Average Sensitivity of Breadth-First Search Algorithm on Grids

by

Mohamad Assari

B.Sc. in Computer Engineering, Amirkabir University of Technology, 2020
B.Sc. in Physics, Amirkabir University of Technology, 2019

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

in the
Department of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Mohamad Assari**
Degree:	**Master of Science**
Thesis title:	**Average Sensitivity of Breadth-First Search Algorithm on Grids**
Committee:	**Chair:** Saba Alimadadi Assistant Professor, Computing Science
	Qianping Gu Supervisor Professor, Computing Science
	Igor Shinkar Committee Member Assistant Professor, Computing Science
	Matt Amy Examiner Assistant Professor, Computing Science

Abstract

The average sensitivity of a graph algorithm is a measure that calculates the stability of the algorithm when the input graph undergoes perturbations. The perturbation is represented by removing some of the edges of the input graph, and the stability is quantified using the Earth mover's distance between the algorithm's outputs on the original and perturbed graphs. In this thesis, we investigate the average sensitivity of the breadth-first search(BFS) algorithm with BFS trees as outputs. It is known that for an arbitrary graph $G(V, E)$, the average sensitivity of the BFS algorithm is $\Theta(|V(G)|)$. We prove that when the input graph G is an $m \times n$ grid graph, there is a BFS algorithm with an average sensitivity of at most 2. We also prove that for specific starting nodes, the randomization of the BFS algorithm reduces its average sensitivity.

Keywords: Average sensitivity; graph algorithms; breadth-first search algorithm; grids

Dedication

This thesis is dedicated to my beloved parents, whose steadfast support and sacrifices have been the foundation of my achievements. Your constant encouragement and the loving environment you provided have allowed me to pursue and realize my dreams. The distance between us has been a heavy burden, yet your strength and resilience have always inspired me.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Qianping Gu. Your consistent support, insightful suggestions, and valuable recommendations have been instrumental in shaping this thesis. Our almost weekly meetings over the past three years have been immensely beneficial, and I am truly grateful for the freedom you provided me to work at my own pace.

I would also like to extend my sincere thanks to my committee member, Professor Igor Shinkar, for taking the time to read and evaluate my thesis. And a special thanks to Professor Matt Amy for agreeing to be my examiner on such short notice and for thoroughly reviewing my work. Your willingness to step in and provide your expertise is greatly appreciated. I am also profoundly grateful to the chair of my thesis, Professor Saba Alimadadi, for her time and willingness to chair my thesis defence.

Finally, I want to thank my friends and family for their support and encouragement throughout this journey. Your belief in me has been a constant source of motivation and strength.

Thank you all for your contributions to this significant milestone in my academic journey.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The average sensitivity of a graph algorithm is a measure that calculates how stable the output of the algorithm is when there is a perturbation in the input graph. The perturbation is represented by removing some of the edges of the input graph, and then the stability of the algorithm can be measured by calculating how close the outputs of the algorithm are when the original and the perturbed graphs are given as inputs. The average sensitivity has been defined in different areas, and a dedicated study on the average sensitivity of graph algorithms appears in the reference [1].

The average sensitivity of graph algorithms can be calculated for algorithms that produce a set of edges(or nodes) as an output. Therefore, the distance between the two outputs is just the distance between the two sets. For deterministic algorithms, this distance is calculated by the Hamming distance, and for randomized algorithms, this distance is the Earth mover's distance(the concept of Earth mover's distance was introduced by Rubner et al.[2]). And since removing different edges of the input graph might create different distances, to calculate the average sensitivity of that algorithm, we need to take the average over the removed edges.

An algorithm is considered stable on average if its average sensitivity is small. This means that in the absence of the original graph, if we only have access to one of its subgraphs, then the algorithm produces an output close(with a low distance) to the output of the algorithm on the original graph. The average sensitivity analysis of algorithms is important because in many applications the input graph is dynamic, therefore it changes all the time, so it is impossible or maybe expensive to keep track of the input graph at all times. However, if the algorithm that we want to use has a low average sensitivity or equivalently is stable on average, it means that the output of the algorithm almost stays the same, when there is a change in the input graph.

Breadth-first search(BFS) is a fundamental graph search algorithm. A key output of the

BFS algorithm is the BFS tree which has numerous applications. The concept of average sensitivity in the context of graph algorithms, such as BFS, relates to understanding how small changes in the input(for example the grid structure) can affect the output of the algorithm. In practical applications involving BFS on grids, the average sensitivity can provide valuable insights into the robustness and reliability of the solutions generated by BFS. It is proved in [1] that for arbitrary graph $G(V, E)$, the average sensitivity of the single source shortest paths algorithms is $\Theta(|V(G)|)$, implying that the average sensitivity of the BFS algorithm with BFS trees as outputs is $\Theta(|V(G)|)$. The average sensitivity of the BFS algorithm on special families of input graphs is yet to be studied.

Grid(lattice) graphs are an important family of graphs that are commonly used to model many networks and structures like roads, city blocks, traffic management etc. The BFS algorithm on grid graphs has many important applications, especially a BFS algorithm of small average sensitivity on the output BFS trees is desirable in practice. In this thesis, we investigate the average sensitivity of the BFS algorithm on $m \times n$ grid graphs. We prove that there is an implementation of the deterministic BFS algorithm that achieves an average sensitivity of at most 2 and a randomized BFS algorithm that has an average sensitivity of O(1) when$m = \Theta(n)$, a significant reduction from $O(|V(G)|)$ for arbitrary graphs.

1.1 Related Work

In computer science, average sensitivity is a broad concept and has been used in various areas. In a high-level view, it tries to measure how the output of a function or an algorithm is dependent on the input. As an example, many papers([3], [4], [5], [6], [7]) have used the concept of average sensitivity for Boolean functions where it quantifies the average number of input bits that need to be flipped to change the output of the function. This concept is important in complexity theory, learning theory, and the study of noise sensitivity.

The concept of average sensitivity is also used in data mining where they study [8] how sensitive the output of a clustering algorithm is to missing edges in the input graph which could be caused by several reasons including measurement error, not having access to some portions of the input graph, or mistakes in data conversion.

However, the notion of average sensitivity of graph algorithms was first introduced in 2021 by Varma and Yoshida[1] to examine the stability of an algorithm against input change. The term average sensitivity in [1] is defined for the perturbation with only one edge being removed(k-sensitivity is used for a perturbation with k edges removed). They used the concept of Earth mover's distance which was set forth by Rubner et al. [2] in the definition of average sensitivity of randomized graph algorithms. They also examined the average sensitivity of some famous problems like the single source shortest paths, the minimum spanning forest, global minimum cut, minimum s-t cut, maximum matching, minimum vertex cover, and 2-colouring problems in the same paper.

Kumabe and Yoshida [9] analyzed Lipschitz continuity of algorithms as a stability measure for weighted graphs and later on in another paper[10], discussed the Lipschitz continuous algorithms for covering problems.

Varma and Yoshida[1] showed that there is a connection between the concept of average sensitivity and the area of local computation algorithms which was introduced by Rubinfield et al.[11]. Using the theorems that were also proved in that paper, Varma and Yoshida answered an open problem regarding the query complexity of every local computation algorithm for the 2-coloring problem which was raised earlier by Czumaj et al. [12].

The idea of average sensitivity was also examined for problems that can be solved using a dynamic programming algorithm by Kumabe and Yoshida[13]. Kumabe and Yoshida provided a stable on-average algorithm for the maximum weight chain problem in a transitive graph and then reduced various dynamic programming problems to the MWC(Maximum Weight Chain) problem such as the longest increasing subsequence problem, the interval scheduling problem, the longest common subsequence problem, the longest palindromic subsequence problem, the knapsack problem with integral weight, and the RNA folding problem.

Yoshida and Ito[14] addressed the stability of Euclidean (k, ℓ)-clustering, which aims to find k centers minimizing the sum of the ℓ-th powers of Euclidean distances from each point to its nearest center. They focused on the average sensitivity of this clustering, measuring the output's stability against the removal of random input points. Their study shows that popular algorithms like K-MEANS++ and D_ℓ-SAMPLING exhibit low average sensitivity. Additionally, they demonstrated that any approximation algorithm for Euclidean (k, ℓ)-clustering can be adapted to maintain low average sensitivity without significantly compromising the approximation guarantee. As a result, they provided several algorithms for consistent and dynamic (k, ℓ)-clustering in the random-order model, aiming to maintain a good solution with minimal changes or update time.

Hara and Yoshida[15] developed decision tree learning algorithms that are stable against minor perturbations in training data by adopting the concept of average sensitivity as a stability measure. They addressed the issue of drastic changes in the learned tree structure due to data variations, which can undermine the reliability of extracted knowledge in data mining. Their algorithm achieves low average sensitivity while maintaining accuracy close to that of the optimal decision tree. Experimental results on real-world datasets show that the proposed algorithm allows users to select decision trees based on a trade-off between average sensitivity and accuracy.

Yoshida and Zhou[16] analyzed the sensitivity of algorithms for the maximum matching problem against edge and vertex modifications, emphasizing the robustness of such algorithms to edge failures or attacks. They introduced a randomized $(1-\epsilon)$-approximation algorithm with worst-case sensitivity $O_\epsilon(1)$, significantly improving upon previous algorithms with higher sensitivity. They also presented a deterministic $1/2$-approximation algorithm with sensitivity $\exp(O(\log^* n))$ for bounded-degree graphs and proved that any deterministic constant-factor approximation algorithm must have sensitivity $\Omega(\log^* n)$. Their results demonstrate that randomized algorithms can achieve sensitivity independent of n, whereas deterministic algorithms cannot. Additionally, they extended their analysis to vertex sensitivity and provided an algorithm for online maximum matching with $O_\epsilon(n)$ total replacements in the vertex-arrival model. They also introduced the concept of normalized weighted sensitivity, offering a $1/4\alpha$-approximation to the maximum weighted matching with $O(m \log_\alpha n)$ time complexity and normalized weighted sensitivity $O(1)$.

Kumabe and Yoshida[17] in another paper investigated the average sensitivity of algorithms for the knapsack problem, a key resource allocation issue where stability against input perturbation is essential to avoid costly and untrustworthy reallocations. They extended the concept of average sensitivity, defined by Varma and Yoshida[1], to measure the stability of an algorithm's output when a single item is deleted. Their work presents a $(1-\epsilon)$-approximation algorithm for the knapsack problem with average sensitivity $O(\epsilon^{-1}\log\epsilon^{-1})$. They also proved that any $(1-\epsilon)$-approximation algorithm must have an average sensitivity of $\Omega(\epsilon^{-1})$. As an application, they explored the incremental knapsack problem in a random-order setting, demonstrating the existence of a $(1-\epsilon)$-approximation algorithm with amortized recourse $O(\epsilon^{-1}\log\epsilon^{-1})$ and amortized update time $O(\log n + f_\epsilon)$, where n is the total number of items and f_ϵ is a value dependent on ϵ.

Our line of work is very similar to these papers particularly [1], and [13], because we apply the concept of average sensitivity to the BFS algorithm. However, there are other approaches where some have tried to create or analyze stable algorithms:

In 2001, Andrew Y. Ng et al.[18] devised two PageRank algorithms that are expected to produce stable rankings when there is a small perturbation to the linkage pattern. Also in another paper[19], they analyzed when the Kleinberg HITS and Google PageRank algorithms are expected to give stable rankings. Similar works on stable algorithms have been done in [20], [21], [22], [23], [24] and [25].

These methods are specific to certain algorithms and cannot be generalized to other graph algorithms. However, average sensitivity is a generalized method that can calculate whether a given graph algorithm is stable or not.

1.2 Applications of Average Sensitivity of BFS

The concept of average sensitivity was introduced only recently [1], and like many theoretical advancements, it will find its place and applications in due time. Given the fundamental nature of BFS as a graph algorithm and the characteristics of average sensitivity, we anticipate that the average sensitivity of the BFS algorithm, particularly on grids, will eventually be utilized in the following areas:

1.2.1 Robotics and Path Planning

Robust Navigation: In autonomous robotics, understanding the average sensitivity of BFS can help in designing more robust navigation algorithms. If a robot's path is highly sensitive to changes in the grid(e.g., newly discovered obstacles), planners can adjust the algorithm to be more resilient to such changes, ensuring reliable operation in dynamic environments. The BFS algorithm is used extensively in robotics and path planning as in [26], [27], [28], [29] and [30].

In particular, Navya and Ranjith [30] analyzed the application of medical robotics, specifically in the context of the COVID-19 pandemic, where service robots assist doctors by offering medication and monitoring patients. It compares two grid-based algorithms, BFS(Breadth-First Search) and DFS(Depth-First Search), to determine the optimal path-finding algorithm for medical robots in a virtual 20×20 grid plan of a hospital ward. Their MATLAB simulations reveal that BFS is more effective in finding an optimal path for the service robots.

1.2.2 Disaster Management and Evacuation Planning

Evacuation Routes: In disaster scenarios(e.g. floods or fires), knowing how sensitive BFS is to changes in the environment can aid in planning evacuation routes. If the average sensitivity is high, it indicates that even small changes in the environment can significantly alter evacuation routes, necessitating more flexible and adaptive planning. Several papers ([31], [32] and [33]) have focused on the application of BFS in evacuation planning.

1.2.3 Network and Communication

Network Robustness: In communication networks, the average sensitivity of BFS can help in assessing the robustness of network routing protocols. If routes are highly sensitive to changes(e.g. node failures or added nodes), network engineers can design more fault-tolerant routing mechanisms. Fault-tolerant network routing is highly investigated as in [34], [35] ,[36], [37], [38], [39], [40], [41] and [42].

1.2.4 Healthcare and Epidemic Modelling

Disease Spread Simulation: Understanding the sensitivity of BFS in epidemic models can help public health officials gauge how small changes in initial conditions(e.g. the location of an outbreak) can affect the spread of a disease. This can improve the accuracy of intervention strategies and resource allocation. In many papers([43], [44], [45], [46] and [47]), the BFS algorithm has been used in studies of epidemic modelling.

1.2.5 Urban Planning and Traffic Management

Traffic Flow Analysis: In urban planning, analyzing the sensitivity of BFS in traffic flow models can help in designing road networks that are less susceptible to disruptions(e.g. road closures or accidents). This can lead to more resilient and efficient traffic management systems.

1.2.6 Agriculture and Irrigation Systems

Irrigation Efficiency: In agricultural planning, understanding the sensitivity of BFS in irrigation systems can help optimize the layout and operation to ensure consistent water distribution, even when minor changes occur(e.g. blockages or changes in terrain).

1.2.7 Games and Puzzles

Game AI: In game development, understanding the sensitivity of BFS can help in creating AI that adapts better to changes in the game environment, providing a more challenging and dynamic experience for players.

1.3 Results of the Thesis

Given a graph G and a starting node s, for $i = 0, .., n-1$, let L_i(the nodes with layer number i) be the set of nodes reachable from s by i edges but not reachable from s by at most $i-1$ edges. The BFS algorithm explores the nodes of L_i for $i = 0, .., n-1$. To find the nodes of L_i, the algorithm selects a node u of L_{i-1} and checks every edge $\{u, v\}$; if v is not explored then mark v explored, put v in L_i and include edge $\{u, v\}$ in the BFS tree. When u has multiple neighbours, these neighbours can be checked in an arbitrary order. However different checking orders may create different BFS trees. When a BFS algorithm uses a deterministic ordering to check the neighbours of u, the algorithm is called a deterministic BFS(DBFS) algorithm, while an algorithm that uses a randomized ordering is called a randomized BFS(RBFS) algorithm. For the deterministic BFS algorithm on grid graphs, there are $4! = 24$ orderings to check the four neighbours of u. Assume the same ordering is used for every node u of a grid, We show that it suffices to analyze the average sensitivities of the DBFS algorithm with three orderings called the Clockwise ordering, Reading ordering and the Sign of the Cross ordering(details are given in Section 2.2.2).

We follow the definition of the average sensitivity in [1] for the perturbation with only one edge removed and analyze the average sensitivity of BFS algorithms on $m \times n$ grids. We obtained the following results for the three orderings of the DBFS algorithm and the RBFS algorithm on any starting node of a grid:

- The Clockwise DBFS algorithm has average sensitivity at most $1 + O(\frac{m}{n}) + O(\frac{n}{m})$, and $O(1)$ when $m = \Theta(n)$ and at most 3 when $m = n$.
- The Reading DBFS algorithm has average sensitivity at most $1 + O(\frac{m}{n}) + O(\frac{n}{m})$, and $O(1)$ when $m = \Theta(n)$ and at most 3 when $m = n$.
- The Sign of the Cross DBFS algorithm has average sensitivity at most 2 for any $2 \leq m, n$.
- The RBFS algorithm has average sensitivity at most $1 + O(\frac{m}{n}) + O(\frac{n}{m})$, and $O(1)$ when $m = \Theta(n)$ and at most 3 when $m = n$.
- The randomized BFS algorithm achieves a smaller average sensitivity than the Sign of Cross DBFS algorithm for some starting nodes in grids.

The above results show that the BFS algorithm has an average sensitivity of small constants on grid graphs, significantly smaller than $\Theta(|V(G)|)$ on arbitrary graphs. The Sign of the Cross DBFS algorithm has the smallest average sensitivity in the three orderings for the DBFS algorithm. The RBFS has a smaller average sensitivity than the Sign of Cross DBFS for many starting nodes in a grid, showing that randomization of the BFS algorithm can reduce the average sensitivity in many cases.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows: Chapter 2 gives the preliminaries. The average sensitivities of deterministic BFS algorithms and randomized BFS algorithms are analyzed in Chapter 3 and Chapter 4, respectively. We compare the deterministic and randomized BFS algorithms in Chapter 5. The final chapter concludes the thesis.

Chapter 2

Preliminaries

We now introduce the notions, terminologies and basic technologies of the thesis. Graphs in this thesis are simple graphs(no weights, no self-loop or multiple edges and no direction for the edges).

2.1 Grid

An $m \times n$ grid is a graph G with node set $V(G) = \{(i, j) | 0 \leq i \leq m - 1, 0 \leq j \leq n - 1\}$ and edge set $E(G) = \{\{(i, j), (i', j')\} | (|i - i'| = 1 \text{ and } j = j') \text{ or } (|j - j'| = 1 \text{ and } i = i')\}$. Where m represents the number of rows and n represents the number of columns of such a grid. Figure 2.1 represents a 4×5 grid.

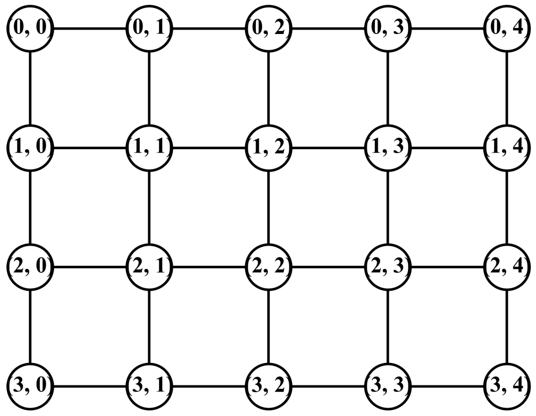

Figure 2.1: A 4×5 grid.
Note that the first number represents the number of rows and the second number represents the number of columns.

We also think of the nodes in the grid as a Cartesian coordinate system. We assume the x-axis is bottom-wards and represents the first number of the ordered pairs and that the y-axis is right-wards and represents the second number of the ordered pairs. These axes intersect at the origin $O(0, 0)$ which is the top-leftmost node of the grid.

For a node (i, j) of G, we call node $(i-1, j)$ the top neighbor, $(i+1, j)$ the bottom neighbor and $(i, j-1)$ the left neighbor, $(i, j+1)$ the right neighbor of (i, j).

2.2 BFS(Breadth-First Search) Algorithm

A high-level description of the BFS algorithm is that given a starting node s of a graph, the algorithm explores s, and then explores the nodes which are reachable from s by i edge but not reachable by $i-1$ edges for $i = 1, .., n-1$. A classical implementation of the BFS algorithm is to use a first-in first-out (FIFO) queue to keep track of the nodes to be explored. The main steps of this BFS algorithm with a BFS tree as output are as follows:

1. Initialize an empty BFS tree T an empty FIFO queue, mark the starting node s explored and add s to the queue.
2. While the queue is not empty, remove a node u from the queue, for every node v adjacent to u, if v is not explored then mark v explored, add v to the queue, and add edge $\{u, v\}$ to T.
3. Output T.

In Step 2, when node u has multiple neighbours which have not been explored, these neighbours can be explored in an arbitrary order. It is important to note that different orderings to explore these neighbours may give different BFS trees and thus result in different average sensitivities. A DBFS(deterministic BFS) algorithm uses a deterministic ordering in Step 2 for every node u, while an RBFS(randomized) algorithm explores the neighbours in a randomized ordering.

2.2.1 DBFS Algorithm on Grids

Since each node u of a grid has multiple neighbours, we need to consider different orderings to these neighbours in the DBFS algorithm. For this purpose, we assume every node u of a grid has four neighbours u_1 (top), u_2 (right), u_3 (bottom) and u_4 (left) (for node u with two or three neighbours, we add two or one dummy neighbours respectively). Then there $4! = 24$ different orderings to explore nodes u_1, u_2, u_3, u_4, each ordering can be represented by a permutation of $(1, 2, 3, 4)$. The DBFS algorithm considered in this thesis uses the same ordering (same permutation) for every node u to explore its four neighbours. We will explain all such orderings in section 2.2.2.

Here is a pseudo-code for the DBFS algorithm for girds:

Algorithm 1 DBFS Algorithm for Grids

```
Require: A grid $G$ and a starting node $s$.
Ensure: A BFS tree $T$ of $G$ rooted at $s$
 1: Initialize an empty tree $T$ and an empty FIFO queue $Q$
 2: Mark $s$ as explored and enqueue $s$ to $Q$
 3: while $Q$ is not empty do
 4:     dequeue $Q$ to get a node $u$
 5:     for each edge $(u, v)$ do (with a specific ordering)
 6:         if $v$ is not explored then
 7:             mark $v$ explored; enqueue $v$ to $Q$; $T = T \cup \{\{u, v\}\}$
 8:         end if
 9:     end for
10: end while
11: Output $T$
```

2.2.2 Orderings of the DBFS Algorithm

When it comes to the DBFS algorithm, as discussed, we should also determine the ordering in which we explore the neighbours of a node. Because a grid can be flipped and rotated, this problem resembles the number of ways we can make a necklace with four beads where the beads are "Top", "Left", "Bottom" and "Right" neighbours.

Now, consider the problem of making a necklace of size n using n distinct beads. Since a necklace can be rotated and flipped and it will still look the same(meaning the arrangement of beads is considered the same when it is rotated or flipped), we deal with a problem of counting unique arrangements where rotations and reflections of an arrangement are considered identical.

The answer for this problem would be $\frac{(n-1)!}{2}$. This is because we can arrange the necklace beads linearly in $n!$ ways and since we can have n circular shifts where they all correspond to the same ordering we are left with $\frac{n!}{n}$ arrangements. In addition to that, by flipping the necklace we can transition to another ordering, which means for every two such orderings, only one is a unique ordering. Hence the number of distinct orderings for this problem would be $\frac{n!}{2n}$ which is the same as $\frac{(n-1)!}{2}$. As an example, a necklace of size four using only four distinct beads would be made in $\frac{(4-1)!}{2} = \frac{3!}{2} = 3$ ways.

Therefore, there would only be three distinct class of orderings. We will now name the orderings here:

1) The Clockwise Ordering: By the clockwise ordering we mean the Top-Right-Bottom-

Left ordering. Since we can rotate and flip the grid and then run the DBFS algorithm with the Top-Right-Bottom-Left ordering all the other clockwise and counter-clockwise orderings correspond to this ordering. To be specific the following 8 orderings correspond to the same clockwise ordering that we have here:

1. Top-Right-Bottom-Left
2. Right-Bottom-Left-Top
3. Bottom-Left-Top-Right
4. Left-Top-Right-Bottom
5. Top-Left-Bottom-Right
6. Left-Bottom-Right-Top
7. Bottom-Right-Top-Left
8. Right-Top-Left-Bottom

We therefore refer to the DBFS with such ordering, as the Clockwise DBFS or simply CDBFS.

2) The Reading Ordering: The Reading Ordering is the Top-Left-Right-Bottom ordering as it resembles the reading or scanning pattern which reads text line by line from top to bottom and for each line from left to right. Again because we can rotate and flip the grid before we run the algorithm, any of the following orderings can be represented using this ordering:

1. Top-Left-Right-Bottom
2. Right-Top-Bottom-Left
3. Bottom-Right-Left-Top
4. Left-Bottom-Top-Right
5. Top-Right-Left-Bottom
6. Right-Bottom-Top-Left
7. Bottom-Left-Right-Top
8. Left-Top-Bottom-Right

We therefore refer to the DBFS with such ordering, as the Reading DBFS or simply RDBFS.

3) The Sign of the Cross Ordering: The Sign of the Cross is the Top-Bottom-Left-Right ordering. It refers to a ritual gesture in Christianity. This gesture involves touching the forehead(top), chest(bottom), and then both shoulders(Left and Right) to trace the shape of a cross. It could also be thought of as vertical first and then horizontal ordering. It is therefore evident that this ordering is the champion of all the following orderings(Since we can rotate and flip the grid first and then run the DBFS algorithm):

1. Top-Bottom-Left-Right
2. Right-Left-Top-Bottom
3. Bottom-Top-Right-Left
4. Left-Right-Bottom-Top
5. Top-Bottom-Right-Left
6. Right-Left-Bottom-Top
7. Bottom-Top-Left-Right
8. Left-Right-Top-Bottom

We therefore refer to the DBFS with such ordering, as the Sign of the Cross DBFS or simply SDBFS. Figure 2.2 shows the three DBFS trees on a grid.

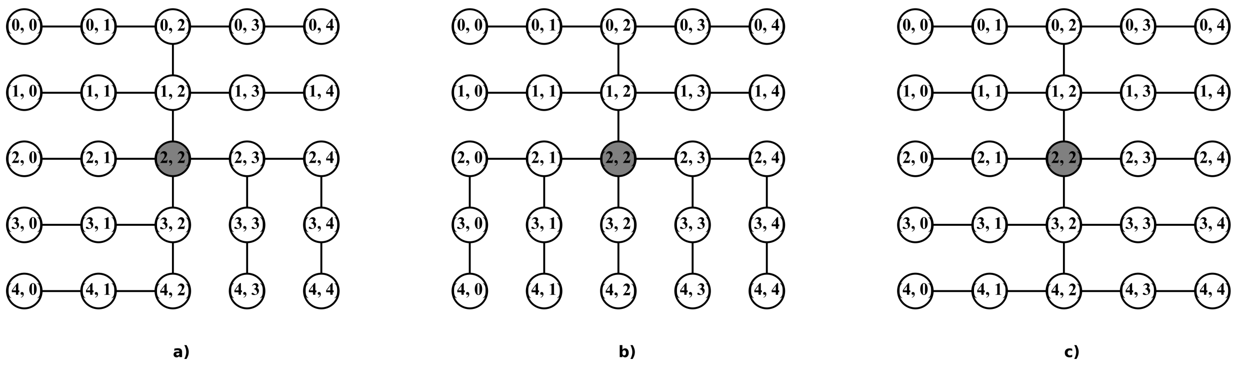

Figure 2.2: DBFS trees for all the 3 orderings on a 5 × 5 Grid
From left to right: a)Clockwise DBFS tree b)Reading DBFS tree
c) Sign of the Cross DBFS tree
The node with the coordinates (2,2) is the starting node in all of these BFS trees

It suffices to analyze the average sensitivities of Algorithm 1 for the Clockwise ordering, Reading ordering and Sign of the Cross ordering.

As an example of rotation, imagine you want to run the deterministic BFS algorithm with the Right-Bottom-Left-Top ordering with the starting node of (i, j) on a $m \times n$ grid. It is equivalent to first rotating the grid counterclockwise(you will then get a $n \times m$ grid) and then running the BFS algorithm with the Top-Right-Bottom-Left with the starting node of $(n - j - 1, m - i - 1)$ and then rotating the result clockwise.

Also, as an example of reflection over a column, imagine you want to run the deterministic BFS algorithm with the Top-Left-Bottom-Right ordering with the starting node of (i, j) on a $m \times n$ grid. It is equivalent to first reflecting the grid over a column(you will still have a $m \times n$ grid) and then running the BFS algorithm with the Top-Right-Bottom-Left with the starting node of $(i, n - j - 1)$ and then reflecting the result one more time.

As a result, the BFS trees of each two orderings within a single class of orderings are the same using rotation and reflection. So it is enough to study only one ordering within each class. Using group theory terms, every two orderings within one class of orderings are congruent to each other using a combination of rotations, reflections, and translations. Therefore, each of these 3 classes is a symmetry group.

Dihedrals are a type of symmetry group that represents the symmetries of a regular polygon, including both rotations and reflections. The dihedral group D_n corresponds to the symmetries of a regular n-gon(a polygon with n sides). For D_4, it represents the symmetries of a square and has 8 elements, consisting of 4 rotations and 4 reflections. In group theory, S_4 refers to the symmetric group of 4 elements. It consists of all the possible permutations of four distinct objects and is one of the most studied groups due to its rich structure and significance in various mathematical contexts. And since $|S_4 : D_4| = \frac{|S4|}{|D4|} = \frac{24}{8}$ we have 3 dihedral groups for the orderings.

2.2.3 RBFS(Randomized BFS) Algorithm on Grids

Given a grid G and a starting node s, for $0 \leq i \leq n - 1$ let L_i be the set of nodes which are reachable from s by i edges but not reachable by $i - 1$ edges. Notice that $L_0 = \{s\}$ and L_1 is the set of neighbors of s. Let $E_{i-1,i}$ be the set of edges between a node of L_{i-1} and a node of L_i. We use the following randomized approach to explore the graph: Assume $E_{i-1,i} = \{e_1, .., e_r\}$, where $r = |E_{i-1,i}|$. We create a permutation $(p_1, .., p_r)$ of $(1, 2, .., r)$ uniformly at random. Then we check edges u, v, $u \in L_{i-1}$, in the order of e_{p_i}, $i = 1, .., r$. If v is not explored then put v in L_i and include edge $\{u, v\}$ in the BFS tree. Below is a pseudo-code of the RBFS algorithm by the above approach:

Algorithm 2 RBFS Algorithm for Grids

```
Require: A grid $G$ and a starting node
Ensure: A BFS tree of the grid
 1: Initialize an empty tree $T, A, B$
 2: $A = \{s\}$; mark $s$ explored
 3: for $i = 1$ to $n - 1$ do
 4:     $E_{a,b} = \{\{u, v\} | u \in A \ and \ v \ not \ explored\} = \{e_1, .., e_r\}$
 5:     Create a random permutation $(p_1, p_2, .., p_r)$ of $(1, 2, .., r)$
 6:     for $j = 1$ to $r$ do
 7:         if $v$ in edge $e_{p_j} = \{u, v\}$ is not explored then
 8:             mark $v$ explored; $B = B \cup \{v\}$; $T = T \cup \{\{u, v\}\}$;
 9:         end if
10:     end for
11:     $A = B; B = \emptyset$;
12: end for
13: Output $T$;
```

Algorithm 2 can be implemented in $O(m + n)$ time for G represented by an adjacent list.

2.3 Average Sensitivity of Graph Algorithms

2.3.1 Definitions

A graph G has the node set $V = \{v_1, \dots, v_{|V|}\}$ and the edge set $E = \{e_1, \dots, e_{|E|}\}$. Varma and Yoshida [1] introduced the notion of average sensitivity of graph algorithms for the study of stable algorithms. Assume that a solution for a problem in G is represented by a subset of E, expressed by a binary sequence $(b_1, \dots, b_{|E|})$, where $b_i = 1$ for edge e_i included in the solution, and $b_i = 0$ otherwise. Let $\{B, B'\}$ be the Hamming distance between two solutions $B = (b_1, \dots, b_{|E|})$ and $B' = (b'_1, \dots, b'_{|E|})$. For a subset E' of E, let $G' = G \setminus E'$ be the graph obtained by updating edges (e.g., remove edge or change edge weight) of E'. For an algorithm A, let $A(G)$ and $A(G')$ be the sets of solutions of A on G and G', respectively. For a randomized algorithm A, let p_j and q_l be the probabilities that solutions $B_j \in A(G)$ and $B'_l \in A(G')$ are outputted by A on G and G', respectively. The Earth mover's distance [2] between $A(G)$ and $A(G')$ is

$$EM(A(G), A(G')) = \min \sum_{B_j \in A(G)} \sum_{B'_l \in A(G')} x_{jl} d_{\text{Ham}}(B_j, B'_l) \tag{2.1}$$

subject to $x_{jl} \geq 0$, and for every j, $\sum_l x_{jl} = p_j$, and for every l, $\sum_j x_{jl} \leq q_l$.

$EM(A(G), A(G'))$ is a metric on the difference between solution sets $A(G)$ and $A(G')$. *Average sensitivity* of algorithm A is defined as:

$$\mathbb{E}_{e \in E}[EM(A(G), A(G \setminus \{e\}))] \tag{2.2}$$

where edge e is selected uniformly at random from E. We can compute the average sensitivity of the randomized BFS on G by an LP solver or a minimum-cost maximum-flow algorithm.

For a deterministic algorithm A which computes a unique solution B on G and unique solution B' on G', $EM(A(G), A(G')) = d_{\mathrm{Ham}}(B, B')$ and average sensitivity of A is

$$\mathbb{E}_{e \in E(G)} d_{\mathrm{Ham}}(A(G), A(G \setminus \{e\})) = \frac{1}{|E(G)|} \sum_{e \in E(G)} d_{\mathrm{Ham}}(A(G), A(G \setminus \{e\})) \tag{2.3}$$

For the DBFS algorithm, Hamming distance can therefore be considered the cardinality of the symmetric differences of the edge sets. i.e.

$$d_{Ham(G,G')} = d_{Ham(E(G),E(G'))} = |E(G) \triangle E(G')| = |(E(G) \setminus E(G')) \cup (E(G') \setminus E(G))|$$

Algorithm A is called *stable-on-average* if A has a small average sensitivity(A finds a solution in $A(G) \cap A(G')$ with high probability).

Average sensitivity of algorithm A can be generalized to k-average sensitivity, $k \geq 1$, with $G' = G \setminus E'$ for every subset E' of E with $|E'| = k$ [1]. Average sensitivity is dependent on solution representation. For a problem with a solution represented by a subset of nodes in V, average sensitivity is defined in terms of subsets of V. We can also define average sensitivity with respect to adding edges or nodes to G.

2.3.2 Average Sensitivity of BFS Algorithm on Arbitrary Graphs

It is shown in [1] that the single source shortest paths algorithms for arbitrary graphs G have average sensitivity $\Theta(|V(G)|)$. This implies that the BFS algorithms with BFS trees as output have an average sensitivity of $\Theta(|V(G)|)$ as well. Below we give a simple example to show the $\Theta(|V(G)|)$ bound.

First, a trivial upper bound on the average sensitivity of the BFS algorithm is $O(|V(G)|)$ as a BFS tree has $|V(H)| - 1$ edges. Next consider a path graph, with set of nodes $V = \{v_1, v_2, \cdots, v_{|V|}\}$ and edge set $E = \{\{i, i+1\} | 1 \leq i \leq |V| - 1\}$. Now let us consider running the BFS algorithm with v_1 as the starting node. Regardless of any ordering to explore new nodes in the BFS, all the edges of this path graph will appear in the output

BFS tree. However, if we remove any edge say $\{i, i+1\}$ then the edges from $i+1$ onward will not appear in the output. As a result, the average sensitivity would be:

$$\frac{1}{|E|}\sum_{i=1}^{|E|} i = \frac{1}{|E|}\frac{(1+|E|)|E|}{2} = \frac{1+|E|}{2} = \frac{1+|V|-1}{2} = \frac{|V|}{2} = \Omega(|V(G)|)$$

So, the average sensitivity of the BFS algorithm is $\Theta(|V(G)|)$ for arbitrary graphs.

2.4 Some Terminologies and Tools

2.4.1 Partitioning the Grid into Regions

Given a starting node of an $m \times n$ grid, we denote s by its coordinates (i, j). Edges of the grid, based on their positions with respect to (i, j), can be partitioned into eight regions(if (i, j) is on the first/last row/column, some of the regions are empty):

1. Let $R_1 := \{\{(y, x), u\} | y < i, j < x, u \in V(G), u \neq (y, x)\}$
   In other words, R_1 represents the set of edges on the top-right region of the starting node.

2. Let $R_2 := \{\{(y, x), u\} | y < i, x < j, u \in V(G), u \neq (y, x)\}$
   In other words, R_2 represents the set of edges on the top-left region of the starting node.

3. Let $R_3 := \{\{(y, x), u\} | i < y, x < j, u \in V(G), u \neq (y, x)\}$
   In other words, R_3 represents the set of edges on the bottom-left region of the starting node.

4. Let $R_4 := \{\{(y, x), u\} | i < y, j < x, u \in V(G), u \neq (y, x)\}$
   In other words, R_4 represents the set of edges on the bottom-left region of the starting node.

5. Let $B_{12} := \{\{(i-k, j), (i-k-1, j)\} | 0 \leq k \leq i-1\}$
   In other words, B_{12} represents the set of edges on the border of R_1 and R_2.

6. Let $B_{23} := \{\{(i, j-k), (i, j-k-1)\} | 0 \leq k \leq j-1\}$
   In other words, B_{23} represents the set of edges on the border of R_2 and R_3.

7. Let $B_{34} := \{\{(i+k, j), (i+k+1, j)\} | 0 \leq k \leq m-i-2\}$
   In other words, B_{34} represents the set of edges on the border of R_3 and R_4.

8. Let $B_{41} := \{\{(i, j+k), (i, j+k+1)\} | 0 \leq k \leq n-j-2\}$
   In other words, B_{41} represents the set of edges on the border of R_4 and R_1.

We also denote $B := B_{12} \cup B_{23} \cup B_{34} \cup B_{41}$ which represents all the borders and $\tilde{B} = E(G) \setminus B$ which represents all the edges not on the borders which equivalently means $\tilde{B} = R_1 \cup R_2 \cup R_3 \cup R_4$

This means that:

$$E = B_{12} \cup B_{23} \cup B_{34} \cup B_{41} \cup \tilde{B} \tag{2.4}$$

For the grid G and a subgraph $G\backslash\{e\}$ obtained by removing an edge e from G, we denote by $T(G)$ a deterministic BFS tree on G and $T(G\backslash\{e\})$ a deterministic BFS tree on $G\backslash\{e\}$.

For each of the edges $\{(i, j \pm k), (i, j \pm (k+1))\}, \{(i \pm k, j), (i \pm (k+1), j)\}$, we define the distance between the edge and the starting node(i, j) to be the Manhattan distance between the starting node and the closer endpoint of the edge(closer to the starting node). That means that the distance is simply equal to k.

2.4.2 Kronecker Delta

The Kronecker delta, denoted as $\delta_{i,j}$, is a function of two variables that is 1 if the variables are equal, and 0 otherwise. Mathematically, it is defined as:

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \tag{2.5}$$

2.4.3 Min-Cost Max-Flow Problem

The Min-Cost Max-Flow(MCMF) problem is a fundamental problem in network flow theory. It combines the concepts of maximum flow and minimum cost, seeking to determine the maximum flow possible from a source to a sink in a flow network, such that the total cost of the flow is minimized. This problem is crucial in various applications, such as transportation, logistics, and network design. We now formally define the problem.

Consider a directed graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Each edge $(u, v) \in E$ has a capacity $c(u, v) \geq 0$ and a cost $\text{cost}(u, v)$. The goal is to find a flow $f : E \to \mathbb{R}$ that satisfies the following conditions:

1. **Capacity Constraints:** For each edge $(u, v) \in E$, the flow $f(u, v)$ must not exceed the capacity of the edge:
$$0 \leq f(u, v) \leq c(u, v).$$

2. **Flow Conservation:** For each vertex $v \in V \setminus \{s, t\}$ (excluding the source s and the sink t), the total flow into the vertex must equal the total flow out of the vertex:

$$\sum_{u:(u,v)\in E} f(u,v) = \sum_{w:(v,w)\in E} f(v,w).$$

3. **Objective:** The objective is to maximize the total flow from the source s to the sink t while minimizing the total cost of the flow:

$$\text{Minimize} \sum_{(u,v)\in E} \text{cost}(u,v) \cdot f(u,v).$$

Chapter 3

DBFS on Grids

In this chapter, we analyze the average sensitivity of the DBFS algorithm with the Clockwise, Reading and Sign of the Cross orderings on grids. For that, we need to have the outputs of the DBFS algorithm on both the original grid and the damaged grid.

3.1 Average Sensitivity of Clockwise DBFS on Grids

Let us first see what the output of the Clockwise DBFS tree would look like on both the original grid and then the damaged grid. After that, we will calculate their Hamming distance leading us to an answer for the average sensitivity of this algorithm.

3.1.1 The Clockwise DBFS Tree on Grids

We take the coordinates of the starting node to be (i, j). Now if we run the clockwise DBFS algorithm as discussed above, we would select $(m \times n) - 1$ edges to form our DBFS tree(since we have a connected grid and the number of edges in a tree is one unit fewer than the number of nodes). Also, as discussed in section 2.4.1 the starting node partitions the grid into at most 8 regions. We say at most because if the starting node is on the first row for example R_1, R_2 and B_{12} would be empty sets.

Now, Let us examine the corresponding edges in the clockwise DBFS tree of a grid in all of these partitions:

- B : As for the edges in the borders set, it is obvious that all the edges will be selected since the nodes on the borders will be explored through the neighbouring node that is closer to the starting node and is in the same direction.
- $\tilde{B}$: As for the edges not in the border set(or equivalently edges in $\tilde{B}$), things are a bit more complicated:

- R_1 : In this case, all horizontal edges will be selected and all the vertical ones will not be selected. This is because the nodes on the top have a higher priority than the nodes on the right. Therefore, for all the nodes in this partition, their left neighbour will be explored before their bottom neighbour.
- R_2 : Similarly, all the horizontal edges will be selected and all the vertical edges will not be selected. This is because the nodes on the top have a higher priority than the nodes on the left.
- R_3 : Similarly, all the horizontal edges will be selected and all the vertical edges will not be selected. This is because the nodes on the bottom have a higher priority than the nodes on the left.
- R_4 : In contrast to the previous three cases, here, all the vertical edges will be selected and all the horizontal edges will not be selected. This is because the nodes on the right have a higher priority than the nodes on the bottom.

Figure 3.1 gives an example of the Clockwise DBFS tree on a grid.

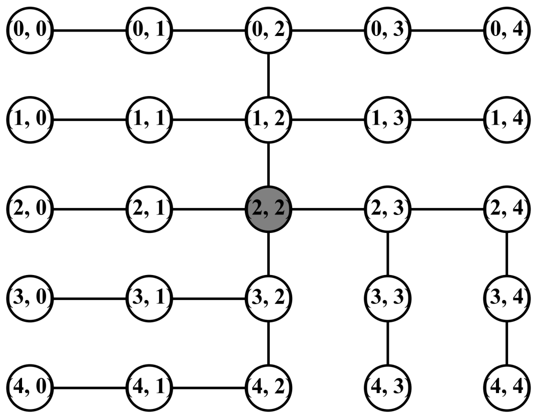

Figure 3.1: The Clockwise DBFS tree on a 5×5 grid
The starting node in this example is (2,2)

3.1.2 The Clockwise DBFS Tree on Damaged Grids

Now let us discuss what happens to the BFS tree if we remove an edge from the grid graph. The edge either belongs to the borders set(B) or the $\tilde{B}$ set.
Depending on how close the removed edge is to the starting node, the Hamming distance between the BFS tree on the damaged grid and the original grid varies.

(1) Removing an Edge from B_{12}

Let us analyze what will happen if one of the edges in this section is removed. For simplicity, we remove the closest edge to the starting node(if one of the other edges is removed, you may assume the starting node is adjacent to this edge). Since the main problem is how the nodes in this section are explored, we need to analyze the left and right nodes of this section. Now, if $j = 0$ or $j = n$ then the nodes in this section will have only one horizontal neighbour, and in all the other cases they will have two. Let us analyze these cases one by one:

If $j = 0$ then the nodes in this section will be explored by their right side neighbour. So the horizontal edge will remain the same in both BFS trees. However, instead of the edges in the border, the parallel edges on their right side will be chosen. And since we have i edges in this border, we also have i parallel edges on the right, so the difference between the BFS trees would be $2i$(see Figure 3.2 for an example).

If $j = n - 1$ then similar to the previous case, the total Hamming distance would be $2i$(see Figure 3.3 for an example).

And in case $j \neq 0, n$, none of the edges on the border will be selected. Instead, if applicable, the closest parallel edges both on the left and right of this border will be selected. Additionally, the closest edges in R_2 which are perpendicular to the border will not be selected. So in general the Hamming distance would be $4i$ if j is not 0 or $n - 1$(see Figure 3.4 for an example). The rest of the grid will remain the same. Therefore we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = \begin{cases} 2i & if \quad j = 0 \quad or \quad n-1, \\ 4i & otherwise. \end{cases}$$

In general, if the removed edge has a distance of d with the starting node, we can use $(i - d)$ instead of i, and the above formula would become:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = \begin{cases} 2(i-d) & if \quad j = 0 \quad or \quad n-1, \\ 4(i-d) & otherwise. \end{cases}$$

Where d is from 0 to $i - 1$. Notice that if i is 0, B_{12} would be an empty set and everything about this distance would be 0.

We can use the Kronecker delta notation, and simplify the results:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(i - d) \tag{3.1}$$

And now we calculate the summation of the Hamming distance over all the edges that can be removed from this section. We will later use this value to calculate the average sensitivity. Therefore the summation over d would be:

$$\begin{aligned}
&\sum_{e\in B_{12}} d_{\text{Ham}}(T(G),T(G\setminus\{e\})) = \sum_{d=0}^{i-1}(4-2(\delta_{j,0}+\delta_{j,n-1}))(i-d)\\
&= (4-2(\delta_{j,0}+\delta_{j,n-1}))\sum_{d=0}^{i-1}(i-d) = (4-2(\delta_{j,0}+\delta_{j,n-1}))\frac{i(i+1)}{2}\\
&= (2-(\delta_{j,0}+\delta_{j,n-1}))i(i+1)
\end{aligned} \tag{3.2}$$

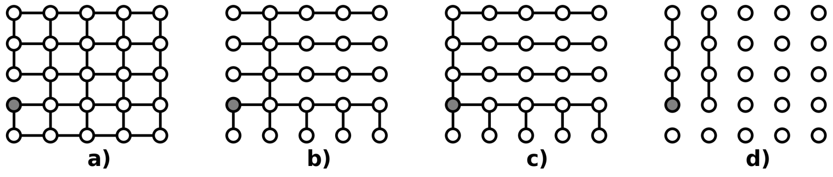

Figure 3.2: The $j = 0$ case of a damaged(from B_{12}) grid in CDBFS

The starting node in all of the following 4 graphs is (3,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

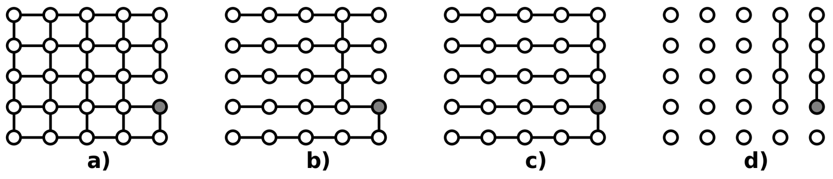

Figure 3.3: The $j = n - 1$ case of a damaged(from B_{12}) grid in CDBFS

The starting node in all of the following 4 graphs is (3,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

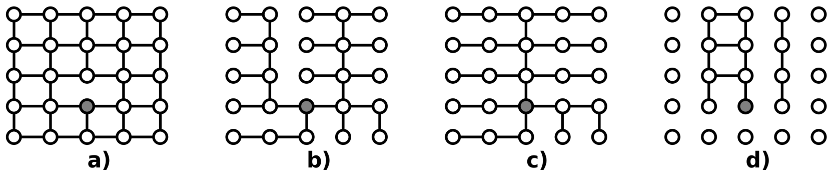

Figure 3.4: The general case of a Damaged(from B_{12}) Grid in CDBFS

The starting node in all of the following 4 graphs is (3,2)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(2) Removing an Edge from B_{23}

If we remove an edge in this partition, instead of getting to the other end of the removed edge directly, we will visit that node through its top node. The rest of the graph remains unchanged. Therefore the Hamming distance between the two outputs will be only 2(one for the removed edge and one for the edge that reaches the next node.)

And even if $i = 0$, we get to the next node through the bottom one and since the top has higher priority than the left, the rest of the graph remains the same. So in this case:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = 2 - 2(\delta_{j,0}) \tag{3.3}$$

So except for the $j = 0$ case where B_{23} would be empty, regardless of d, the mentioned Hamming distance is always 2.

It means that the summation of the Hamming distance over all the edges that can be removed from this section is:

$$\sum_{e \in B_{23}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = 2j \tag{3.4}$$

Figures 3.5, 3.6 and 3.7 illustrate different cases of removing an edge from this section.

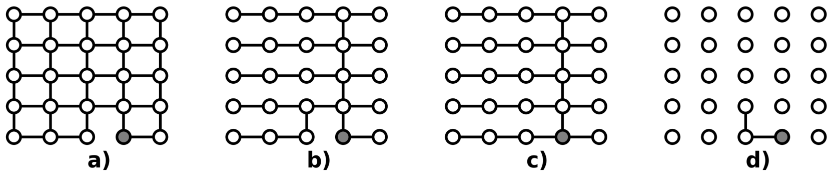

Figure 3.5: The $i = m - 1$ case of a damaged(from B_{23}) grid in CDBFS

The starting node in all of the following 4 graphs is (4,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

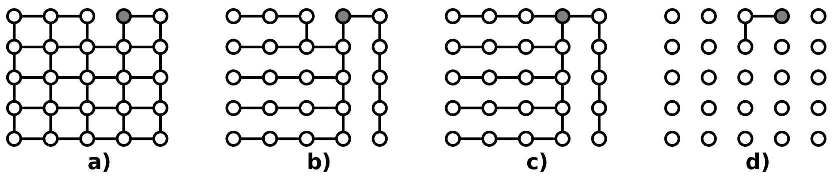

Figure 3.6: The $i = 0$ case of a damaged(from B_{23}) grid in CDBFS

The starting node in all of the following 4 graphs is (0,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

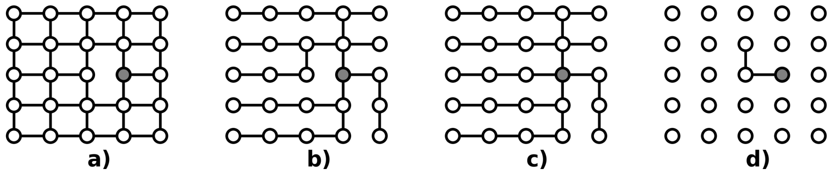

Figure 3.7: The general case of a damaged(from B_{23}) grid in CDBFS

The starting node in all of the following 4 graphs is (2,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(3) Removing an Edge from B_{34}

This case is very similar to the case 1 as it produces the same structure. Therefore we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = \begin{cases} 2(m - i - d' - 1) & if \quad j = 0 \quad or \quad n - 1, \\ 4(m - i - d' - 1) & otherwise. \end{cases}$$

Which is the same as saying:

$$\begin{aligned} & d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(m - i - d' - 1) \\ & \text{Where } 0 \leq d' \leq m - i - 2 \end{aligned} \tag{3.5}$$

Now we take the summation of all the edges that can be removed from this section:

$$\begin{aligned} & \sum_{e \in B_{34}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = \sum_{d'=0}^{m-i-2} (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(m - i - d' - 1) \\ & = (4 - 2(\delta_{j,0} + \delta_{j,n-1})) \sum_{d'=0}^{m-i-2} (m - i - d' - 1) \\ & = (4 - 2(\delta_{j,0} + \delta_{j,n-1})) \frac{(m - i)(m - i - 1)}{2} = (2 - (\delta_{j,0} + \delta_{j,n-1}))(m - i)(m - i - 1) \end{aligned} \tag{3.6}$$

Figures 3.8, 3.9 and 3.10 illustrate different cases of removing an edge from this section.

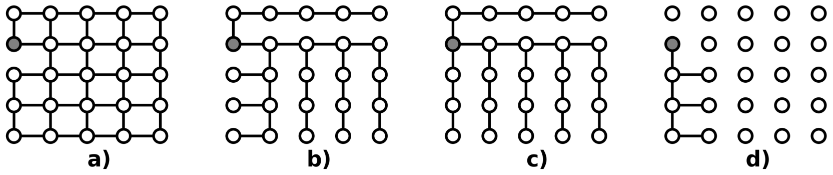

Figure 3.8: The $j = 0$ case of a damaged(from B_{34}) grid in CDBFS

The starting node in all of the following 4 graphs is (1,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

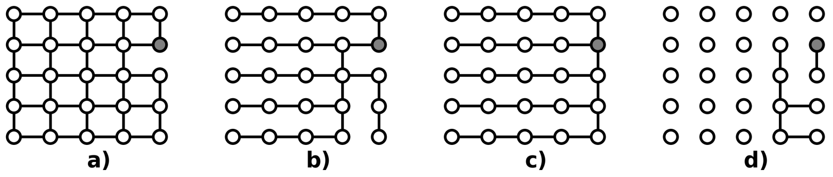

Figure 3.9: The $j = n - 1$ case of a damaged(from B_{34}) grid in CDBFS

The starting node in all of the following 4 graphs is (1,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

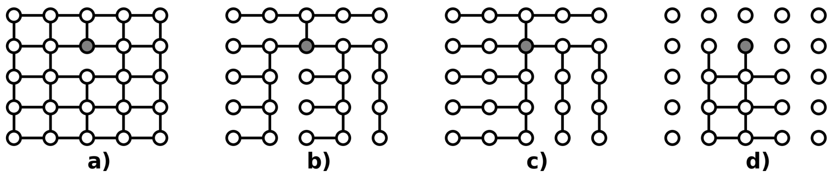

Figure 3.10: The general case of a damaged(from B_{34}) grid in CDBFS

The starting node in all of the following 4 graphs is (1,2)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(4) Removing an Edge from B_{41}

This is also similar to the previous case. However, we should use j instead of i. In addition to that we should use n instead of m. Therefore we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = \begin{cases} 2(n-j-d''-1) & if \quad i = 0 \quad or \quad m-1, \\ 4(n-j-d''-1) & otherwise. \end{cases}$$

Which is the same as saying:

$$\begin{aligned} & d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{i,0} + \delta_{i,m-1}))(n-j-d''-1) \\ & \text{Where} 0 \le d'' \le n-j-2 \end{aligned} \tag{3.7}$$

Now we take the summation of all the edges that can be removed from this section:

$$\begin{aligned} & \sum_{e \in B_{41}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = \sum_{d''=0}^{n-j-2} (4 - 2(\delta_{i,0} + \delta_{i,m-1}))(n-j-d''-1) \\ & = (4 - 2(\delta_{i,0} + \delta_{i,m-1})) \sum_{d''=0}^{n-j-2} (n-j-d''-1) \\ & = (4 - 2(\delta_{i,0} + \delta_{i,m-1})) \frac{(n-j)(n-j-1)}{2} = (2 - (\delta_{i,0} + \delta_{i,m-1}))(n-j)(n-j-1) \end{aligned} \tag{3.8}$$

Figures 3.11, 3.12 and 3.13 illustrate different cases of removing an edge from this section.

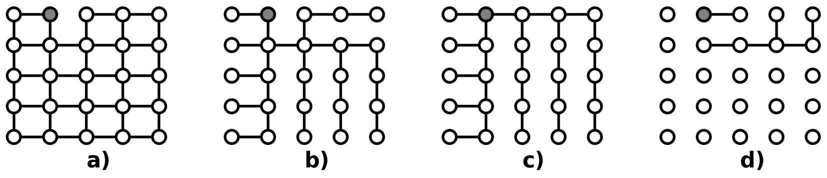

Figure 3.11: The $i = 0$ case of a damaged(from B_{41}) grid in CDBFS

The starting node in all of the following 4 graphs is (0,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} *to the starting node is removed.*

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

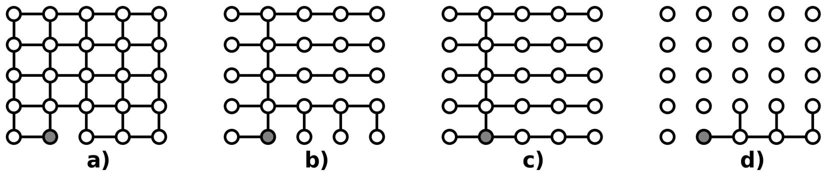

Figure 3.12: The $i = m - 1$ case of a damaged(from B_{41}) grid in CDBFS

The starting node in all of the following 4 graphs is (4,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} *to the starting node is removed.*

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

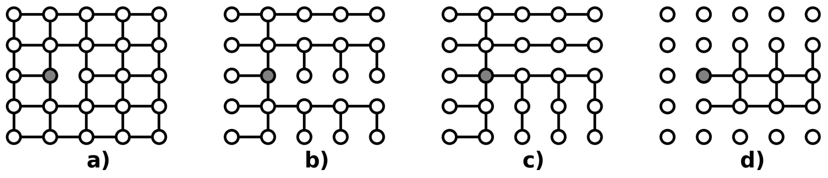

Figure 3.13: The general case of a damaged(from B_{41}) grid in CDBFS

The starting node in all of the following 4 graphs is (2,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} to the starting node is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(5) Removing an Edge from $\tilde{B}$

As explained so far, only some of the edges of the grid will be in its corresponding BFS tree. If the edge that is removed does not belong to the BFS tree, then it does not affect the BFS tree at all so the Hamming distance would be 0. However, if it does belong to the BFS tree of the original grid, then in addition to the edge being removed we get to the farther end point of this edge through another edge that was not selected before, leading us to the Hamming distance of 2. And the rest of the BFS tree remains the same. This is because the farther endpoint of the removed edge is the first node that will be visited through its other adjacent neighbour. And because of that the queue will be the same from onwards.

We also know that the number of edges in this set that are in the BFS tree is equal to the number of edges that are not in the BFS tree(because for every vertical edge, there is a horizontal edge) therefore, the Hamming distance on average is 1 in on this set. This means that the summation of the Hamming distance of all the edges that can be removed is equal to the number of edges in this set:

$$\sum_{e \in \tilde{B}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = 2(m-1)(n-1) \tag{3.9}$$

Figure 3.14 illustrates the case of removing an edge from this section.

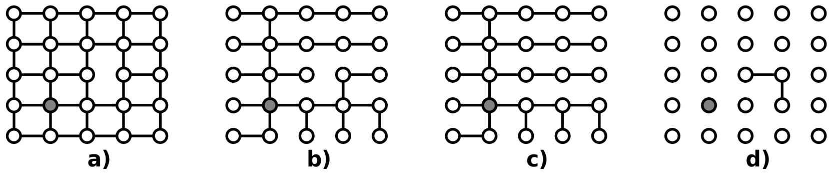

Figure 3.14: The damaged(from $\tilde{B}$) grid in CDBFS

The starting node in all of the following 4 graphs is (3,1)

From left to right: Figure a) represents a damaged grid where one of the edges in $\tilde{B}$ is removed.

Figure b) illustrates a CDBFS tree on the graph a).

Figure c) shows a CDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

3.1.3 Calculating an Upper Bound for the Average Sensitivity of Clockwise DBFS

As stated in Equation 2.3 the average sensitivity is

$$\begin{aligned}&\text{AvgS\_CDBFS}_{i,j} = \\ &\mathbb{E}_{e \in E(G)} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = \frac{1}{|E(G)|} \sum_{e \in E(G)} d_{\text{Ham}}(T(G), T(G \setminus \{e\}))\end{aligned}$$

From Equation 2.4 we have $E(G) = B_{12} \cup B_{23} \cup B_{34} \cup B_{41} \cup \tilde{B}$. Now the sum of Hamming distances can be expressed as:

$$\begin{aligned}\sum_{e \in E(G)} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = &\sum_{e \in B_{12}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) \\ &+ \sum_{e \in B_{23}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) \\ &+ \sum_{e \in B_{34}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) \\ &+ \sum_{e \in B_{41}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) \\ &+ \sum_{e \in \widetilde{B}} d_{\text{Ham}}(T(G), T(G \setminus \{e\}))\end{aligned} \tag{3.10}$$

We already have calculated each of these summations in Equations 3.2, 3.4, 3.6, 3.8 and 3.9. Substituting for the summations we get:

$$\begin{aligned}\sum_{e\in E(G)} d_{\mathrm{Ham}}(T(G),T(G\setminus\{e\})) = {} & (2-(\delta_{j,0}+\delta_{j,n-1}))i(i+1)\\ & +2j\\ & +(2-(\delta_{j,0}+\delta_{j,n-1}))(m-i)(m-i-1)\\ & +(2-(\delta_{i,0}+\delta_{i,m-1}))(n-j)(n-j-1)\\ & +2(m-1)(n-1)\end{aligned}$$

Substituting everything back to the definition of average sensitivity(2.3) we get:

$$\begin{aligned}\mathrm{AvgS\_CDBFS}_{i,j} &= \frac{\sum_{e\in E(G)} d_{\mathrm{Ham}}(T(G),T(G\setminus\{e\}))}{|E(G)|}\\ &= \frac{(2-(\delta_{j,0}+\delta_{j,n-1}))(i(i+1)+(m-i)(m-i-1))}{|E(G)|}\\ &+ \frac{2j}{|E(G)|}\\ &+ \frac{(2-(\delta_{i,0}+\delta_{i,m-1}))(n-j)(n-j-1)}{|E(G)|}\\ &+ \frac{|E(G)|-m-n+2}{|E(G)|}\end{aligned} \tag{3.11}$$

Now, Let us find an upper bound for this average sensitivity:
The function $f(i)=i(i+1)+(m-i)(m-i-1)$ is parabolic and has a maximum value of $f(0)=f(m-1)=m(m-1)$ for $0\le i\le m-1$. For $1\le i\le m-2$, $f(i)\le(m-1)^2$. The functions $g(j)=2(n-j)((n-j-1)+2j)$ and $h(j)=(n-j)(n-j-1)+2j$ are monotone decreasing in j for $0\le j\le n-1$.

For ($i=0$ or $i=m-1$) and ($j=0$ or $j=n-1$), we have

$$\sum_{e\in E(G)} \frac{d_{\mathrm{Ham}}(T(G),T(G)\setminus\{e\})}{|E(G)|} \le \frac{m(m-1)+n(n-1)-|E(G)|-m-n+2}{|E(G)|}$$

$$=1+\frac{(m-1)^2+(n-1)^2}{m(n-1)+n(m-1)} \le 1+\frac{(m-1)^2+(n-1)^2}{2(m-1)(n-1)}$$

$$=1+\frac{m-1}{2(n-1)}+\frac{n-1}{2(m-1)}.$$

For $1\le i\le m-2$ and ($j=0$ or $j=n-1$),

$$\sum_{e \in E(G)} \frac{d_{\mathrm{Ham}}(T(G), T(G) \setminus \{e\})}{|E(G)|} \le \frac{(m-1)^2 + 2n(n-1) - |E(G)| - m - n + 2}{|E(G)|}$$

$$= 1 + \frac{(m-1)^2 + 2n(n-1) - m - n + 2}{m(n-1) + n(m-1)} \le 1 + \frac{(m-1)^2 + 2(n-1)^2 + n - m}{2(m-1)(n-1)}$$

$$= 1 + \frac{m-1}{2(n-1)} + \frac{n-1}{(m-1)} + \frac{n-m}{2(m-1)(n-1)}.$$

For $(i = 0$ or $i = m-1)$ and $1 \le j$,

$$\sum_{e \in E(G)} \frac{d_{\mathrm{Ham}}(T(G), T(G) \setminus \{e\})}{|E(G)|} \le \frac{2m(m-1) + (n-1)^2 - |E(G)| - m - n + 2}{|E(G)|}$$

$$= 1 + \frac{2(m-1)^2 + (n-1)^2 + m - n}{m(n-1) + n(m-1)} \le 1 + \frac{2(m-1)^2 + (n-1)^2}{2(m-1)(n-1)}$$

$$= 1 + \frac{m-1}{(n-1)} + \frac{n-1}{2(m-1)} + \frac{m-n}{2(m-1)(n-1)}.$$

For $1 \le i \le m-1$ and $1 \le j \le n-1$,

$$\sum_{e \in E(G)} \frac{d_{\mathrm{Ham}}(T(G), T(G) \setminus \{e\})}{|E(G)|} \le \frac{2(m-1)^2 + 2(n-1)^2 - |E(G)| - m - n + 2}{|E(G)|}$$

$$= 1 + \frac{2(m-1)^2 + 2(n-1)^2}{m(n-1) + n(m-1)} \le 1 + \frac{2(m-1)^2 + 2(n-1)^2}{2(m-1)(n-1)}$$

$$= 1 + \frac{m-1}{n-1} + \frac{n-1}{m-1}.$$

Therefore, we have the following theorem:

Theorem 3.1 The Clockwise DBFS algorithm on $m \times n$ grids has the average sensitivity of at most $1 + \frac{m-1}{n-1} + \frac{n-1}{m-1}$, and $O(1)$ for $m = \Theta(n)$ and at most 3 for $m = n$ for any starting node.

3.1.4 Average of Average Sensitivity over All Starting Nodes for CDBFS

Now let us analyze the average of average sensitivity over all nodes of a grid as a starting node for the Clockwise DBFS algorithm. Before we do that, Let us rewrite the average sensitivity in this ordering. Rewriting Equation 3.11:

$$\text{AvgS\_CDBFS}_{i,j} = \frac{\sum_{e\in E(G)} d_{\text{Ham}}(T(G), T(G\setminus\{e\}))}{|E(G)|}$$
$$= \frac{(2-(\delta_{j,0}+\delta_{j,n-1}))(i(i+1)+(m-i)(m-i-1))}{|E(G)|}$$
$$+ \frac{2j}{|E(G)|}$$
$$+ \frac{(2-(\delta_{i,0}+\delta_{i,m-1}))(n-j)(n-j-1)}{|E(G)|}$$
$$+ \frac{|E(G)|-m-n+2}{|E(G)|}$$

We want to get rid of i and j. So now we take an average over all the possible starting nodes.

$$\text{AvgS\_CDBFS}_{T} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\text{AvgS\_CDBFS}_{i,j}$$
$$= \frac{|E(G)|-m-n+2}{|E(G)|} + \frac{1}{mn|E(G)|}\sum_{j=0}^{n-1}[2-(\delta_{j,0}+\delta_{j,n-1})]$$
$$\times \sum_{i=0}^{m-1}\Big[\ 2i^2+i(2-2m)+m^2-m\Big]$$
$$+ \frac{1}{mn|E(G)|}\sum_{i=0}^{m-1}[2-(\delta_{i,0}+\delta_{i,m-1})]\times\sum_{j=0}^{n-1}\Big[j^2+j(1-2n)+n^2-n\Big]$$
$$+ \frac{2}{n|E(G)|}\sum_{j=0}^{n-1} j$$

Now, let us simplify all of the summations a little bit further. For the first summation, we have:

$$\sum_{j=0}^{n-1}[2-(\delta_{j,0}+\delta_{j,n-1})] = 2n-2$$

For the second summation, we have:

$$\sum_{i=0}^{m-1}[2i^2+i(2-2m)+m^2-m]$$
$$= 2\frac{(m-1)m(2(m-1)+1)}{6} + \frac{(2m^2-2m+(m-1)(2-2m))m}{2}$$
$$= \frac{2m^3-3m^2+m}{3} + m^2 - m = \frac{2m^3-2m}{3}$$

For the third summation, we have:

$$\sum_{i=0}^{m-1} [2 - (\delta_{i,0} + \delta_{i,m-1})] = 2m - 2$$

For the fourth summation, we have:

$$\begin{aligned}
&\sum_{j=0}^{n-1} \Big[j^2 + j(1-2n) + n^2 - n\Big] = \frac{2n^3 - 3n^2 + n}{6} + \frac{n(2n^2 - 2n + (n-1)(1-2n))}{2} \\
&= \frac{2n^3 - 3n^2 + n}{6} + \frac{n^2 - n}{2} = \frac{2n^3 - 2n}{6} = \frac{n^3 - n}{3}
\end{aligned}$$

And lastly, for the fifth summation, we have:

$$\sum_{j=0}^{n-1} j = \frac{n(n-1)}{2} = \frac{n^2 - n}{2}$$

Replacing the answers back into the original equation, we would get:

$$\begin{aligned}
&\text{AvgS\_CDBFS}_T = \frac{|E(G)| - m - n + 2}{|E(G)|} + \frac{1}{mn|E(G)|}(2n-2)\frac{2m^3 - 2m}{3} \\
&+ \frac{1}{mn|E(G)|}(2m-2)\frac{n^3 - n}{3} + \frac{2}{n|E(G)|}\frac{n^2 - n}{2} \\
&= 1 + \frac{1-m}{|E(G)|} + \frac{4}{3}\frac{(1-\frac{1}{n})(m^2-1)}{|E(G)|} + \frac{2}{3}\frac{(1-\frac{1}{m})(n^2-1)}{|E(G)|} \leq 1 + \frac{4}{3}\frac{m^2}{|E(G)|} + \frac{2}{3}\frac{n^2}{|E(G)|}
\end{aligned} \tag{3.12}$$

Since $|E(G)| = m(n-1) + n(m-1) > 2(m-1)(n-1)$,

$$\frac{n^2}{|E(G)|} < \frac{(n-1)^2 + 2n - 1}{2(m-1)(n-1)} = \frac{n-1}{2(m-1)} + \frac{2(n-1)+1}{2(m-1)(n-1)}$$

and

$$\frac{m^2}{|E(G)|} < \frac{(m-1)^2 + 2m - 1}{2(m-1)(n-1)} = \frac{m-1}{2(n-1)} + \frac{2(m-1)+1}{2(m-1)(n-1)}$$

So we have:

$$\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \text{AvgS\_CDBFS}_{i,j} \leq 1 + \frac{2(m-1)}{3(n-1)} + \frac{n-1}{3(m-1)} + O(\frac{1}{m}) + O(\frac{1}{n})$$

As a result, we prove the following theorem:

Theorem 3.2 The average of average sensitivity over all starting nodes of the CDBFS algorithm is at most $1 + \frac{2(m-1)}{3(n-1)} + \frac{n-1}{3(m-1)} + O(\frac{1}{m}) + O(\frac{1}{n})$.

3.2 Average Sensitivity of Reading DBFS on Grids

Similar to the previous chapter, we will first examine the BFS tree generated by this algorithm on both the original and the damaged grids. Following this, we will calculate their Hamming distance leading us to an answer for the average sensitivity of this algorithm.

3.2.1 The Reading DBFS Tree on Grids

When we run the RDBFS algorithm on the original grid, we only select the horizontal edges of R_1 and R_2 and the vertical edges of R_3 and R_4. Figure 3.15 illustrates an example of an RDBFS tree on a grid.

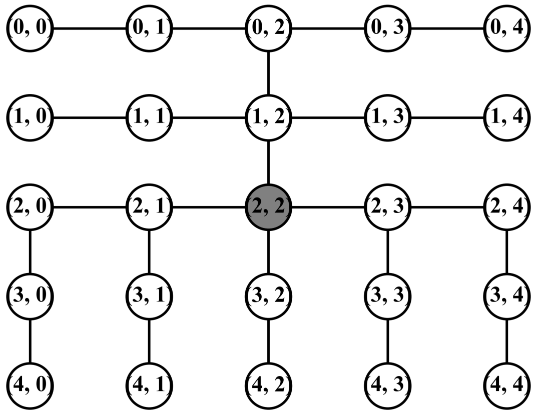

Figure 3.15: The Reading DBFS tree on a 5×5 grid
The starting node in this example is (2,2)

3.2.2 The Reading DBFS Tree on Damaged Grids

Similarly to the previous section, we remove an edge from the grid graph and calculate the corresponding Hamming distance.

(1) Removing an Edge from B_{12}

Similar to the previous ordering, there would be 4i edges that differ between the two BFS trees. So we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(i - d) \tag{3.13}$$

And now we calculate the summation of the Hamming distance over all the edges that can be removed from this section. We will later use this value to calculate the average sensitivity. Therefore the summation over d would be:

$$\sum_{e \in B_{12}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = \sum_{d=0}^{i-1} (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(i - d)$$
$$= (4 - 2(\delta_{j,0} + \delta_{j,n-1})) \sum_{d=0}^{i-1} (i - d) = (4 - 2(\delta_{j,0} + \delta_{j,n-1})) \frac{i(i+1)}{2} \tag{3.14}$$
$$= (2 - (\delta_{j,0} + \delta_{j,n-1})) i(i+1)$$

Figures 3.16, 3.17 and 3.18 illustrate different cases of removing an edge from this section.

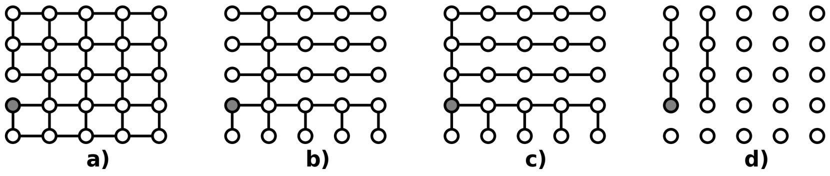

Figure 3.16: The $j = 0$ case of a damaged(from B_{12}) grid in RDBFS

The starting node in all of the following 4 graphs is (3,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

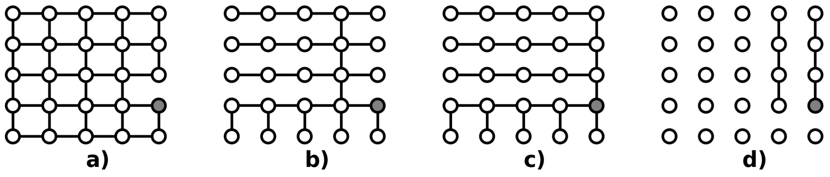

Figure 3.17: The $j = n - 1$ case of a damaged(from B_{12}) grid in RDBFS

The starting node in all of the following 4 graphs is (3,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

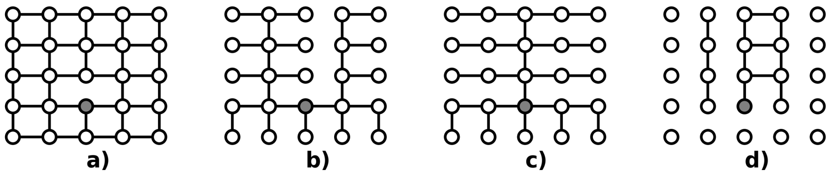

Figure 3.18: The general case of a Damaged(from B_{12}) Grid in RDBFS
The starting node in all of the following 4 graphs is (3,2)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.
Figure b) illustrates a RDBFS tree on the graph a).
Figure c) shows a RDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

(2) Removing an Edge from B_{23}

In this case, the difference would be only 4j edges in contrast to the previous ordering where it was 2. Therefore, we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{i,0} + \delta_{i,m-1}))(j - d)$$

And now we calculate the summation of the Hamming distance over all the edges that can be removed from this section. We will later use this value to calculate the average sensitivity. Therefore the summation over d would be:

$$\begin{aligned}
&\sum_{e \in B_{23}} d_{\mathrm{Ham}}(T(G), T(G \setminus \{e\})) = \sum_{d=0}^{j-1} (4 - 2(\delta_{i,0} + \delta_{i,m-1}))(j - d) \\
&= (4 - 2(\delta_{i,0} + \delta_{i,m-1})) \sum_{d=0}^{j-1} (j - d) = (4 - 2(\delta_{i,0} + \delta_{i,m-1})) \frac{j(j+1)}{2} \\
&= (2 - (\delta_{i,0} + \delta_{i,m-1})) j(j+1)
\end{aligned} \tag{3.15}$$

Figures 3.19, 3.20 and 3.21 illustrate different cases of removing an edge from this section.

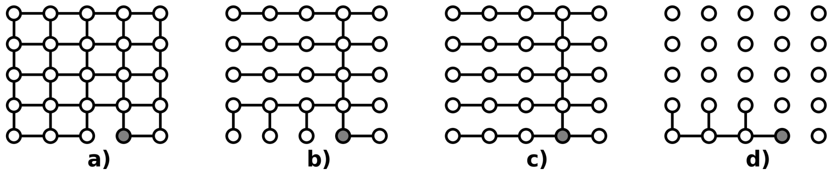

Figure 3.19: The $i = m - 1$ case of a damaged(from B_{23}) grid in RDBFS
The starting node in all of the following 4 graphs is (4,3)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.
Figure b) illustrates a RDBFS tree on the graph a).
Figure c) shows a RDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

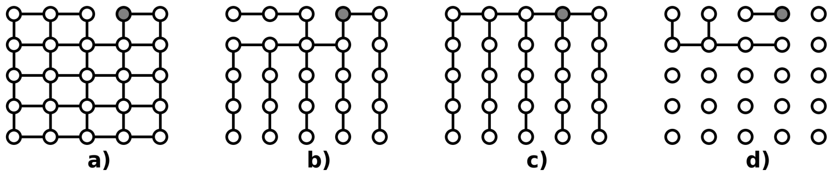

Figure 3.20: The $i = 0$ case of a damaged(from B_{23}) grid in RDBFS
The starting node in all of the following 4 graphs is (0,3)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.
Figure b) illustrates a RDBFS tree on the graph a).
Figure c) shows a RDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

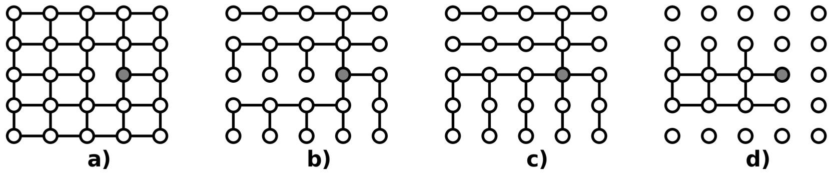

Figure 3.21: The general case of a damaged(from B_{23}) grid in RDBFS

The starting node in all of the following 4 graphs is (2,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(3) Removing an Edge from B_{34}

In this case, the difference would be only 2 in contrast to the previous case when it was $4(m-i)$. But similar to Equation 3.4 we have:

$$\sum_{e \in B_{34}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = 2(m-1-i) \tag{3.16}$$

Figures 3.22, 3.23 and 3.24 illustrate different cases of removing an edge from this section.

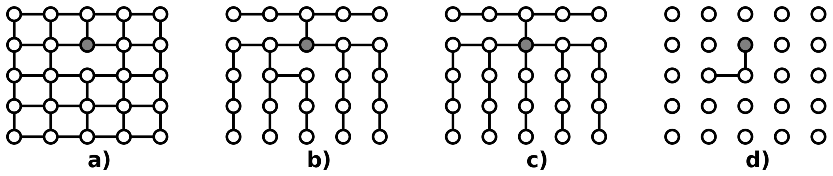

Figure 3.22: The general case of a damaged(from B_{34}) grid in RDBFS

The starting node in all of the following 4 graphs is (1,2)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

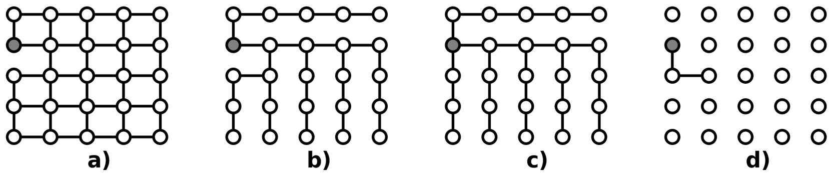

Figure 3.23: The $j = 0$ case of a damaged(from B_{34}) grid in RDBFS

The starting node in all of the following 4 graphs is (1,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

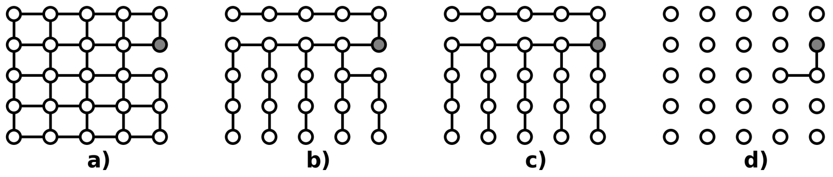

Figure 3.24: The $j = n - 1$ case of a damaged(from B_{34}) grid in RDBFS

The starting node in all of the following 4 graphs is (1,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(4) Removing an Edge from B_{41}

This case would be exactly similar to the previous quadrants and the number of edges in their Hamming distance would be $4(n - i)$. So for this case, we can basically rewrite Equation 3.8. That is:

$$\sum_{e\in B_{41}} d_{\mathrm{Ham}}(T(G),T(G\setminus\{e\})) = \sum_{d''=0}^{n-j-2}(4-2(\delta_{i,0}+\delta_{i,m-1}))(n-j-d''-1)$$
$$= (4-2(\delta_{i,0}+\delta_{i,m-1}))\sum_{d''=0}^{n-j-2}(n-j-d''-1)$$
$$= (4-2(\delta_{i,0}+\delta_{i,m-1}))\frac{(n-j)(n-j-1)}{2} = (2-(\delta_{i,0}+\delta_{i,m-1}))(n-j)(n-j-1) \tag{3.17}$$

Figures 3.25, 3.26 and 3.27 illustrate different cases of removing an edge from this section.

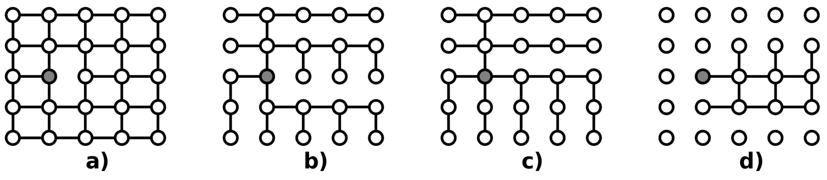

Figure 3.25: The general case of a damaged(from B_{41}) grid in RDBFS

The starting node in all of the following 4 graphs is (2,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

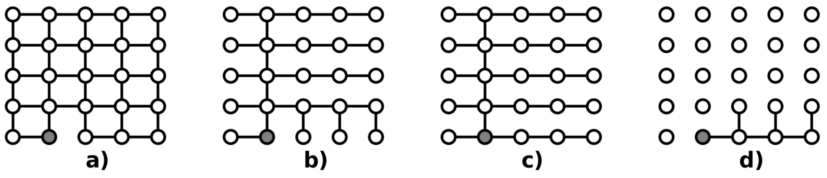

Figure 3.26: The $i = m - 1$ case of a damaged(from B_{41}) grid in RDBFS

The starting node in all of the following 4 graphs is (4,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} to the starting node is removed.

Figure b) illustrates a RDBFS tree on the graph a).

Figure c) shows a RDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

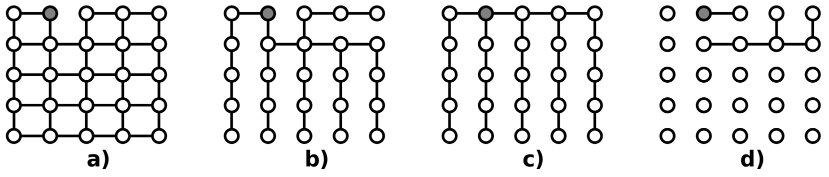
Figure 3.27: The $i = 0$ case of a damaged(from B_{41}) grid in RDBFS
The starting node in all of the following 4 graphs is (0,1)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} to the starting node is removed.
Figure b) illustrates a RDBFS tree on the graph a).
Figure c) shows a RDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

(5) Removing an Edge from $\tilde{B}$

Again this case would be similar and on average we would have a distance of only one edge. This is because removing one of the edges that are selected in the BFS tree on the original graph would yield a 2-edge difference and removing other edges would not change anything. So we have:

$$\sum_{e \in \tilde{B}} d_{\text{Ham}}(T(G), T(G \setminus \{e\})) = 2(m-1)(n-1) \tag{3.18}$$

Figure 3.28 illustrates the case of removing an edge from this section.

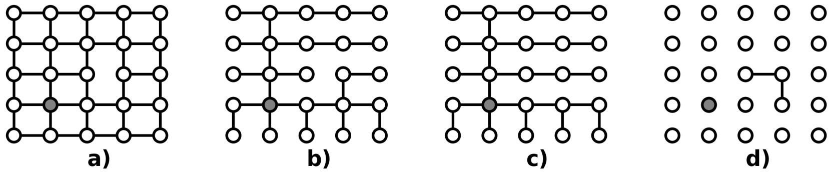
Figure 3.28: The damaged(from $\tilde{B}$) grid in RDBFS
The starting node in all of the following 4 graphs is (3,1)
From left to right: Figure a) represents a damaged grid where one of the edges in $\tilde{B}$ is removed.
Figure b) illustrates a RDBFS tree on the graph a).
Figure c) shows a RDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

3.2.3 Calculating an Upper Bound for the Average Sensitivity of the Reading DBFS

The average sensitivity of RDBFS is exactly equal to that of CDBFS except for when we remove an edge from B_{23} and B_{34}. However, the average sensitivity when we remove an edge from B_{23} in the RDBFS is similar to when we remove an edge from B_{34} in CDBFS and vice versa. Because of that, in terms of average sensitivity, we can assume RDBFS with the top-left-right-bottom ordering is just the same as CDBFS with left-top-right-bottom ordering. So we get the following theorems:

Theorem 3.3 The Reading DBFS algorithm on $m \times n$ grids has the average sensitivity of at most $1 + \frac{n-1}{m-1} + \frac{m-1}{n-1}$, and $O(1)$ for $m = \Theta(n)$ and at most 3 for $m = n$ for any starting node.

Theorem 3.4 The average of average sensitivity over all starting nodes of the RDBFS algorithm is at most $1 + \frac{2(n-1)}{3(m-1)} + \frac{m-1}{3(n-1)} + O(\frac{1}{m}) + O(\frac{1}{n})$.

3.3 Average Sensitivity of Sign of the Cross DBFS On Grids

We first see how the output of the Sign of the Cross DBFS algorithm would look like on both the original grid and the damaged grid and then we will calculate their Hamming distance leading us to an answer for the average sensitivity of this algorithm.

3.3.1 The Sign of the Cross DBFS Tree on Grids

If we run the DBFS algorithm on the original grid, then except for the borders of quadrants(only the vertical border), all the horizontal edges would be selected. Figure 3.29 illustrates an example of an SDBFS tree on a grid.

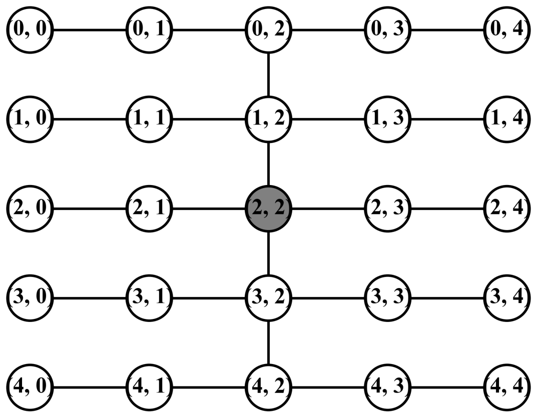

Figure 3.29: The sign of the Cross DBFS tree on a 5 × 5 grid
The starting node in this example is (2,2)

3.3.2 The Sign of the Cross DBFS Tree on Damaged Grids

We now discuss what happens to the BFS tree if we remove an edge from the grid graph. The edge either belongs to the borders set(B) or the $\tilde{B}$ set.
Depending on how close the removed edge is to the starting node, the Hamming distance between the BFS tree on the damaged grid and the original grid varies.

(1) Removing an Edge from B_{12}

This case remains the same compared to the two previous cases and the distance would be 4i. Therefore in general we would have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(i - d)$$

Where d is from 0 to $i - 1$. Figures 3.30, 3.31 and 3.32 illustrate different cases of removing an edge from this section.

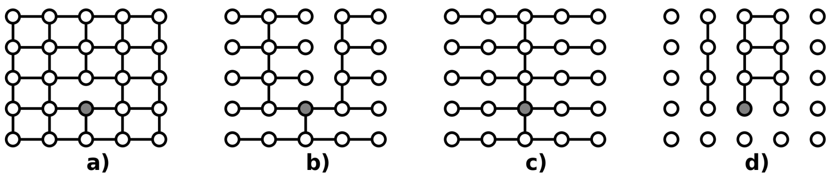

Figure 3.30: The general case of a damaged(from B_{12}) grid in SDBFS
The starting node in all of the following 4 graphs is (3,2)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} *to the starting node is removed.*
Figure b) illustrates an SDBFS tree on the graph a).
Figure c) shows an SDBFS tree on a 5×5 *grid.*
Figure d) depicts the Hamming distance between graphs b) and c).

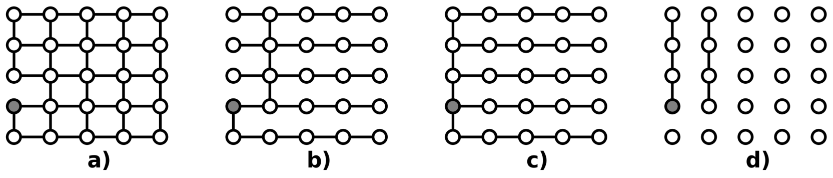

Figure 3.31: The $j = 0$ case of a damaged(from B_{12}) grid in SDBFS

The starting node in all of the following 4 graphs is (3,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

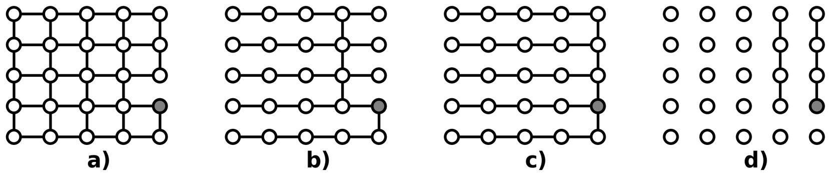

Figure 3.32: The $j = n - 1$ case of a damaged(from B_{12}) grid in SDBFS

The starting node in all of the following 4 graphs is (3,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{12} to the starting node is removed.

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(2) Removing an Edge from B_{23}

The difference would be only 2 edges similar to the first ordering. And it would be zero when the starting node is on the left-most column. Therefore we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = 2 - 2(\delta_{j,0})$$

Figures 3.33, 3.34 and 3.35 illustrate different cases of removing an edge from this section.

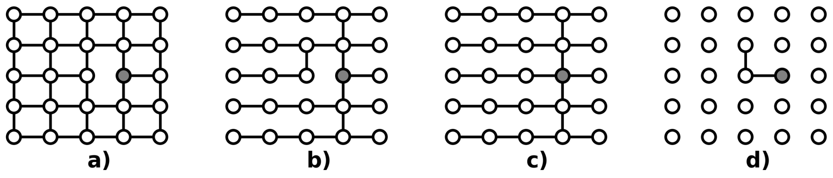

Figure 3.33: The general case of a damaged(from B_{23}) grid in SDBFS

The starting node in all of the following 4 graphs is (2,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} *to the starting node is removed.*

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

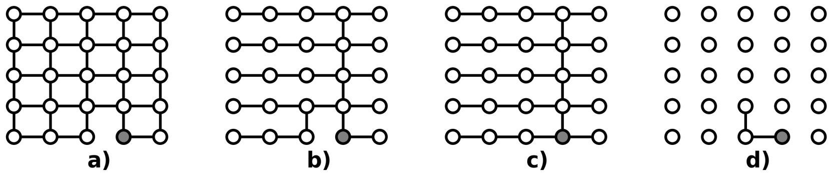

Figure 3.34: The $i = m - 1$ case of a damaged(from B_{23}) grid in SDBFS

The starting node in all of the following 4 graphs is (4,3)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} *to the starting node is removed.*

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

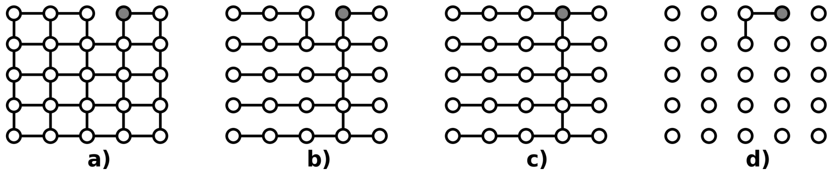

Figure 3.35: The $i = 0$ case of a damaged(from B_{23}) grid in SDBFS
The starting node in all of the following 4 graphs is (0,3)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{23} to the starting node is removed.
Figure b) illustrates an SDBFS tree on the graph a).
Figure c) shows an SDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

(3) Removing an Edge from B_{34}

In this case, the difference is $4(m - i)$ which is similar to the first ordering. So in general we would have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = (4 - 2(\delta_{j,0} + \delta_{j,n-1}))(m - i - 1 - d')$$
$$\text{Where } 0 \le d' \le \max(m - i - 2, 0)$$

Figures 3.36, 3.37 and 3.38 illustrate different cases of removing an edge from this section.

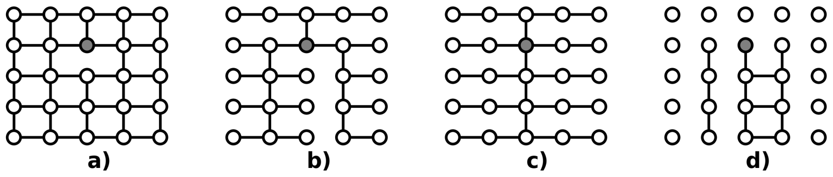

Figure 3.36: The general case of a damaged(from B_{34}) grid in SDBFS
The starting node in all of the following 4 graphs is (1,2)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.
Figure b) illustrates an SDBFS tree on the graph a).
Figure c) shows an SDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

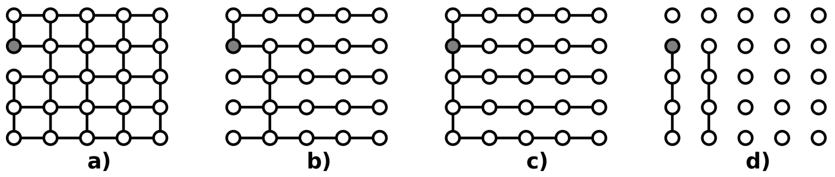

Figure 3.37: The $j = 0$ case of a damaged(from B_{34}) grid in SDBFS

The starting node in all of the following 4 graphs is (1,0)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

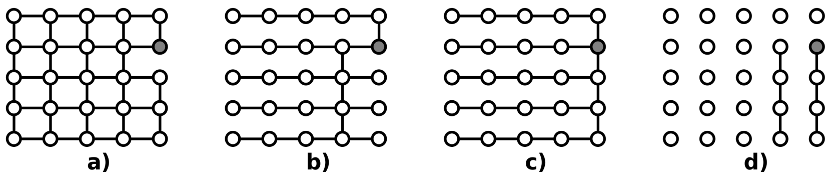

Figure 3.38: The $j = n - 1$ case of a damaged(from B_{34}) grid in SDBFS

The starting node in all of the following 4 graphs is (1,4)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{34} to the starting node is removed.

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 grid.

Figure d) depicts the Hamming distance between graphs b) and c).

(4) Removing an Edge from B_{41}

In contrast to the two previous cases, the difference now is only 2 edges. Obviously, when the starting node is on the right-most column, the difference would be 0. Therefore, in general, we have:

$$d_{Ham}(BFS(G), BFS(G - \{e_1\})) = 2 - 2(\delta_{j,n-1})$$

Figures 3.39, 3.40 and 3.41 illustrate different cases of removing an edge from this section.

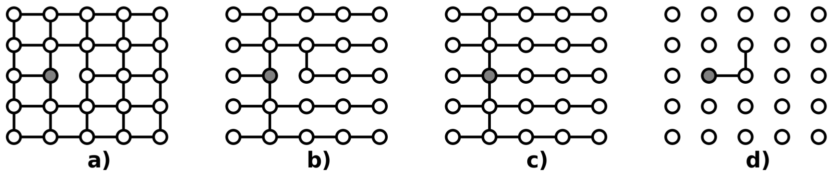

Figure 3.39: The general case of a damaged(from B_{41}) grid in SDBFS

The starting node in all of the following 4 graphs is (2,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} *to the starting node is removed.*

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

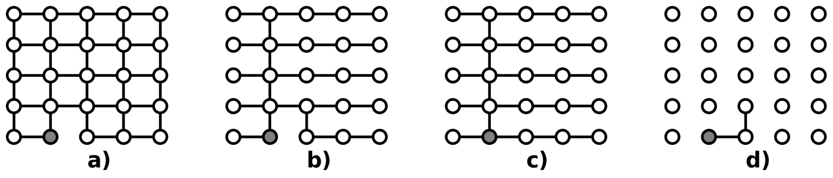

Figure 3.40: The $i = m - 1$ case of a damaged(from B_{41}) grid in SDBFS

The starting node in all of the following 4 graphs is (4,1)

From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} *to the starting node is removed.*

Figure b) illustrates an SDBFS tree on the graph a).

Figure c) shows an SDBFS tree on a 5×5 *grid.*

Figure d) depicts the Hamming distance between graphs b) and c).

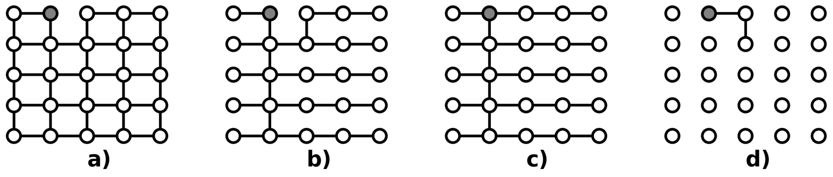

Figure 3.41: The $i = 0$ case of a damaged(from B_{41}) grid in SDBFS
The starting node in all of the following 4 graphs is (0,1)
From left to right: Figure a) represents a damaged grid where the closest edge in B_{41} to the starting node is removed.
Figure b) illustrates an SDBFS tree on the graph a).
Figure c) shows an SDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

(5) Removing an Edge from $\tilde{B}$

Similar to the previous cases, on average the difference would be only one edge. Figure 3.42 illustrates the case of removing an edge from this section.

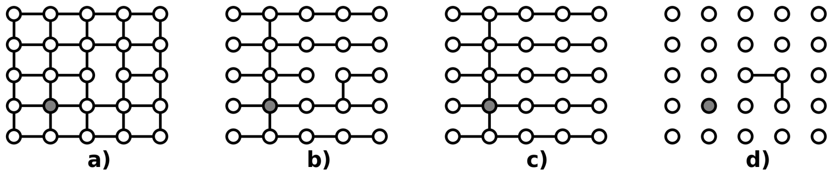

Figure 3.42: The damaged(from $\tilde{B}$) grid in SDBFS
The starting node in all of the following 4 graphs is (3,1)
From left to right: Figure a) represents a damaged grid where one of the edges in $\tilde{B}$ is removed.
Figure b) illustrates an SDBFS tree on the graph a).
Figure c) shows an SDBFS tree on a 5×5 grid.
Figure d) depicts the Hamming distance between graphs b) and c).

3.3.3 Calculating an Upper Bound for the Average Sensitivity of the Sign of the Cross DBFS

The average sensitivity of the SDBFS is similar to that of CDBFS except for when we remove an edge from B_{41}. And because for this section, the average sensitivity of SDBFS is smaller than CDBFS, in total, SDBFS has a lower average sensitivity.

Now to calculate the average sensitivity of SDBFS, by replacing the Equations 3.14 3.15 3.16 3.17 3.18 to Equation 3.10 we would get:

$$
\begin{aligned}
\sum_{e\in E(G)} d_{\text{Ham}}(T(G), T(G\setminus\{e\})) = {} & (2-(\delta_{j,0}+\delta_{j,n-1}))i(i+1) \\
& + 2j \\
& + (2-(\delta_{j,0}+\delta_{j,n-1}))(m-i)(m-i-1) \\
& + 2(n-j-1) \\
& + 2(m-1)(n-1)
\end{aligned}
$$

Substituting everything back to the definition of average sensitivity(2.3) we get:

$$
\begin{aligned}
\text{AvgS\_SDBFS}_{i,j} &= \frac{\sum_{e\in E(G)} d_{\text{Ham}}(T(G), T(G\setminus\{e\}))}{|E(G)|} \\
&= \frac{(2-(\delta_{j,0}+\delta_{j,n-1}))(i(i+1)+(m-i)(m-i-1))}{|E(G)|} \\
&+ \frac{2(n-1)}{|E(G)|} \\
&+ \frac{|E(G)|-m-n+2}{|E(G)|} \\
&= 1 + \frac{n-m}{|E(G)|} + \frac{(2-(\delta_{j,0}+\delta_{j,n-1}))(2i^2+2i+m^2-2mi-m)}{|E(G)|}
\end{aligned} \tag{3.19}
$$

Now Let us calculate an upper bound for this average sensitivity. Since $2i(i+1-m) \le 0$ and $-(\delta_{j,0}+\delta_{j,n-1}) \le 0$, we have:

$$\text{AvgS\_SDBFS}_{i,j} \le 1 + \frac{n-m}{|E(G)|} + \frac{2(m^2-m)}{|E(G)|}$$

We can assume $m \le n$(if not, we can rotate the grid first and then run the BFS algorithm). Now we prove that this average sensitivity is at most 2 for any starting node:

$$
\begin{aligned}
& 1 \le m \Leftrightarrow n-m \le m(n-m) \Leftrightarrow n-m+m^2 \Leftrightarrow mn \Leftrightarrow 2n-2m+2m^2 \le 2mn \Leftrightarrow \\
& n-3m+2m^2 \le 2mn-m-n = |E(G)| \Leftrightarrow \frac{n-m}{|E(G)|} + \frac{2m(m-1)}{|E(G)|} \le 1
\end{aligned}
$$

Therefore:

$$\text{AvgS\_SDBFS}_{i,j} \le 2$$

Theorem 3.5 The Sign of the Cross DBFS algorithm on $m\times n$ grids with $2 \le m \le n$ has average sensitivity at most 2 for any starting node.

Corollary 3.1 There is a deterministic BFS algorithm with average sensitivity at most 2 for any starting node in an $m \times n$ grid with $m \geq 2$ and $n \geq 2$.

3.3.4 Average of Average Sensitivity over All Starting Nodes for SDBFS

Now let us analyze the average of average sensitivity over all nodes of a grid as a starting node for the Sign of the Cross DBFS algorithm. Before we do that, Let us rewrite the average sensitivity in this ordering with respect to the coordinates of the starting node. Therefore we have:

$$\begin{aligned}
&\text{AvgS\_SDBFS}_{i,j} = \frac{1}{|E(G)|}\{1 \times 2(m-1)(n-1) + 2(n-1) + \\
&(4 - 2(\delta_{j,0} + \delta_{j,n-1}))\sum_{k=0}^{i} k + (4 - 2(\delta_{j,0} + \delta_{j,n-1}))\sum_{k=0}^{m-i-1} k]\} \\
&= \frac{1}{|E(G)|}\{2(m)(n-1) + [4 - 2(\delta_{j,0} + \delta_{j,n-1})][\frac{i(i+1)}{2} + \frac{(m-i-1)(m-i)}{2}]\} \\
&= \frac{2m(n-1)}{|E(G)|} + \frac{1}{|E(G)|}[4 - 2(\delta_{j,0} + \delta_{j,n-1})]\left[\frac{i(i+1)}{2} + \frac{m^2}{2} - \frac{m}{2} - im\right] \\
&= \frac{2m(n-1)}{|E(G)|} + \frac{1}{|E(G)|}[4 - 2(\delta_{j,0} + \delta_{j,n-1})]\left[\frac{m^2}{2} - \frac{m}{2} + i^2 + i(1-m)\right]
\end{aligned}$$

Now that we have calculated the general form, we want to get rid of i and j. So now we take an average over all the possible starting nodes.

$$\text{AvgS\_SDBFS}_T = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \text{AvgS\_SDBFS}_{i,j}$$

$$\begin{aligned}
&= \frac{2m(n-1)}{m(n-1) + n(m-1)} + \frac{1}{mn[m(n-1) + n(m-1)]}\sum_{j=0}^{n-1}[4 - 2(\delta_{j,0} + \delta_{j,n-1})] \\
&\times \sum_{i=0}^{m-1}\left[\frac{m^2}{2} - \frac{m}{2} + i^2 + i(1-m)\right]
\end{aligned}$$

Now, let us simplify the second summation a little bit further:

$$\sum_{i=0}^{m-1}\left[\frac{m^2}{2}-\frac{m}{2}+i^2+i(1-m)\right]=\left[\frac{m^3}{2}-\frac{m^2}{2}+\frac{(m-1)m(2m-1)}{6}+\frac{(m-1)m(1-m)}{2}\right]$$
$$=\left[\frac{m^3}{2}-\frac{m^2}{2}+\frac{(m^2-m)(2m-1)}{6}-\frac{(m-1)^2m}{2}\right]$$
$$=\left[\frac{m^3}{2}-\frac{m^2}{2}+\frac{2m^3-m^2-2m^2+m}{6}-\frac{m^3-2m^2+m}{2}\right]$$
$$=\left[\frac{m^3}{2}-\frac{m^2}{2}+\frac{m^3}{3}-\frac{m^2}{2}+\frac{m}{6}-\frac{m^3}{2}+m^2-\frac{m}{2}\right]$$
$$=\left[\frac{m^3}{3}-\frac{m}{3}\right]$$

Replacing the answer back into the original equation, we would get:

$$\text{AvgS\_SDBFS}_T=\frac{2m(n-1)}{m(n-1)+n(m-1)}+\frac{1}{mn[m(n-1)+n(m-1)]}\sum_{j=0}^{n-1}[4-2(\delta_{j,0}+\delta_{j,n-1})]$$
$$\times\left[\frac{m^3}{3}-\frac{m}{3}\right]$$
$$=\frac{2m(n-1)}{m(n-1)+n(m-1)}+\frac{1}{mn[m(n-1)+n(m-1)]}[4n-4]\left[\frac{m^3}{3}-\frac{m}{3}\right]$$
$$=\frac{2m(n-1)}{m(n-1)+n(m-1)}+\frac{4(n-1)(m^3-m)}{3mn[m(n-1)+n(m-1)]}$$

Since $m(n-1)+n(m-1)>m(n-1)+(n-1)(m-1)=(n-1)(2m-1)$,

$$\frac{2m(n-1)}{m(n-1)+n(m-1)}<\frac{2m(n-1)}{(n-1)(2m-1)}=\frac{2m}{2m-1}=1+\frac{1}{2m-1}$$

Since $m(n-1)+n(m-1)>2(m-1)(n-1)$,

$$\frac{4(n-1)(m^3-m)}{3mn[m(n-1)+n(m-1)]}<\frac{4(n-1)m(m^2-1)}{6mn(m-1)(n-1)}=\frac{2(m^2-1)}{3n(m-1)}=\frac{2(m-1)(m+1)}{3n(m-1)}$$
$$=\frac{2(m+1)}{3n}=\frac{2m}{3n}+\frac{2}{3n}$$

Hence for $m\leq n$:

$$\text{AvgS\_SDBFS}_T=\frac{1}{mn}\sum_{i=0}^{m-1}\leq 1+\frac{2}{3}+O(\frac{1}{m})+O(\frac{1}{n})=\frac{5}{3}+O(\frac{1}{m})+O(\frac{1}{n}).\qquad(3.20)$$

Theorem 3.6 The average of average sensitivity over all starting nodes of the SDBFS algorithm is at most $\frac{5}{3} + O(\frac{1}{m}) + O(\frac{1}{n})$.

Chapter 4

Average Sensitivity of RBFS

As mentioned in section 2.3, we can represent the Earth mover's distance in the average sensitivity of randomized graph algorithms using a min-cost max-flow problem by definition. This can help visualize the parameters and give a more understandable approach to obtain the results. This is why in this section we first explain how this representation is done, and then try to calculate the average sensitivity of the RBFS algorithm using this approach.

From Equation 2.1, the Earth mover's distance is:

$$EM(A(G), A(G')) = \min \sum_{B_j \in A(G)} \sum_{B'_l \in A(G')} x_{jl} d_{\text{Ham}}(B_j, B'_l) \tag{4.1}$$

subject to $x_{jl} \geq 0$, and for every j, $\sum_l x_{jl} = p_j$, and for every l, $\sum_j x_{jl} \leq q_l$.

For a randomized algorithm we know that the summation of the probabilities of the solutions must add up to 1 That is why we have

$$\sum_j p_j = 1 \quad and \quad \sum_l q_l = 1$$

So now, we represent each B_j and B'_l by a node and also define nodes s(representing a source node) and t(representing a target node). We now add edges from s to each B_j with a capacity and lower bound of p_j and a weight of 0 and from B'_l to t with a capacity and lower bound of q_l(although having the lower bound of q_l is redundant since it will be satisfied regardless of having this) and a weight(cost) of 0. We also add edges from B_j to B'_l with a weight of their Hamming distance i.e. $d_{\text{Ham}}(B_j, B'_l)$ and a capacity of 1(the capacity could be any value larger than p_j and the result would be the same).

Now, the min-cost max-flow on this flow network would represent the desired Earth mover's distance. This is because the flow over each of the edges between B_j and B'_l represents their corresponding x_{jl} and therefore the condition that $x_{jl} \geq 0$ is met since the flow cannot be negative. Also, the lower bounds on the incoming edge of B_j and on the outgoing edge of B'_l ensure that the condition of $\sum_l x_{jl} = p_j$, and $\sum_j x_{jl} \leq q_l$ are satisfied respectively. Figure 4.1 represents the flow network suitable for finding the Earth mover's distance.

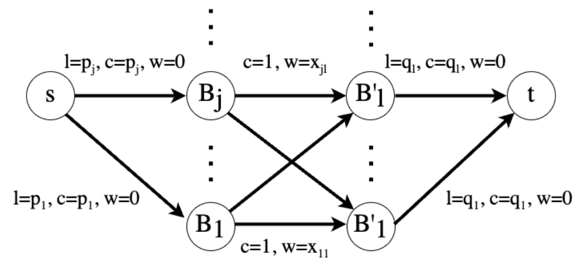

Figure 4.1: The flow network representing the Earth mover's distance
l represents the lower bounds, c represents the capacities, and w represents the weights of the edges

It is important to note, that to calculate the average sensitivity of the algorithm we need to take the average of the Earth mover's distances for all the removed edges.

4.1 The RBFS Tree on Grids

When running the RBFS on grids, all the edges in set B will be selected with a probability 100% and the edges that are not in set B, i.e. edges in $\tilde{B}$ set, will be selected with a probability of 50%. This is because each node in the $\tilde{B}$ set is the neighbour of two nodes that are in the previous layer, however, nodes in set B, have only one node that is in the previous layer. Because of that, we have $2^{(m-1)(n-1)}$ RBFS trees on the grid graph($(m-1)(n-1)$ is the number of nodes in $\tilde{B}$). Therefore, the probability of outputting each of the RBFS trees would be $2^{-(m-1)(n-1)}$. Figure 4.2 illustrates how such BFS trees are constructed.

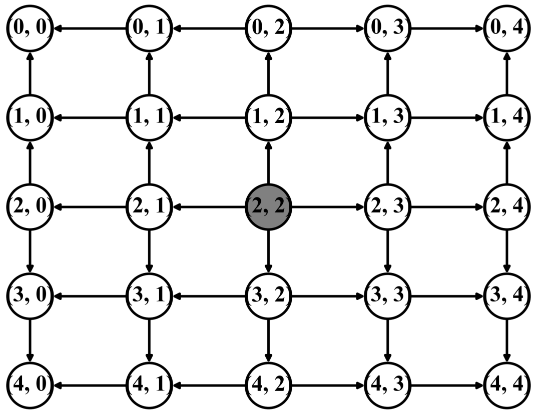

Figure 4.2: Illustration of potential RBFS trees on a 5×5 grid
In this figure the starting node is $(2, 2)$*. Since a* 5×5 *grid is a connected graph, all the nodes will be explored. The only way to explore a node is using the incoming edges of a node(although the original graph is not directed, because we have layers in the RBFS algorithm, we represented the grid as a directed graph). For nodes in set* B *we only have one incoming edge so they will definitely appear in all the RBFS trees. and for the nodes in set* $\tilde{B}$ *we have two incoming edges, therefore exactly one of them will be selected in the RBFS tree. Because of that we can have* $2^{4\times4}$ *RBFS trees for this grid*

4.2 The RBFS Tree on Damaged Grids

Now let us discuss what happens to the BFS tree if we remove an edge from the grid graph. The edge either belongs to the borders set(B) or the $\tilde{B}$ set.

Depending on where the removed edge is, the Earth mover's distance between the RBFS trees on the damaged grid and the original grid varies.

4.2.1 Removing an Edge from Set $\tilde{B}$

In this case, the Earth mover's distance would always be 1. This is because when we remove an edge from $\tilde{B}$ the farthest endpoint of this edge can only be explored via the other incoming edge as discussed in figure 4.2. Hence, that edge will always appear in all of the RBFS trees. As a result, the number of RBFS trees would be half of the RBFS trees on the original grid.

Now, for every RBFS tree in the damaged grid, we have an identical RBFS tree in the outputs of the RBFS on the original grid, hence their Hamming distance would be 0. So in the min-cost max-flow assignment, the flow would use up all of the capacities of these edges by p_j. But still, half of the capacity of the edges from B'_l to t is not handled. For that, the best assignment is to assign the RBFS trees on the damaged graph to the RBFS trees on the original graph that apart from these two incoming edges to the mentioned node, are exactly the same. So in this case, the Hamming distance would be 2. And since half of the flow will cross over edges with a weight of 2, the Earth mover's distance would be $\frac{1}{2} \times 2 = 1$.

4.2.2 Removing an Edge from Set B

Depending on where the starting node is, the Earth mover's distance would have a different value. We can partition the positions of the starting node into the following four cases: 1) The corners 2) On the first or last row but not the corners 3) On the first or last column but not the corners 4) Not on the grid's perimeter.

1) The Corners

We have four different corners $(i, j) \in \{(0,0), (0, n-1), (m-1, 0), (m-1, n-1)\}$ Because of symmetry, the results for all four corners are the same. To better understand the approach, we start with a 4×5 grid and then we remove the $\{(0,0), ((0,1)\}$ edge. Figure 4.3 shows the possible RBFS trees on the original and the damaged grid.

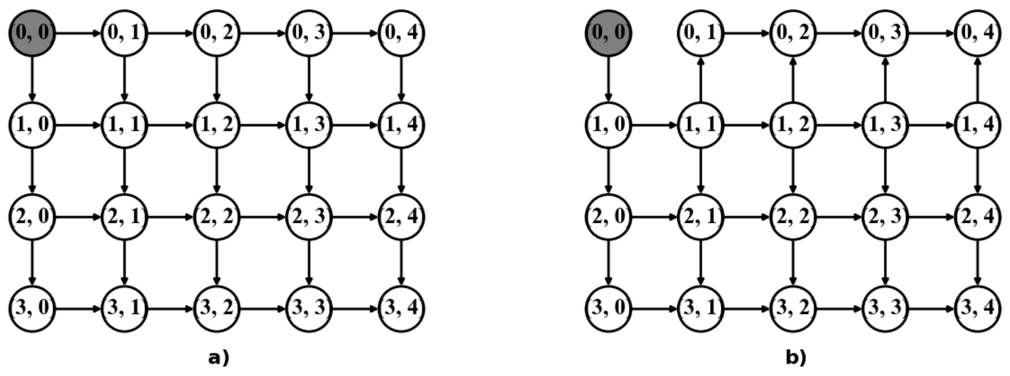

Figure 4.3: Illustration of RBFS trees on the original and damaged grids
Since all nodes need to be explored, exactly one of their incoming edges will be selected. It means that for those nodes having only one incoming edge, that edge will definitely appear in all the RBFS trees

So, as you can see in figure 4.3, when we run the RBFS algorithm on the original grid, all the edges in the first row and column would be in all the possible outputs of the algorithm. However, in the damaged grid, since we don't have the edge $\{(0,0), (0,1)\}$, all the edges in the second row will be selected and also for node $(0,1)$ there is only one incoming edge. Now, this means that the node $(1,1)$ in the original grid will be explored via $(1,0)$ or $(0,1)$. Whichever that is the Hamming distance between these outputs and that of RBFS on the damaged grid for the first square of the grid, will always be 2(one for the removed edge and one for the incoming edge of node $(1, 1)$) For simplicity we merge any two RBFS trees on the original grid if the rest of the edges are the same.

In both the outputs of the damaged grid and the original grid, every node in the third row to the last row, has two incoming edges, meaning there are two possibilities to explore these nodes. So there are as many BFS trees caused by these two areas. The problem now is how to handle the first two rows and the vertical edges between them.

In the first two rows, apart from the first square, we can have 0 to $n-2$ vertical edges in both of our outputs. The number of RBFS trees with r $(0 \le r \le n-1)$ vertical edges between the first two rows is $\binom{n-2}{r}$, each RBFS tree is created with the same probability of $\frac{1}{2^{n-2}}$. The Hamming distance between an RBFS tree with r vertical edges between the first two rows and an RBFS tree on the original grid is $2(r+1)$. Therefore the Earth mover's distance for this case would be:

$$\begin{aligned} EM(A(G), A(G')) &= \frac{1}{2^{n-2}}\left(\binom{n-2}{0}2 + \binom{n-2}{1}4 + \cdots + \binom{n-2}{n-2}2(n-1)\right) \\ &= \frac{1}{2^{n-1}}\left(\binom{n-2}{0}1 + \binom{n-2}{1}2 + \cdots + \binom{n-2}{n-2}(n-1)\right) \end{aligned} \tag{4.2}$$

Now the question is can this formula be simplified? And the answer is yes! It is equal to n. Now we want to prove why: For simplicity, we rewrite $n-2$ with k where $k \ge 0$. Therefore the inner parenthesis is equal to:

$$\binom{k}{0}1 + \binom{k}{1}2 + \cdots + \binom{k}{k}(k+1)$$

To calculate what this term is equal to, consider the expansion of $(x+1)^k$:

$$(x+1)^k = \binom{k}{0}x^0 + \binom{k}{1}x^1 + \cdots + \binom{k}{k}x^k$$

Now multiplying both sides by x we get:

$$x(x+1)^k = \binom{k}{0}x^1 + \binom{k}{1}x^2 + \cdots + \binom{k}{k}x^{k+1}$$

And now taking the derivative of both sides with respect to x we get:

$$(x+1)^k + kx(x+1)^{k-1} = \binom{k}{0}1 + \binom{k}{1}2x^1 + \cdots + \binom{k}{k}(k+1)x^k$$

And finally replacing x with 1 we get:

$$2^k + k2^{k-1} = \binom{k}{0}1 + \binom{k}{1} + \cdots + \binom{k}{k}(k+1) = (k+2)2^{k-1}$$

Replacing the result back into Equation 4.2 we get:

$$EM(A(G), A(G')) = \frac{1}{2^{n-1}}(n2^{n-1}) = n$$

Similarly, if we remove the second edge in the first row, then the Earth mover's distance would be $n-1$. And if we remove the last edge in this section, the Earth mover's distance would be 2. As a result, the average of the Earth mover's distance on the first row would be equal to $\frac{(n+2)(n-1)}{2}$. Similarly, for the first column, we have $\frac{(m+2)(m-1)}{2}$. To conclude, the average sensitivity would be equal to:

$$AvgS = \frac{1}{m(n-1)+n(m-1)}\left[(n-1)\frac{n+2}{2} + (m-1)\frac{(m+2)}{2} + 2(m-1)(n-1)\right] \tag{4.3}$$

We want to have an upper bound for this. Therefore we have:

$$AvgS \leq \frac{(n-1)\frac{n+2}{2} + (m-1)\frac{(m+2)}{2} + 2(m-1)(n-1)}{2(m-1)(n-1)} \leq 1 + \frac{n+2}{4(m-1)} + \frac{m+2}{4(n-1)}$$

2) On the First or Last Row but not the Corners

The result for this section would be the same whether the starting node is on the first or last row. We will assume the starting node is at the first row:

In this case, if we remove an edge from B_{41} then it would be similar to the previous case only as if the first row has $(n-j)$ nodes, and if we remove an edge from B_{23} then it would also be similar to the previous case only as if the first row has $(j+1)$ nodes. Therefore, the summation of the Earth mover's distances of all the edges in this section would be equal to:

$$\frac{(j+3)j}{2} + \frac{(n-j+2)(n-j-1)}{2}$$

Now, analyzing the case where we remove one of the vertical edges that belong to the set B is more difficult. Let us consider the case where we remove the closest edge to the starting node. In this case, the node at the end of the removed edge can only be discovered either through left or right and all vertical edges in column $j-1$ and column $j+1$ are present in the RBFS tree. However, in the outputs of the RBFS algorithm on the original grid, each of these edges appears in an RBFS tree with probability 1/2. Figure 4.4 illustrates the possible RBFS trees of this case.

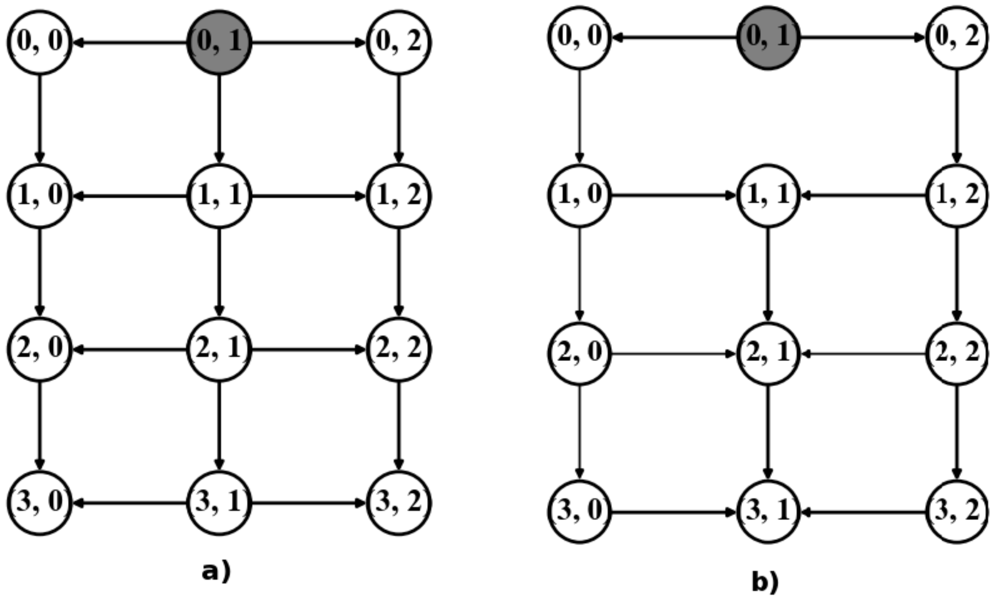

Figure 4.4: Illustration of RBFS trees on the original and damaged grids (starting node at the top)

Each node must be discovered through exactly one of its incoming edges

Since the rest of the grid would be explored in the same way in both of the input graphs, we only need to examine the three columns $j-1$, j, $j+1$. as depicted in figure 4.4. To explore that, we break down this graph into $m-1$ sections of two rows. Each section contains the vertical edges between the two rows and the horizontal edges in the second row of the section. Since node $(1,1)$ has two incoming edges and the rest of the nodes in this column have 3 incoming edges, the analysis for this node is different than the rest. For the first section(first two rows) we can have the following RBFS trees(Figure 4.5):

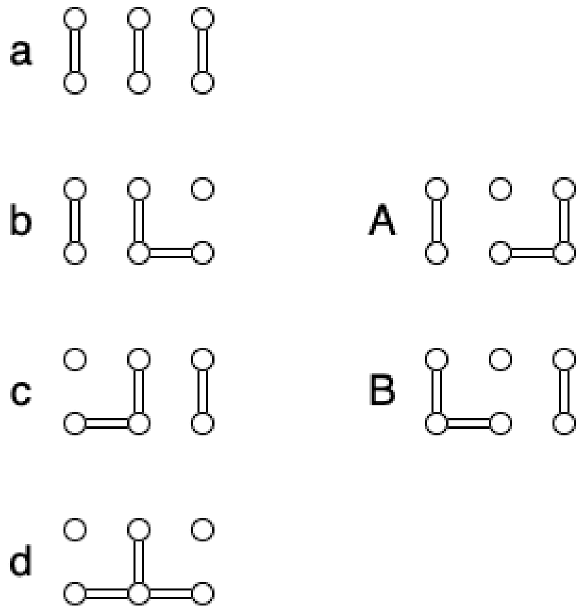

Figure 4.5: The analysis of the first two rows of RBFS(starting node at the top) *subtrees a,b,c,d are all the possible subtrees for the RBFS on the original grid, and subtrees A, and B are the possible subtrees for the damaged grid*

To find the best assignments(with the shortest Earth mover's distance) of the subtrees on the left to the right, we can assign a and b to A and also c and d to B. Notice that each of the subtrees on the left has $\frac{1}{4}$ probability of appearing in an output and the subtrees on the right have a probability of $\frac{1}{2}$. Therefore, the Earth mover's distance for this section would be equal to:

$$\frac{1}{4}2 + \frac{1}{4}2 + \frac{1}{4}2 + \frac{1}{4}4 = \frac{10}{4} = 2.5$$

Now, let us analyze the second section(contains 2nd and 3rd rows). Notice that the rest of the grid follows the exact same pattern as these two rows. So by finding the best assignment for these two rows we basically find the best assignment for the whole grid. Similarly, we have the following subtrees(Figure 4.6):

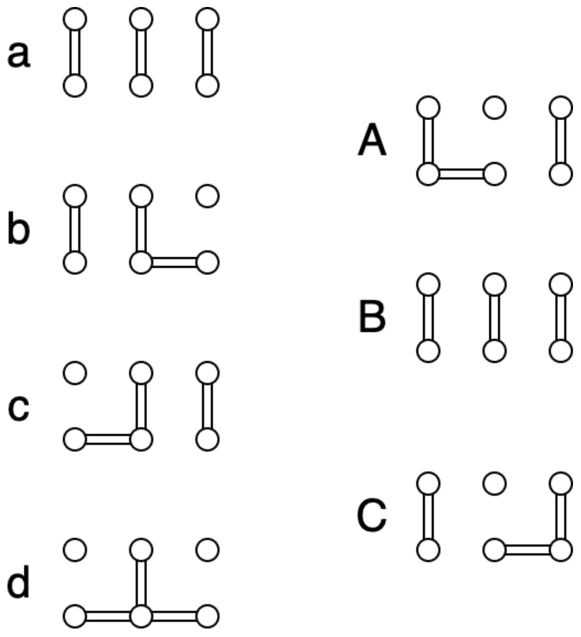

Figure 4.6: The analysis of the second two rows of RBFS(starting node at the top) *subtrees a,b,c and d are all the possible subtrees for the RBFS on the original grid, and subtrees A, B and C are the possible subtrees for the damaged grid*

Here the best assignment is to assign a to B completely since all the elements on the left have a probability of $\frac{1}{4}$ and the elements on the right have a probability of $\frac{1}{3}$ we will still have $\frac{1}{3} - \frac{1}{4} = \frac{1}{12}$ after this assignment for B. Then we assign b to C and c to A and finally d will be assigned to all of A, B and C equally with the rest of their probabilities which would be $\frac{1}{12}$. Therefore, the Earth mover's distance for this section would be equal to:

$$\frac{1}{4}0 + \frac{1}{4}2 + \frac{1}{4}2 + (\frac{1}{12} + \frac{1}{12} + \frac{1}{12})4 = \frac{8}{4} = 2$$

This means that the Earth mover's distance when we remove the first edge would be equal to:

$$EM(A(G), A(G')) = 2.5 + 2(m-2)$$

If we remove the second edge, it's as if the number of rows is $m-1$ and so on. So it means that the summation of Earth mover's distance of all the edges that can be removed from this section is:

$$\sum_{k=1}^{m-1}(2.5 + 2(m-1-k)) = 2.5(m-1) + (m-2)(m-1) = (m-1)(m+0.5)$$

Therefore, the average sensitivity is equal to:

$$AvgS = \frac{1}{m(n-1)+n(m-1)}[\frac{(j+3)j}{2} + \frac{(n-j+2)(n-j-1)}{2} + (m-1)(m+0.5) \\ + 2(m-1)(n-1)] \tag{4.4}$$

We want to have an upper bound for this. And since $(j+3)j - 2nj + j^2 - j \leq 0$ we have:

$$\leq 1 + \frac{n^2+n-2+2m^2-m-1}{4(m-1)(n-1)} \leq 1 + \frac{n+1}{4(m-1)} + \frac{2m+1}{4(n-1)}$$

3) On the First or Last Column but not the Corners

Similar to Equation 4.4 we get:

$$AvgS = \frac{1}{m(n-1)+n(m-1)}[\frac{(i+3)i}{2} + \frac{(m-i+2)(m-i-1)}{2} + (n-1)(n+0.5) \\ + 2(m-1)(n-1)] \tag{4.5}$$

Which is:

$$\leq 1 + \frac{m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$$

4) Not on the Grid's Perimeter

In this case, removing an edge from the row and column that contains the starting node would look like figures 4.5 and 4.6. Therefore the average sensitivity is equal to the following:

$$AvgS = \frac{1}{m(n-1)+n(m-1)}[j(j+1.5) + (n-j-1)(n-j+0.5) + \\ i(i+1.5) + (m-i-1)(m-i+0.5) + 2(m-1)(n-1)]$$

Which is equal to:

$$= \frac{1}{m(n-1)+n(m-1)}[i(i+1) + (m-i)(m-i-1) + 2(m-1)(n-1) + j(j+1) \\ + (n-j)(n-j-1) + \frac{m-1}{2} + \frac{n-1}{2}] \tag{4.6}$$

We want to find an upper bound for this. Since we have $i(i+1) - 2mi + i^2 + i \leq 0$ and $j(j+1) - 2nj + j^2 + j \leq 0$ we get:

$$\leq 1 + \frac{1}{2(m-1)(n-1)}(m(m-1) + n(n-1) + \frac{m-1}{2} + \frac{n-1}{2}) \\ \leq 1 + \frac{2m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$$

4.3 Average Sensitivity of RBFS on an Arbitrary Starting Node

We want to get rid of the i and j indices so we take an average on the starting node. Also, we are looking for an upper bound for this value, so we assume both m and n are large values. Therefore we would have the following:

$$\text{AvgDBFS}_T = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \text{AvgDBFS}_{i,j}$$

Now, since the upper bound that we obtained for the average sensitivity of RBFS when the starting node is not on the grid's premier is larger than all the other cases, we can use that upper bound as the upper bound for any arbitrary starting node. Therefore we have:

$$\text{AvgDBFS}_T \leq 1 + \frac{2n+1}{4(m-1)} + \frac{2m+1}{4(n-1)}$$

Now for the special case of $m = n$ we have:

$$\leq 1 + \frac{2m+1}{2(m-1)} = 2 + \frac{3}{2m-2} \leq 2 + O(\frac{1}{m})$$

Therefore we have the following theorems:

Theorem 3.7 The Randomized BFS algorithm on $m \times n$ grids has the average sensitivity of at most $1 + \frac{2m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$, and $O(1)$ for $m = \Theta(n)$ and at most $2 + O(\frac{1}{m})$ for $m = n$ for any starting node.

Theorem 3.8 The average of average sensitivity over all starting nodes of the RDBFS algorithm is at most $1 + \frac{2m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$.

Chapter 5

The Comparison of SDBFS with RBFS

So far we have obtained upper bounds for the average sensitivity of both deterministic and randomized BFS algorithms for arbitrary starting nodes. Just because the upper bound for the average sensitivity of one algorithm is lower, it doesn't mean that that algorithm has a lower average sensitivity on a specific starting node. So for a more precise analysis, we need to see for specific starting nodes which algorithm has a lower average sensitivity. In this chapter, we will compare the average sensitivity of SDBFS to RBFS for specific starting nodes. We do this because SDBFS has the lowest average sensitivity compared to the other versions of the DBFS that we explained. We also assume $m \leq n$ since this is the case when SDBFS performs best. And even if $m > n$, we can rotate the grid first and then perform the BFS algorithm.

Since the value of average sensitivity has different formulas depending on the position of the starting node, we will compare the average sensitivity of DBFS with RBFS based on this position (i, j). Therefore, similar to our analysis for RBFS the position of the starting node can belong to the following:

1. The corners
2. On the first or last row but not the corners
3. On the first or last column but not the corners
4. Not on the grid's perimeter

1) The Corners

For SDBFS from Equation 3.19 we have the following average sensitivity:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{m-n}{|E(G)|} + \frac{(2-(\delta_{j,0}+\delta_{j,n-1}))(2i^2+2i+m^2-2mi-m)}{|E(G)|}$$

Since for the corners, the sum of Kronecker delta terms will be 1 and also the sum of the terms dependant on i will be 0 we have:

$$\text{AvgS\_SDBFS}_{corners} = 1 + \frac{m-n}{|E(G)|} + \frac{+m^2 - m}{|E(G)|} = 1 + \frac{m^2 - n}{|E(G)|}$$

And for RBFS from Equation 4.3 we have the following average sensitivity:

$$AvgS = \frac{\left[(n-1)\frac{n+2}{2} + (m-1)\frac{(m+2)}{2} + |E(G)| - (m-1) - (n-1)\right]}{|E(G)|}$$
$$AvgS = 1 + \frac{\left[(n-1)\frac{n}{2} + (m-1)\frac{m}{2}\right]}{|E(G)|}$$

And since we assumed $m \leq n$ then, their average sensitivities would be equal for the case that $m = n$ and for the case that $m < n$ the SDBFS has a lower average sensitivity.

2) On the First or Last Row but not the Corners

For SDBFS from Equation 3.19 we have the following average sensitivity:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{m-n}{|E(G)|} + \frac{(2 - (\delta_{j,0} + \delta_{j,n-1}))(2i^2 + 2i + m^2 - 2mi - m)}{|E(G)|}$$

And simplifying for the first or last row but not corner starting nodes we have:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{m-n}{|E(G)|} + \frac{2(m^2 - m)}{|E(G)|} = 1 + \frac{2m^2 - m - n}{|E(G)|}$$

And for RBFS from Equation 4.4 we have:

$$AvgS = \frac{1}{m(n-1) + n(m-1)}\left[\frac{(j+3)j}{2} + \frac{(n-j+2)(n-j-1)}{2} + (m-1)(m+0.5)\right.$$
$$\left. + 2(m-1)(n-1)\right]$$

As a result, if we have the following condition, the average sensitivity of the RBFS would be lower than SDBFS:

$$0 \geq j^2 + (1-n)j + \frac{n^2 + n + 1 - 2m^2 - m}{2} \tag{5.1}$$

For the special case of $m = n$ and for large values of m we see that the inequality always holds. This means that for such cases, RBFS always performs better than SDBFS. In the general case where j is between the two roots of this quadratic inequality, RBFS performs better. That is we need to have:

$$\frac{n - 1 - \sqrt{-n^2 - 4n - 1 + 4m^2 + 2m}}{2} \leq j \leq \frac{n - 1 + \sqrt{-n^2 - 4n - 1 + 4m^2 + 2m}}{2}$$

We need to analyze if the argument of the square root term is negative or not. Because if it is negative, then RBFS is never better than SDBFS. We assume $n = m + k$ where $0 \leq k$. For the argument to be positive, we need to have the following:

$$0 \geq k^2 + (2m + 4)k + (1 - 3m^2 + 2m)$$

Since k cannot be a negative number, the sufficient condition for RBFS to sometimes be better than DBFS is that:

$$k \leq -m + 2 + 2m\sqrt{1 + \frac{1}{2m} + \frac{3}{4m^2}}$$

For large values of m it means that we need to have the following:

$$k \leq m + 2$$

3) On the First or Last Column but not the Corners

For SDBFS from Equation 3.19 we have the following average sensitivity:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{m - n}{|E(G)|} + \frac{(2 - (\delta_{j,0} + \delta_{j,n-1}))(2i^2 + 2i + m^2 - 2mi - m)}{|E(G)|}$$

And simplifying for the first or last column but not corner starting nodes we have:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{2i^2 + 2i + m^2 - 2mi - n}{|E(G)|}$$

And for RBFS from Equation 4.5 we have:

$$AvgS = \frac{1}{m(n-1) + n(m-1)}[\frac{(i+3)i}{2} + \frac{(m-i+2)(m-i-1)}{2} + (n-1)(n+0.5)$$
$$+ 2(m-1)(n-1)]$$

As a result, if we have the following condition, the average sensitivity of the RBFS would be lower than SDBFS:

$$2i^2 + 2i + m^2 - 2mi + m \geq n^2 - n + 1 \tag{5.2}$$

Equivalently we need to have:

$$0 \leq i^2 + (1 - m)i + \frac{m^2 + m - n^2 + n - 1}{2}$$

For the special case of $m = n$ and for large values of m the inequality only holds for $i = 0$ and $i = m - 2$. Now, for the general case, if the above inequality doesn't have any real roots, RBFS is always better. Also, if the roots are real, i needs to be either smaller than the smaller root or greater than the greater root for RBFS to be better than DBFS. Let us first analyze the case where the roots are not real. For that, we need to have the following:

$$n^2 - n + \frac{-m^2 + 3 - 4m}{2} < 0$$

Which for large values of m never holds(since $m \leq n$). Now the other case is where i satisfies either of the following inequalities:

$$i \leq \frac{m-1}{2} - \sqrt{\frac{n^2}{2} - \frac{n}{2} + \frac{3}{4} - \frac{m^2}{4} - m}$$

or

$$\frac{m-1}{2} + \sqrt{\frac{n^2}{2} - \frac{n}{2} + \frac{3}{4} - \frac{m^2}{4} - m} \leq i$$

4) Not on the Grid's Perimeter

For SDBFS from Equation 3.19 we have the following average sensitivity:

$$\text{AvgS\_SDBFS}_{i,j} = 1 + \frac{m-n}{|E(G)|} + \frac{2(2i^2 + 2i + m^2 - 2mi - m)}{|E(G)|}$$

And for RBFS from Equation 4.6 we have:

$$\text{EMD}_{\text{Total}} = \frac{1}{m(n-1) + n(m-1)}[i(i+1) + (m-i)(m-i-1) + 2(m-1)(n-1) + j(j+1) + (n-j)(n-j-1) + \frac{m-1}{2} + \frac{n-1}{2}]$$

As a result, if we have the following condition, the average sensitivity of the RBFS would be lower than SDBFS:

$$\frac{m}{2} + 2i^2 + 2i + m^2 - 2mi \geq -\frac{n}{2} + 1 + n^2 + 2j^2 + 2j - 2nj \qquad (5.3)$$

Equivalently we have the following:

$$(i - \frac{m-1}{2})^2 - (j - \frac{n-1}{2})^2 \geq \frac{1}{4}(n^2 - m^2 - 3m + n + 2)$$

This is the equation of a hyperbola with $(\frac{m-1}{2}, \frac{n-1}{2})$ as its center. For the special case of $m = n$ therefore we have:

$$(i - \frac{m-1}{2})^2 - (j - \frac{m-1}{2})^2 \geq \frac{1}{2}(1 - m)$$

Figure 5.1 represents for which starting nodes, the RBFS algorithm performs better than or equal to SDBFS on a 7×7 grid.

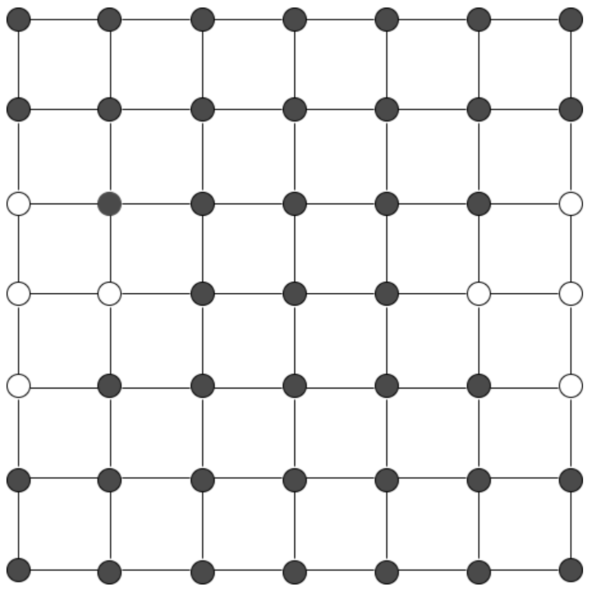

Figure 5.1: The Comparison of RBFS vs SDBFS on a 7×7 grid for all the starting nodes
The black nodes represent starting nodes that have a lower or equal average sensitivity for the RBFS algorithm.

Chapter 6

Conclusion and Future Work

To summarize, the ordering in which we explore the neighbours of a node in a grid and insert them in the queue in the BFS algorithm can change the output of the algorithm and result in different BFS trees. Therefore, different tie-breaking rules of the BFS algorithm can have different average sensitivity.

In Section 2.2.2 we saw that for the BFS algorithm in grids, if we want to have a specific deterministic tie-breaking rule for all the nodes, we have 24 possibilities which can be grouped only in 3 different categories, namely: Clockwise DBFS(CDBFS), Reading DBFS(RDBFS) and the sign of cross DBFS(SDBFS).

We proved upper bounds on the average sensitivity of CDBFS(Section 3.1), RDBFS(Section 3.2) and SDBFS(Section 3.3). In Section 3.3.3 we concluded that the SDBFS algorithm has always a lower average sensitivity compared to CDBFS. This means that SDBFS has the lowest or equal average sensitivity on grids compared to the other two deterministic BFS algorithms. For $m \times n$ grids with $m = \Theta(n)$, all deterministic BFS and randomized BFS algorithms have average sensitivity $O(1)$ on any source node. In contrast, the average sensitivity of the BFS algorithm on an arbitrary graph G is $\Theta(|V(G)|)$ [1].

When it comes to comparing SDBFS and RBFS, on average(over starting nodes), the SDBFS algorithm has a lower average sensitivity. However, for specific starting nodes, as outlined in Equations 5.1, 5.2 and 5.3, RBFS has a lower average sensitivity. Table 6.1 compares the upper bound obtained for the average over the starting nodes of average sensitivity of different BFS algorithms on grids. Also, table 6.2 compares the upper bound obtained for the average sensitivity of different BFS algorithms on grids for an arbitrary node.

Algorithm	Upper bound of Average of the Average Sensitivity over Starting Nodes
CDBFS	$1 + \frac{2(m-1)}{3(n-1)} + \frac{n-1}{3(m-1)} + O(\frac{1}{m}) + O(\frac{1}{n})$
RDBFS	$1 + \frac{m-1}{3(n-1)} + \frac{2(n-1)}{3(m-1)} + O(\frac{1}{m}) + O(\frac{1}{n})$
SDBFS	$\frac{5}{3} + O(\frac{1}{m}) + O(\frac{1}{n})$
RBFS	$1 + \frac{2m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$

Table 6.1: The Comparison of upper bounds of average(over starting node) of average sensitivities of different BFS algorithms

Algorithm	Upper bound of the Average Sensitivity for an Arbitrary Node
CDBFS	$1 + \frac{m-1}{n-1} + \frac{n-1}{m-1}$
RDBFS	$1 + \frac{m-1}{n-1} + \frac{n-1}{m-1}$
SDBFS	2
RBFS	$1 + \frac{2m+1}{4(n-1)} + \frac{2n+1}{4(m-1)}$

Table 6.2: The Comparison of upper bounds of average sensitivities of different BFS algorithms for an arbitrary node

6.1 Limitations and Future Work

The notion of k-sensitivity of graph algorithms is defined in [1] for the perturbation with arbitrary k edges removed from the original graph. The average sensitivity in this thesis is 1-sensitivity in the notion of k-sensitivity. It is interesting to analyze the k-sensitivity of the BFS algorithms on grids for $k \geq 2$. The analysis is expected to be much more complex than the 1-average sensitivity since every combination of k edges in the grid needs to be considered.

Another interesting problem worth exploring is analyzing the average sensitivity for the perturbation with edges added to the original grid. The added edge could be a diagonal edge between (i, j) and $(i+1, j+1)$ or an edge connecting two arbitrary nodes.

Analyzing the average sensitivity of 3-dimensional grids is also worth investigating.

Bibliography

[1] Nithin Varma and Yuichi Yoshida. Average sensitivity of graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 684–703. SIAM, 2021.

[2] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[3] Ehud Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18(1):27–35, 1998.

[4] Ilias Diakonikolas, Prahladh Harsha, Adam Klivans, Raghu Meka, Prasad Raghavendra, Rocco A Servedio, and Li-Yang Tan. Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 533–542, 2010.

[5] Ilias Diakonikolas, Prasad Raghavendra, Rocco A Servedio, and Li-Yang Tan. Average sensitivity and noise sensitivity of polynomial threshold functions. *SIAM Journal on Computing*, 43(1):231–253, 2014.

[6] Benjamin Rossman. The average sensitivity of bounded-depth formulas. *computational complexity*, 27:209–223, 2018.

[7] Anna Bernasconi, Carsten Damm, and Igor Shparlinski. The average sensitivity of square-freeness. *computational complexity*, 9:39–51, 2000.

[8] Pan Peng and Yuichi Yoshida. Average sensitivity of spectral clustering. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1132–1140, 2020.

[9] Soh Kumabe and Yuichi Yoshida. Lipschitz continuous algorithms for graph problems. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 762–797. IEEE, 2023.

[10] Soh Kumabe and Yuichi Yoshida. Lipschitz continuous algorithms for covering problems. *arXiv preprint arXiv:2307.08213*, 2023.

[11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv preprint arXiv:1104.1377*, 2011.

[12] Artur Czumaj, Yishay Mansour, and Shai Vardi. Sublinear graph augmentation for fast query implementation. In *International Workshop on Approximation and Online Algorithms*, pages 181–203. Springer, 2018.

[13] Soh Kumabe and Yuichi Yoshida. Average sensitivity of dynamic programming. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1925–1961. SIAM, 2022.

[14] Yuichi Yoshida and Shinji Ito. Average sensitivity of euclidean k-clustering. *Advances in Neural Information Processing Systems*, 35:32487–32498, 2022.

[15] Satoshi Hara and Yuichi Yoshida. Average sensitivity of decision tree learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[16] Yuichi Yoshida and Samson Zhou. Sensitivity analysis of the maximum matching problem. *arXiv preprint arXiv:2009.04556*, 2020.

[17] Soh Kumabe and Yuichi Yoshida. Average sensitivity of the knapsack problem. In *30th Annual European Symposium on Algorithms (ESA 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[18] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Stable algorithms for link analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–266, 2001.

[19] Andrew Y Ng, Alice X Zheng, and Michael I Jordan. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17(1), pages 903–910. Lawrence Erlbaum Associates Ltd, 2001.

[20] Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. Sharper bounds for uniformly stable algorithms. In *Conference on Learning Theory*, pages 610–626. PMLR, 2020.

[21] Gerlind Plonka and Manfred Tasche. Fast and numerically stable algorithms for discrete cosine transforms. *Linear algebra and its applications*, 394:309–345, 2005.

[22] Bengt Fornberg and Cécile Piret. A stable algorithm for flat radial basis functions on a sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2008.

[23] James W Daniel, Walter Bill Gragg, Linda Kaufman, and Gilbert W Stewart. Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. *Mathematics of Computation*, 30(136):772–795, 1976.

[24] Juan Carlos Simo and Kachung Kevin Wong. Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. *International journal for numerical methods in engineering*, 31(1):19–52, 1991.

[25] Stephen A Vavasis. Stable numerical algorithms for equilibrium systems. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1108–1131, 1994.

[26] Penggang Gao, Zihan Liu, Zongkai Wu, and Donglin Wang. A global path planning algorithm for robots using reinforcement learning. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1693–1698. IEEE, 2019.

[27] Prabin Kumar Panigrahi, Sukant Kishoro Bisoy, and Hrudaya Kumar Tripathy. Intelligent path planning with improved bfs (i-bfs) strategy for mobile robot in rfid equipped unknown environment. In *Proceedings of International Conference on Computational Intelligence and Data Engineering: ICCIDE 2021*, pages 237–249. Springer, 2022.

[28] M Bala Subramanian, K Sudhagar, and G RajaRajeswari. Optimal path forecasting of an autonomous mobile robot agent using breadth first search algorithm. *Int. J. Mech. Mechatron. Eng*, 14(2):85–89, 2014.

[29] Juan P Vásconez, Fernando Basoalto, Inesmar C Briceño, Jenny M Pantoja, Roberto A Larenas, Jhon H Rios, and Felipe A Castro. Comparison of path planning methods for robot navigation in simulated agricultural environments. *Procedia Computer Science*, 220:898–903, 2023.

[30] Pulimi Navya and R Ranjith. Analysis of path planning algorithms for service robots in hospital environment. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2021.

[31] Kunxiang Deng, Qingyong Zhang, Hang Zhang, Peng Xiao, and Jiahua Chen. Optimal emergency evacuation route planning model based on fire prediction data. *Mathematics*, 10(17):3146, 2022.

[32] Wei Zhang, Jianzhong Zhou, Yi Liu, Xiao Chen, and Chao Wang. Emergency evacuation planning against dike-break flood: a gis-based dss for flood detention basin of jingjiang in central china. *Natural Hazards*, 81:1283–1301, 2016.

[33] Anshul Jindal, Vaibhav Agarwal, and Prasenjit Chanak. Emergency evacuation system for clogging-free and shortest-safe path navigation with iot-enabled wsns. *IEEE Internet of Things Journal*, 9(13):10424–10433, 2021.

[34] Danny Dolev, Joe Halpern, Barbara Simons, and Ray Strong. A new look at fault tolerant network routing. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 526–535, 1984.

[35] Maria Engracia Gomez, Nils Agne Nordbotten, Jose Flich, Pedro Lopez, Antonio Robles, Jose Duato, Tor Skeie, and Olav Lysne. A routing methodology for achieving fault tolerance in direct networks. *IEEE transactions on Computers*, 55(4):400–415, 2006.

[36] Esfahanian and Hakimi. Fault-tolerant routing in debruijn comrnunication networks. *IEEE Transactions on Computers*, 100(9):777–788, 1985.

[37] Jose Duato. A theory of fault-tolerant routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(8):790–802, 1997.

[38] Hind Alwan and Anjali Agarwal. A survey on fault tolerant routing techniques in wireless sensor networks. In *2009 third international conference on sensor technologies and applications*, pages 366–371. IEEE, 2009.

[39] Jae H Kim, Ziqiang Liu, and Andrew A Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. *IEEE Transactions on Parallel and Distributed Systems*, 8(3):229–244, 1997.

[40] Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, and Wolfgang Rosenstiel. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 527–534. IEEE, 2007.

[41] Andrei Broder, Michael Fischer, Danny Dolev, and Barbara Simons. Efficient fault tolerant routings in networks. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 536–541, 1984.

[42] Dung Nguyen Quoc, Niansheng Liu, and Donghui Guo. A hybrid fault-tolerant routing based on gaussian network for wireless sensor network. *Journal of Communications and Networks*, 24(1):37–46, 2021.

[43] Pei-Duo Yu, Chee Wei Tan, and Hung-Lin Fu. Epidemic source detection in contact tracing networks: Epidemic centrality in graphs and message-passing algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 16(2):234–249, 2022.

[44] Polina Rozenshtein, Aristides Gionis, B Aditya Prakash, and Jilles Vreeken. Reconstructing an epidemic over time. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1835–1844, 2016.

[45] Prathyush Sambaturu, Bijaya Adhikari, B Aditya Prakash, Srinivasan Venkatramanan, and Anil Vullikanti. Designing effective and practical interventions to contain epidemics. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020.

[46] Matija Šošić and Mile Šikić. Cuda implementation of the algorithm for simulating the epidemic spreading over large networks. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1807–1810. IEEE, 2012.

[47] Siddhartha Banerjee, Aditya Gopalan, Abhik Kumar Das, and Sanjay Shakkottai. Epidemic spreading with external agents. *IEEE Transactions on Information Theory*, 60(7):4125–4138, 2014.