum xs(v) < k$. Otherwise, we check if we can finish clearing the graph using search reuse strategy. The fact that the property (1) is necessary directly follows from the following claim:

Claim 4.2.1. *For any tree of rings TR with maximum ring degree Δ, $xs(TR) > 2 \times (\Delta - 1)$ if we only consider the searcher resue strategy*

Proof. Again we consider the simple case where multiple single rings are merged to a single vertex. Suppose v is the common vertex.

Since for each branch we need two guard searchers and there are Δ branches. We know we have use $(\Delta - 1) \times 2$ searchers as guard searchers. And there has to be extra searchers in order to make the search proceed.

□

The property (2) is necessary by the following claim:

Claim 4.2.2. *Any tree of rings TR and any subgraph TR' of TR, $xs(TR') \leq xs(TR)$.*

Proof. Restricted by the structure of tree of rings, any TR and its subgraph TR' can only share a single common vertex. Consider any search strategy that cleans TR', the searchers in TR' can only clean the edges in TR', in order to clean the edges in $TR\backslash\{TR'\}$, the searchers has to move from TR' to $TR\backslash\{TR'\}$ through the shared common vertex, this makes the problem of cleaning TR' a sub problem of cleaning TR. Thus if subgraph TR' uses $xs(TR')$ searches, the TR can not be cleaned with less resources. □

We then prove the following claim to show condition (3) is necessary

Claim 4.2.3. *Let $k \geq 1$. For any tree of rings TR, if there exists $v \in TR$ and an even integer $i > 1$ such that there is a set $B = \{TR_j : xs(TR_j) \geq k - i + 1\}$ of branches at v and $|B| > i$, then $xs(TR) > k$.*

Proof. To show the claim, let S be any exclusive strategy that clears TR. For the sake of contradiction, suppose S uses at most k searchers. Then, at some step, at least $k-i+1$ searchers are in TR_j of B in order to clear the branch according to the definition of set B. Let s_j be the last such step of S that occurs in TR_j . Without loss of generality, assume that $s_i < s_j$ for any $1 < i < j \leq |B|$. Then we may assume before step s_j , TR_j is not completely clear. Then at step $s_{i/2+1}$, at least $k-i+1$ searches are in $TR_{i/2+1}$, some edges have been cleared in TR_j for any $j \leq i/2$, TR_j cannot become fully contaminated again.(Otherwise there would be another step after s_j that has $k-i+1$ searches in TR_j). Now consider step $s_{i/2+1}$, at least $k-i+1$ searches are in $TR_{i/2+1}$ and there are at most $i-1$ searches outside $TR_{i/2+1}$. What we have here is that there is at least one branch at v in $TR_{i/2+2}, ...TR_{|B|}$ with still contaminated edges(at least one branch at v in $TR_{i/2+2}, ...TR_{|B|}$ is not guarded),and at least one branch at v in $TR_1, ...TR_{i/2}$ with (at least) some clear edges that must not be recontaminated and no searchers occupy nodes in both these branches □

By Claims 4.2.1−4.2.3, we have proved the necessary conditions of Theorem 4.2.1. Next we prove that the conditions in Theorem 4.2.1 are sufficient.

Claim 4.2.4. *Given a Tree of rings TR, if*

I $\exists v \in TR, \sum xs(v) < k$ *or*

II for every vertex v, the following three properties hold:

1 v has ring degree at most $\left\lfloor \frac{k+1}{2} \right\rfloor$*;*

2 for any branch B at v, $xs(B) \leq k$*;*

3 for any even $i > 1$, at most i branches B at v have $xs(B) \geq k - i + 1$

we have an exclusive search strategy using k searchers.

Proof. if there exists a vertex v in TR, $\sum xs(v) < k$, we can place a searcher on v and assign enough searchers for each branch that is connected to v. Otherwise, we define the searcher strategy as following.

First sort all the branches in non-increasing order of xs, suppose the order is $\{B_1, B_2, ...\}$. We partition all the branches into two sets $S_1 = \{B_1, B_3, ...\}$ and $S_2 = \{B_2, B_4, ...\}$. After the partition is done, the clean process is divided into two steps. Set S_1 is cleared first in non-increasing order. During this process, we leave two searchers to guard the previous cleared branch as we proceed. Some of these selected branches may use lazy search strategy, thus search number for a selected branch may be smaller than 2, however, since v has ring degree at most $\left\lfloor \frac{k+1}{2} \right\rfloor$, we always have enough searchers to guard each branch. Also the third property restricts the structure of the TR, the searchers we need to clean the next branch is always decreasing. Then we switch the searchers from guarding the cleared branches to guarding the contaminate branches. After all contaminate branches are guarded, we can move on to step 2

In step 2, S_2 is cleaned in non-decreasing order. It is similar to step 1 since we can reuse the guard searchers as we procceed. After we clear B_2 we have finished cleaning the given tree of rings. □

By Claims 4.2.1-4.2.4, Theorem 4.2.1 is proved.

Similarly, the property check can be done in $O(n)$.

4.3 Extended Exclusive Search on Tree of Rings

Now we apply our extended exclusive search strategy on Tree of Rings

An interesting observation is that if the recontamination constrain m is even, that means at least $\frac{m}{2}$ incident dirty branches would cause recontamination. Otherwise, $\left\lceil \frac{m}{2} \right\rceil$ dirty branches would cause recontamination.

Here we still consider two type of tree of rings graphs.

Theorem 4.3.1. *Let $k \geq 1$. For any tree of rings TR, $xs_m(T) \leq k - \left\lceil \frac{m}{2} \right\rceil + 1$ if and only if for any node v, the following three properties hold:*

1. *v has ring degree at most $\left\lfloor \frac{k+1}{2} \right\rfloor$;*
2. *for any branch B at v, $xs_m(B) \leq k - \left\lceil \frac{m}{2} \right\rceil + 1$;*
3. *for choosing $\frac{k+1}{2} - m + 2$ branches, for any even $i > 1$, at most i branches B at v have $xs_m(B) \geq k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$.*

The fact that the first property is necessary directly follows from the following claim:

Claim 4.3.1. *For any tree of rings TR with maximum ring degree Δ and recontaminating constrain m, $xs_m(T) > 2 \times \Delta - m - 1$.*

Proof. The situation of clearing a tree of rings with maximum ring degree Δ and recontamination constrain m is equivalent to clear a tree of rings with maximum ring degree $\Delta - \left\lceil \frac{m}{2} \right\rceil + 1$ in exclusive search model. From claim 4.2.1, the searchers we need for the equivalent problem will always use searchers more than $2 \times \Delta - m - 1$. Thus claim 4.3.1 is proved. □

The second property is necessary by the next claim.

Claim 4.3.2. *Any tree of rings TR and any subtree TR' of TR, $xs_m(TR') \leq xs_m(TR)$.*

Proof. Contrary to classical graph searching, the proof of this result is not trivial because of the exclusivity property. To prove it, we have to transform an exclusive strategy S for TR into a strategy S' for T' using the same number of searchers, and without violating the exclusivity property. The fact that S may be not monotone (i.e., some recontamination may occur during S) makes the proof technical, because one has to "control" the recontamination of T' in S'.

An easy way to get the idea is like this: for the cases where the exclusive search number for a subgraph is larger than the whole graph is that a single slide move can clean multiple edges, that is, a move in one section can help the cleaning for another section, but for trees, we could never achieve this goal since two sections are connected by a single common vertex. □

The idea to show the third property in the theorem is necessary is to reduce the problem to a smaller scope and use the strategy from what we developed before. Suppose that we are able to clean $\left\lfloor \frac{k+1}{2} \right\rfloor - \left\lceil \frac{m}{2} \right\rceil + 2$ branches from a node v. Then there are at most $\left\lfloor \frac{m}{2} \right\rfloor - 1$ branches left to be cleaned. Considering the re-contamination constrain stated in the definition. Node v will never get recontaminated again. Then the problem becomes easy, simply clean remaining branches one by one and finally finish cleaning the whole tree. So the key point is to clean a sub-graph with $\left\lfloor \frac{k+1}{2} \right\rfloor - \left\lceil \frac{m}{2} \right\rceil + 2$ branches. We have reduced the original problem to a problem that exclusive graph search a sub-graph with $\left\lfloor \frac{k+1}{2} \right\rfloor - \left\lceil \frac{m}{2} \right\rceil + 2$ branches. We prove the following claim to show the third condition

Claim 4.3.3. *Let $k \geq 1$. For any tree of rings TR, if there exists $v \in TR$ and an even integer $i > 1$ such that any branch B at v, $xs_m(B) \leq k - \left\lceil \frac{m}{2} \right\rceil + 1$ and there is a set $B = \{TR_j : xs(TR_j) \geq k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1\}$ of branches at v and $|B| > i + \left\lceil \frac{m}{2} \right\rceil - 1$, then $xs_m(TR) > k - \left\lceil \frac{m}{2} \right\rceil + 1$.*

Proof. Let S be any exclusive strategy that clears TR. For the sake of contradiction, suppose S uses at most $k - \left\lceil \frac{m}{2} \right\rceil + 1$ searchers. Then, at some step, at least $k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$ searchers are in TR_j of B in order to clear the branch according to the definition of set B. Let s_j be the last such step of S that occurs in TR_j . Without loss of generality, assume that $s_i < s_j$ for any $1 < i < j \leq |B|$. Then we may assume before step s_j , TR_j is not completely clear. Then at step $s_{i/2+1}$, at least $k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$ searches are in $TR_{i/2+1}$, some edges have been cleared in TR_j for any $j \leq i/2$ and TR_j cannot become fully contaminated again.(Otherwise there would be another step after s_j that has $k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$ searches in TR_j). Now consider step $s_{i/2+1}$, at least $k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$ searches are in $TR_{i/2+1}$ and there are at most $i - 1$ searches outside $T_{i/2+1}$. What we have here is that there is at least one branch at v in $TR_{i/2+2}, ...TR_{|B|}$ with still contaminated edges, and at least one branch at v in $T_1, ...T_{i/2}$ with (at least) some clear edges that must not be recontaminated and no searchers occupy nodes in both these branches □

Next we prove that the conditions in Theorem 4.3.1 are sufficient.

Claim 4.3.4. *Given a tree of rings TR, if for any node v in TR satisfies*

1. *v has ring degree at most $\left\lfloor \frac{k+1}{2} \right\rfloor$;*
2. *for any branch B at v, $xs_m(B) \leq k - \left\lceil \frac{m}{2} \right\rceil + 1$;*
3. *for choosing $\left\lfloor \frac{k+1}{2} \right\rfloor - m + 2$ branches, for any even $i > 1$, at most i branches B at v have $xs_m(B) \geq k - i - (\left\lceil \frac{m}{2} \right\rceil - 1) + 1$.*

we have a winning extended exclusive search strategy using at most $k - \left\lceil \frac{m}{2} \right\rceil + 1$ searchers.

Proof. The search strategy here is very similar to the strategy we developed for extended exclusive search on trees so it is not stated in detail here. □

By Claims 4.3.1-4.3.4, Theorem 4.3.1 is proved. Similarly, the property check can be done in $O(n)$.

4.4 Classic Graph Search on Tree of Rings

In this section we present the classic graph search results on tree of rings. These results can be trivially obtained from the previous results of the classical search models for trees but to our best knowledge, they are not reported in the literature.

Theorem 4.4.1. *For any tree of rings TR and integer $k \geq 1$, $es(T) \geq k+1$ if and only if T has a vertex v at which there are three or more branches that have search number k or more.*

Proof. The proof idea of the theorem is very similar to the proof of Parson's lemma[35] since we are also utilizing the tree structure to figure out the search number. □

Theorem 4.4.2. *For any tree of rings TR and integer $k \geq 1$, $ms(T) \geq k+1$ if and only if T has a vertex v at which there are three or more branches that have search number k or more.*

The strategy for mixed search is similar to edge search since they both allow a searcher to slide along an edge in the graph.

Theorem 4.4.3. *For any tree of rings TR, $ns(TR) \geq k+1$ for $k \geq 2$ if and only if there exists a vertex $t \in V(T)$ with at least three branches $TR_u, TR_v, andTR_w$ such that $ns(TR_u) \geq k, ns(TR_v) \geq k$, and $ns(TR_w) \geq k$. For any tree T, $ns(T) \geq 2$ if and only if there exists a vertex $t \in V(TR)$ with at least one branch.*

Proof. The proof idea of the theorem is very similar to the proof of Scheffler's lemma[34] since we are also utilizing the tree structure to figure out the search number. □

Chapter 5

Graph Search for Power Law Graphs

5.1 Power Law Graph Basics

Complex networks are common in many high technological areas. For example, the Internet is a complex network of routers and computers linked by various physical or wireless links. For social network, the nodes are human beings and edges represent various social relationships. the World Wide Web is an virtual network of Web pages connected by hyperlinks. These systems are just a few of the many examples that draws the attention of the scientific community to investigate.

It was experimentally observed that the majority of these networks are scale-free and follow power law degree distribution. The following three concepts are widely accepted to be the characteristic of these networks.

Small worlds[2]: The small-world concept in simple terms describes the fact that despite their often large size, in most networks there is a relatively short path between any two nodes. The distance between two nodes is defined as the number of edges in the shortest path connecting them. The most popular manifestation of small worlds is the "six degrees of separation" concept, uncovered by the social psychologist Stanley Milgram (1967), who concluded that there was a path of acquaintances with a typical length of about six between most pairs of people in the United States (Kochen, 1989). The small-world property appears to characterize most complex networks.

Clustering [2]: A common property of social networks is the cliques form, representing circles of friends or acquaintances in which every member knows every other member. Suppose we have a selected node i in the network, having k_i edges which connect it to k_i other nodes. If the nearest neighbors of the original node were part of a clique, there would be $k_i(k_i-1)/2$ edges between them. The ratio between the number of edges E_i that actually exist between these k_i nodes and the total number $k_i(k_i-1)/2$ gives the value of the clustering coefficient of node i

$$C_i = \frac{2E_i}{k_i(k_i-1)}$$

The clustering coefficient of the whole network is the average of all individual C_i.

Degree distribution [2]: Not all nodes in a network have the same number of edges (same node degree). The spread in the node degrees is characterized by a distribution function $P(k)$, which gives the probability that a randomly selected node has exactly k edges. Power law graph has an interesting function $P(k)$.

There are multiple definitions of power law networks. Some of them state that in a power law network the number of vertices of degree k is proportional to $k^{-\alpha}$ for some parameter α [38]. In other cases power law is defined with respect to random graphs and only talks about expected degrees of vertices[4, 17]. Both these approaches may not be applied to the analysis of algorithms running on real-world networks. The first one suffers from two serious drawbacks. First, it is often not stated in a formal way. Second, it seems that it effectively disallows even a single vertex with high degree. On the other hand the stochastic definition can only be applied to graphs randomly drawn from some distribution. This is not the case for real-world graphs, which are fixed.

Most previous research papers use the following definition. A power law graph is a network whose vertex degree distribution follows a power law. That is, the fraction $P(k)$ of nodes in the network having k connections to other nodes goes for large values of k as

$P(k)$ directly proportional to $k^{-\gamma}$

where γ is a parameter whose value is typically in the range $2 < \gamma < 3$.
Formally, a power law graph is defined as the follows

Definition 5.1 Define d_i to be the number of vertices with degree i, the power law graph has the following property.

$$d_i = \alpha i^{-\gamma}$$

Where α is a constant, $2 < \gamma < 3$

Pawl et. al. [33] pointed out possible drawbacks of the above power law graph definition above. They define a deterministic condition for checking whether a graph has a power law degree distribution and show that many real-world networks satisfy it. Graphs satisfying their definition are called power law bounded networks (PLB). It captures the power law behavior of degree distribution that is necessary for the theoretical analysis of algorithms. At the same time it is flexible enough to cover many real-world graphs. The main difference between Definition 5.1 and Definition 5.2 is that Definition 5.2 does not impose any lower bounds on the numbers of vertices of given degrees.

Definition 5.2 [33] Let G be an undirected n-vertex graph and $c_1 > 0$ be a universal constant. We say that G is power law bounded (PLB) for some parameters $1 < \alpha = O(1)$ and $t \geq 0$ if for every integer $k \geq 0$, the number of vertices v, such that $deg_(v) \in [2d, 2d+1)$ is at most

$$c_1 n(t+1)^{\alpha-1} \sum_{i=2^d}^{2^{d+1}-1} (i+t)^{-\alpha}$$

The $(t+1)^{\alpha-1}$ factor in the above definition is necessary to ensure that the sum of the above upper bounds over all k is $O(n)$. The above power law distribution that includes the shift by the parameter t is called shifted power law and was observed in different real-world networks. In particular, the parameter t allows us to better fit the degree distributions in experiments. As experiments show, the value of t is very small. However, in general it is unknown whether and how t depends on other parameters of the network and we are not aware of the models that would describe such dependence. A reasonable assumption here seems to be that $t = O(n^\epsilon)$ for every $\epsilon > 0$.

The rest of this chapter is organized as follows: We first show how to find a cycle base in a power law graph. Then we point out that the classical power law graph definition (Definition 5.1) may have a draw back that does not specify the number of edges in realistic power law graphs correctly and give some observations that Definition 5.2 may address this problem.

Finally, we give heuristic algorithms for the classical search models on power law graphs.

5.2 Cycles in Power Law Graph

Throughout this section we use n to represent the number of vertices in a graph, m to be the number of edges, d_k to be the number of vertices of degree k, and $d_{\geq k}$ to be the number of vertices of degree at least k.

A node-to-node adjacency structure is an $n \times n$ matrix such that entry $a_{ij} = 1$ if node i is adjacent to node j and 0 otherwise. A node-to-edge adjacency structure has a list for each node v that contains all nodes adjacent to v.

As shown in the sections before, we already have a well-known algorithm for graph search on tree structure. The main obstacle which prevent us from using the existing technique is that for a power law graph we can have cycles.

Depth-first search (DFS) [8] can be used to find cycles in a graph. It is usually assumed that the graph data structure is of the type node-edge adjacency instead of node-node adjacency. In the node-edge adjacency structure, the nodes are numbered from 1 to n. The node-i record lists the nodes adjacent to node i (connected to node i by an edge).

> If one or more nodes of the node-i record were not yet visited from i, let node j be the first node not visited. If node j was already visited by DFS (obviously from node other than node i), mark edge (i, j) as back edge otherwise mark edge (i, j) as tree edge and set i as parent of j. Mark node j as visited in the node-i record. Set node j as current node.
> If all vertices of vertex-i are yet visited, set the parent vertex of vertex i as current.

The algorithm stops when we are back to the first visited node called source. If all nodes were visited, then the graph is connected. Each back edge (i, j) defines a cycle. A cycle consists of the back edge (i, j) and unique tree edges forming the path from j to i. The cycles so defined by the back edges form a cycle base of the graph. Every cycle of the graph is the union (exclusive OR) of two or more cycles from this cycle base(see below). Of course, the number of cycles in a graph can be exponential in the number of nodes of the graph.

Definition 5.3 A cycle base is a set of $m-(n-1)$ cycles that are independent in the sense that we cannot reconstruct one cycle from the set by the union (defined below) of two or more other cycles of the set.

The set of $m - (n - 1)$ back edges defines a cycle base. This is not the only cycle base. Actually there can be an exponential number of them.

Given a connected power law graph, perform DFS on the graph, the number of back edges is always $m-(n-1)$. This can be easily verified. We know the maximum number of edges in a tree is $(n-1)$, and in the DFS algorithm above the tree edges form a spanning tree since we went over all the vertices in the graph. The tree edge marked by go through DFS process induce a connected sub-graph that contains every vertex of the graph and no cycles.

We need searchers to guard the corresponding cycles refer to these back edges. Thus we need $m-(n-1)$ guard searchers, then the graph becomes a tree and we could apply previous graph search results on the spanning tree.

5.3 Problem with Classic Power Law Graph Definition

Lemma 5.3.1. *Let G be a graph and $k \geq 0$. The number of edges of G is $\frac{1}{2}\sum_{i=1}^{d_{max}} d_i \times i$*

Proof. A vertex of degree k is counted k times in the summation. Thus the sum is equal to the total degree of all vertices, which is twice the number of edges. □

Let B be the set of back edges and d_{max} be the largest degree of a given power law graph. Then,

$$|B| = m-(n-1) = \frac{1}{2}\sum_{i=1}^{d_{max}} d_i \times i - \sum_{i=1}^{d_{max}} d_i + 1 \tag{5.1}$$

Definition 5.4 The Riemann zeta function or Euler–Riemann zeta function, $\zeta(s)$, is a function of a complex variable s that analytically continues the sum of the infinite series$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$ for when the real part of s is greater than 1.

It has been proved that ζ function convergence when $s >= 2$ and

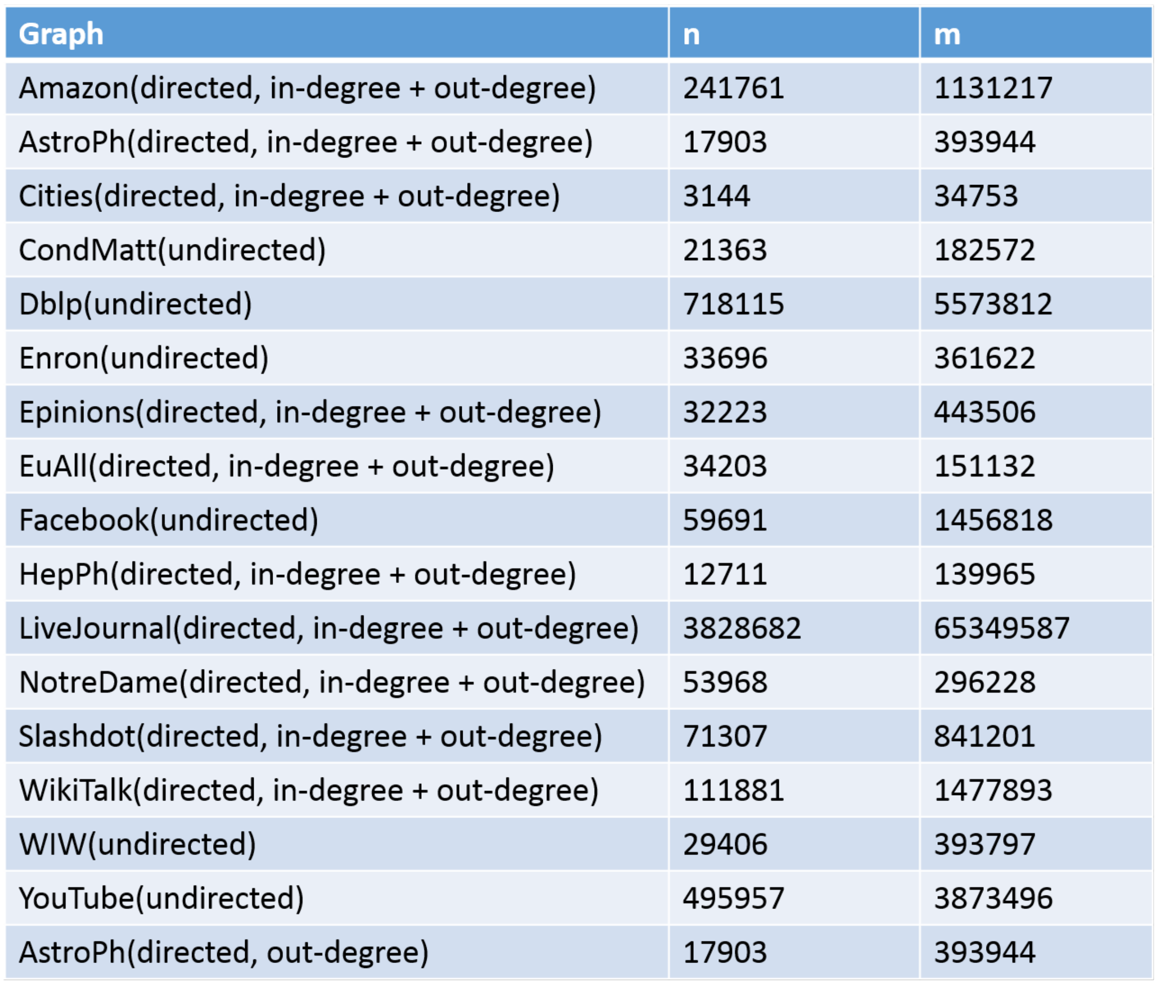
$$\zeta(1) = 1 + \frac{1}{2} + \frac{1}{3} + ... = \infty \tag{5.2}$$

$$\zeta(1.5) = 1 + \frac{1}{2^{1.5}} + \frac{1}{3^{1.5}} + ... = 2.612 \tag{5.3}$$

$$\zeta(2) = 1 + \frac{1}{2^2} + \frac{1}{3^2} + ... = \frac{\pi^2}{6} \tag{5.4}$$

$$\zeta(3) = 1 + \frac{1}{2^3} + \frac{1}{3^3} + ... = 1.202 \tag{5.5}$$

Figure 5.1: Numbers of Vertices and Edges of Real Graphs Reported in [33]



Graph	n	m
Amazon(directed, in-degree + out-degree)	241761	1131217
AstroPh(directed, in-degree + out-degree)	17903	393944
Cities(directed, in-degree + out-degree)	3144	34753
CondMatt(undirected)	21363	182572
Dblp(undirected)	718115	5573812
Enron(undirected)	33696	361622
Epinions(directed, in-degree + out-degree)	32223	443506
EuAll(directed, in-degree + out-degree)	34203	151132
Facebook(undirected)	59691	1456818
HepPh(directed, in-degree + out-degree)	12711	139965
LiveJournal(directed, in-degree + out-degree)	3828682	65349587
NotreDame(directed, in-degree + out-degree)	53968	296228
Slashdot(directed, in-degree + out-degree)	71307	841201
WikiTalk(directed, in-degree + out-degree)	111881	1477893
WIW(undirected)	29406	393797
YouTube(undirected)	495957	3873496
AstroPh(directed, out-degree)	17903	393944

By combining (5.3) with (5.5) together with the classic definition of power law graph, when γ is 2.5, the number of back edges is less than $\frac{1}{10}$ of n. However, the following graph shows that in a realistic power law graph, the number of back edges is usually 5 to 10 times the number of vertices(see Figure 5.1).

The PLB definition, instead of specifying detailed vertex distribution for each degree, defines the total number of vertices for each exponentially increased range to be the total number of vertices if the vertices within the range follows power law distribution. When the parameter d is small, we see the restriction on distribution is just like the original power law definition distribution. As the parameter d increases, this restriction is relaxed thus it allows a portion of low degree vertexes within a range to be shifted to high degree within the same range. The main reason why the original power law graph definition fails to appeal to the realistic network is that the graph does not have enough edges. By adjusting the vertex distribution, we allow the number of high degree vertices to be larger, thus makes the graph have more edges.

Observed from the communication network, in a realistic undirected power law graph, the number of edges is usually five to ten times the number of edges in the graph. This actually proves our thinking that the original power law definition is short of edges.

Recall definition 5.1, $d_i = \alpha i^{-\gamma}$. Assume we only have one vertex of degree d_{max} in G, then

$$1 = \alpha d_{max}^{-\gamma} \tag{5.6}$$

Using the property of Riemann zeta function, $\alpha > \frac{n}{2}$, thus

$$1 > \frac{n}{2} d_{max}^{-\gamma} \tag{5.7}$$

$$d_{max} > \sqrt[\gamma]{\frac{n}{2}} \tag{5.8}$$

which means given a power law graph G of n vertices, there exists a vertex of degree $\sqrt[\gamma]{\frac{n}{2}}$ with high probability

5.4 Heuristic Graph Search Algorithms on Power Law Graphs

We first consider the mixed search on power law graph. As we have seen from the previous section, the number of back edges is usually five to ten times the number of vertices. This means for high degree vertex, usually it is associate with several back edges. Thus instead of guarding the back edges, we now seek to guard high degree vertices.

Our algorithm starts with guarding all the vertices with degree at least four. For the rest of the graph we consider the number of searchers and the number of guardians separately.

We first consider the number of guardians for degree one, two and tree vertices separately. Note that the number of back edges for vertex v is at most the degree of v minus one.

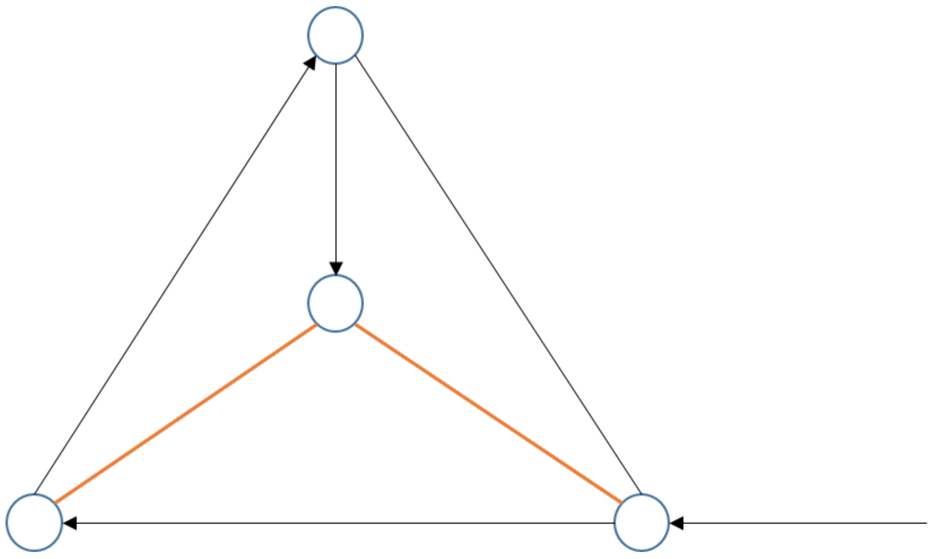
Degree one vertices are the leaf vertices of the graph thus we do not need searchers to guard these vertices.

For the vertices of degree two, we can always choose to form the DFS tree starting from a vertex with degree at least three. Thus all vertices with degree two will never have back edges.(The only situation where a degree two vertex can have a back edge is that we start from a degree two vertex and follows DFS and return to itself, which means that we start out from a degree two vertex, following a cycle when performing our DFS and return to the starting vertex itself.). Thus we don't need to guard degree two vertices.

For the vertices of degree tree, we have three possibilities. If all the three edges incident to the vertex are not back edges, that is, for vertex v we have zero back edges, we do not need to guard it.

For degree three vertices with two back edges, it has to be leaf in the DFS tree, since we already put guard vertices on all the vertices with degree at least four, the only situation we need to consider is shown in figure 5.2 where the back edges are in red.

Figure 5.2: Degree three vertex with two back edges



Hence one guardian searcher is enough to guard the all cycles in this subgraph. Thus in this situation, 1/3 of d_3 is enough for the guardians

For degree three vertices with one back edges, we do the following. We trace staring from the back edge and follow the back tracking path until we reach another vertex of degree at least three with back edges. If we met a vertex with degree at least four, we know we already have a guardian on the vertex and thus break this cycle. If we met a vertex with degree three, we check the number of back edges associate with it. If the vertex has two back edges, we know we are considering situation two.

Otherwise we reach a vertex of degree three with one back edge. Then we choose to guard either of the two vertices to break the cycles. Thus in situation three, 1/2 of d_3 is enough for the guardians.

Sum all the situations up, we see for the remaining graph with vertices of degree at most three, we need at most 1/2 of d_3 for the guardians. Now we consider the number of searchers to actually search the graph.

Recall Property 2.2.1, the searchers we need to clean the subgraph consist of vertices with degree at most 3 is at most $1 + \log_3(d_3 - 1)$. Thus

Theorem 5.4.1. *Mixed search number for any power law graph is at most* $d_{\geq 4} + \frac{1}{2}d_3 + \log_3(d_3 - 1) + 1$.

The following two theorems can be easily derived from Theorem 5.2 together with Theorem 1.0.1.

Theorem 5.4.2. *Edge search number for power law graph is at most* $d_{\geq 4} + \frac{1}{2}d_3 + \log_3(d_3 - 1) + 2$.

Theorem 5.4.3. *Node search number for power law graph is at most* $d_{\geq 4} + \frac{1}{2}d_3 + \log_3(d_3 - 1) + 2$.

The exclusive search number can be derived from Property 2.3.2 and Theorem 5.4.1

Theorem 5.4.4. *Exclusive search number for power law graph is at most* $2(\log_3(d_3 - 1) + 1) + \frac{1}{2}d_3 + d_{\geq 4}$

Now let us consider extended exclusive search on power law graph. Let v be a vertex of degree d, then after cleaning $d - m + 1$ branches, there are only $m - 1$ dirty edges connecting to v which would never cause recontamination. Thus the strategy to clean such vertex is similar to the exclusive search stragety to clean a vertex of degree $d - m + 1$. Hence we have the following:

Theorem 5.4.5. *Extended exclusive search number for power law graph is at most* $2(\log_3(d_{3+(m-1)} - 1) + 1) + \frac{1}{2}d_{3+(m-1)} + d_{\geq 4+(m-1)}$

The number of searchers used by our heuristic algorithms are upper bounded by $O(d_3) = O(n)$ for all search models discussed. For power law graphs with $\Omega(n)$ maximum node degree, the numbers of searchers used by our algorithms are within a constant factor from the search number of the graphs.

A remark here is that our algorithm can be applied to general graphs, and would have a good performance on graphs with a small number of vertices with degree at least three.

Chapter 6

Conclusion

In this thesis, we first reviewed existing graph search models and show their differences and connections. Based on review, we proposed a new graph search model called extended exclusive search. Then we show graph search results on an interesting type of graph called tree of rings. Finally, we proposed heuristic search algorithms for power law graphs based on the known and new proposed search models.

In the area of power law graph searching, there are still many interesting problems. Based on the content of this thesis and other works we have done, we list some problems here and provide some suggestions for the future work.

1. Develop a new power law graph definition which gives a better description of real world power law graphs. We have pointed out that the classic definition for power law graph does not provide enough high degree vertices. A simple shift parameter introduced in a modified definition for power law graphs may not be enough to address this problem.

2. Our heuristic mixed search strategy on power law graphs consider vertices of degree at least 4 to be high degree vertices. If we consider the vertices of degree at least 5 to be high degree vertices, whether we could have search heuristics using a smaller number of searchers.

3. It would be interesting to analyze the theoretic performances of our heuristic algorithms.

Bibliography

[1] Narrowness A. Kornai, Z. Tuza. pathwidth, and their application in natural language processing. *Discrete Appl. Math*, 36:87–92, 1992.

[2] Barabási A. L Albert, R. Statistical mechanics of complex networks. *Reviews of modern physics*, 2002.

[3] I.H. Sudborough B. Monien. Min cut is np-complete for edge weighted trees. *Theoret. Comput. Sci.*, 58:209–229, 1988.

[4] D. Bienstock. Graph searching, path-width, tree-width and related problems. *Reliability of Computer and Communication Networks, DIMACS series in Disc. Math. and Theoretical Comp. Scie.*, 5:33–49, 1991.

[5] Burman J. Nisse N. Blin, L. Exclusive graph searching. *Lecture Notes in Computer Science*, page 181–192, 2013.

[6] R. L. Breisch. An intuitive approach to speleotopology. *Southwestern Cavers*, 6:72–78, 1967.

[7] R. L. Breisch. Lost in a cave-applying graph theory to cave exploration. 2012.

[8] T. H. Cormen. *Introduction to algorithms. MIT press.* 2009.

[9] P. Seymour D. Bienstock. Monotonicity in graph searching. *J. Algorithms*, 12:239–245, 1991.

[10] P. Seymour D. Bienstock. Monotonicity in graph searching. *J. Algorithms*, 12:239–245, 1991.

[11] T. Erlebach. Approximation algorithms and complexity results for path problems in trees of rings. *the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS2001)*, LNCS2136:351–362, 2001.

[12] I.H. Sudborough F. Makedon. On minimizing width in linear layouts. *Disc. Appl. Math.*, 23:243–265, 1989.

[13] D. Thilikos F.V. Fomin. On the monotonicity of games generated by symmetric submodular functions. *Discret. Appl. Math.*, 131:323–335, 2003.

[14] J. Gustedt. On the pathwidth of chordal graphs. *Discrete Appl. Math*, 45:233–248, 1993.

[15] D. Kratsch H.L. Bodlaender, T. Kloks. Treewidth and pathwidth of permutation graphs. *SIAM J. Discrete Math*, 8:606–616, 1995.

[16] D. Kratsch H. Muller H.L. Bodlaender, T. Kloks. Treewidth and minimum fill-in on d-trapezoid graphs. *Technical Report UU-CS-1995-34, Department of Computer Science, Utrecht University, Utrecht, the Netherlands*, 1995.

[17] D. Kratsch H. Muller H.L. Bodlaender, T. Kloks. Treewidth and minimum
ll-in on d-trapezoid graphs. 1995.

[18] R.H. M ohring H.L. Bodlaender. The pathwidth and treewidth of cographs. *SIAM J. Discrete Math*, 6:181–188, 1993.

[19] T. Kloks H.L. Bodlaender. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

[20] J.S. Turner J.A. Ellis, I.H. Sudborough. The vertex separation and search number of a graph. *Inform.Comput*, 113:50–79, 1994.

[21] N.G. Kinnersley. The vertex separation number of a graph equals its path-width. *Inform. Process. Lett.*, 42:345–350, 1992.

[22] T. Kloks. Treewidth — computations and applications. *Lecture Notes in Computer Sciense*, 842, 1994.

[23] A.S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40:224–245, 1993.

[24] C.H. Papadimitriou L.M. Kirousis. Searching and pebbling. *Theor. Comput. Sci.*, 47:205–216, 1986.

[25] C.H. Papadimitriou L.M. Kirousis. Searching and pebbling. *Theor. Comput. Sci.*, 47:205–216, 1986.

[26] C.H. Papadimitriou L.M. Kirousis. Interval graph and searching. *Discrete Math*, 55:181–184, 55.

[27] R.H. Mohring. Graph problems related to gate matrix layout and pla folding. *G. Tinnhofer et al. (Eds.), Computational Graph Theory*, page 17–32, 1990.

[28] M. Garey D. Johnson C.H. Papadimitriou N. Megiddo, S. L. Hakimi. The complexity of searching a graph. *J. ACM*, 35:18–44, 1988.

[29] M.R. Garey D.S. Johnson C.H. Papadimitriou N. Megiddo, S.L. Hakimi. The complexity of searching a graph. *J. Assoc. Comput. Mach.*, 35:18–44, 1988.

[30] P. Seymour N. Robertson. Graph minors i: excluding a forest. *J. Comb. Theory Ser. B*, 35:39–61, 1983.

[31] P. Seymour N. Robertson. Graph minors ii: algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[32] P.D. Seymour N. Robertson. Graph minors i. excluding a forest, j. combin. page 39–61, 1983.

[33] J. Lacki P. Brach, M. Cygan and P. Sankowski. Algorithmic complexity of power law networks. *In R. Krauthgamer, editor, Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA*, page 1306–1325, 2016.

[34] N.N. Petrov P.A. Golovach. The search number of a complete graph. *Vestn. Leningr. Univ., Math*, 19:15–19, 1986.

[35] T. D. Parsons. The search number of a connected graph. *9th Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Congress*, page 549–554, 1978.

[36] B. Reed. Treewidth and tangles: a new connectivity measure and some applications. *Surveys in Combinatorics, ed. by R.A. Bailey*, page 87–162, 1997.

[37] A. Proskurowski S. Arnborg, D.G. Corneil. Complexity of finding embeddings in a k-tree. *SIAM J. Algebra Discrete Meth*, 8:277–284, 1987.

[38] A. Proskurowski S. Arnborg, D.G. Corneil. Complexity of finding embeddings in a k-tree. *SIAM J. Algebra Discrete Meth.*, 8:277–284, 1987.

[39] P. Scheffler. A linear algorithm for the pathwidth of trees. *R. Bodendiek, R. Henn (Eds.), Topics in Combinatorics and Graph Theory, Physica-Verlag, Heidelberg*, page 613–620, 1990.

[40] P. Scheffler. Optimal embedding of a tree into an interval graph in linear time. *Fourth Czechoslovakian Symposium on Combinatorics, Graphs and Complexity*, page 287–291, 1992.

[41] C.W. Ho T.-s. Hsu C.Y. Tang S.L. Peng, M.T. Ko. Graph searching on chordal graphs. *Lecture Notes in Computer Science*, 1178:156–165, 1996.

[42] H. Muller D. Kratsch T. Kloks, H. Bodlaender. Computing treewidth and minimum fill-in, all you need are the minimal separators. *ESA'93, Lecture Notes in Computer Sciense*, 726:260–271, 1993.

[43] Q. Gu Z. Bian and X. Zhou. Efficient algorithms for wavelength assignment on trees of rings. *Discrete Applied Mathematics*, 2008.