HARDNESS AND APPROXIMATION OF SOME GRAPH THEORETIC PROBLEMS

Diptendu Chatterjee

HARDNESS AND APPROXIMATION OF SOME GRAPH THEORETIC PROBLEMS

Thesis submitted to
Indian Statistical Institute, Kolkata
for the award of the degree

of

Doctor of Philosophy

in

Computer Science

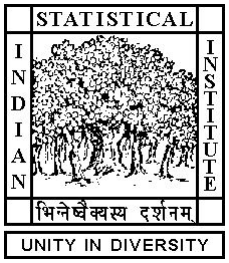
by

Diptendu Chatterjee

under the guidance of

Professor Bimal Kumar Roy
(Applied Statistics Unit, Indian Statistical Institute, Kolkata, India)
and

Dr. Rishiraj Bhattacharyya
(School of Computer Science, University of Birmingham, Birmingham, UK)



APPLIED STATISTICS UNIT
INDIAN STATISTICAL INSTITUTE, KOLKATA

July 2022

- To my PARENTS and SISTER -

Publications from this Thesis

Journals

- Diptendu Chatterjee, Bimal Kumar Roy, "**An Improved Scheduling Algorithm for Traveling Tournament Problem with Maximum Trip Length Two**," *Open Access Series in Informatics (OASIcs)*, Volume 96, ATMOS 2021, DOI: `https://doi.org/10.4230/OASICS.ATMOS.2021.16`, Part of ISSN: 2190-6807

Submitted(Under Review)

- Diptendu Chatterjee, "**Complexity of Traveling Tournament Problem with Trip Length More Than Three**," arXiv preprint, DOI: `https://doi.org/10.48550/arXiv.2110.02300`, under review with *Theory of Computing Systems*, Submitted on 29-05-2022 with Submission ID: 634868cf-b829-48a4-8b53-bba737a77926.

- Diptendu Chatterjee, "**An Approximation Algorithm of Firebreak Problem on Split Graphs**," under review with *Networks*, Submitted on 24-02-2022 with Manuscript ID: NET-22-0042.

- Diptendu Chatterjee, Rishiraj Bhattacharyya, "**Firefighter Problem with Minimum Budget: Hardness and Approximation Algorithm for Unit Disk Graphs**," arXiv preprint, DOI: `https://doi.org/10.48550/arXiv.2203.15509`, under review with *Journal of Combinatorial Optimization*, Submitted on 29-06-2022 with Manuscript ID: JOCO-D-22-00245.

Acknowledgment

This thesis marks the end of a crucial and meaningful journey during which there were many people whom I would like to acknowledge for their encouragement and support. I would like to express my sincere gratitude to my supervisors, Professor Bimal Kumar Roy and Dr. Rishiraj Bhattacharyya, for their patience and constant encouragement during my Ph.D. study and research. I also thank them for their valuable guidance and suggestions, without which this thesis would not be in its present form.

I express my gratitude to the members of the Doctoral Scrutiny Committee for their advice and care. I also express my earnest thanks to the Head of the Applied Statistics Unit, ISI Kolkata for providing all the possible facilities for this research work. I would like to thank all the faculty members, especially the Ph.D. coordinators of the Computer Science Division.

I thank my lab-mates Avishek, Samir, Amit, Nishant, Aniruddha, Subhadeep, Prabal, Subhra, Mustaf, Jyotirmay, Animesh, Manish, Gourab, Laltu, Pravat, Aparajita, Abhishek, Subhabrata Da(Samajdar), Sanjay Da(Bhattacharyya), Nayana, Pritam and Soumya for the enjoyable and helpful company I had with. I would like to thank all my friends at ISI Kolkata for the refreshing company I had with them.

I dedicate this work to my parents and my sister for their support, love, encouragement and patience.

Diptendu Chatterjee

Diptendu Chatterjee

Abstract

In the real world, we encounter many problems that can be modeled as graph-theoretic problems. This modeling gives a concrete view of the constraints and objectives of the problem and allows us to apply some well-known techniques to solve it. Many of these problems do not have their computational complexity settled; on the other hand, many others have been proved to be NP-hard. Thus should be approximated. This thesis focuses on these aspects of some graph theoretic problems.

The Traveling Tournament Problem is one of the interests of this thesis. A constrained Traveling Tournament Problem(TTP-k) asks for a schedule of a double round-robin tournament with an upper bound(k) on the lengths of home stands and away trips of the teams where the total travel distance is minimized. The hardness of the problem varies with the upper bound. This thesis attempts an approximation algorithm for TTP-2, which is assumed to be NP-Hard. Then considers a study on the hardness analysis of TTP-k where $k > 3$ and $k \in \mathbb{N}$.

The Firefighter Problem is an important graph theoretic problem with practical application in a recent pandemic scenario. The firefighter problem asks for a solution to save vertices in a graph by placing firefighters on some of them where a fire broke out in a vertex and spread through the network with time. This thesis considers Firefighter Problem on Unit Disk Graphs. Most networks can be modeled in this wireless era as Unit Disk Graphs. The hardness of the problem and an approximation algorithm for the same is attempted in this thesis. Then a special version of the firefighter problem called the Firebreak Problem is considered where the firefighters can be placed on the vertices only at the initial time instance when the fire breaks out. An approximation algorithm is attempted for the Firebreak Problem on Split Graphs which has been proven to be NP-Hard.

Keywords: Traveling Tournament Problem, Double Round-robin, Firefighter Problem, Firebreak Problem, NP-Hardness.

Contents

List of Figures

List of Tables

List of Abbreviations

TTP	Traveling Tournament Problem
Min-Budget	Minimum Budget
NP	Nondeterministic Polynomial Time

CHAPTER 1

Introduction

In the field of Discrete Mathematics, Graph Theory is one of the most important subjects. The main reason for this is the similarity of graphs with real-world situations and structures. The similarities are mainly found when dealing with connectivity, travel, or distribution-related problems. In this thesis, hardness or computational complexity-related works on some of the graph theoretic problems have been presented.

1.1 Motivation

This thesis is concerned with two types of graph theoretic problems: Tournament Scheduling Problem and Firefighter Problem.

Nowadays, sports tournaments are very popular events. Many of these tournaments involve travel by the participating teams between the home venues of the participating teams. This travel cost is a major part of the tournament budget. The Tournament management wants to minimize the total travel in the tournament. Scheduling the tournament properly can optimize the cost to a great extent. This practical real-world cause makes the tournament scheduling problem worth studying.

Any social, distribution or communication network can be modeled as a graph. If a harmful contagion or material or program starts spreading through this network, then the nodes or elements are needed to be saved from that. This situation in real life can be modeled as a firefighter problem on graphs where it is needed to save the vertices of the graph from a spreading fire by placing firefighters on some of the vertices. Keeping the recent pandemic scenario of the world in mind, different versions of the firefighter problem becomes worth studying.

1.2 Background

In this thesis, two problems related to tournament scheduling have been discussed. They both are traveling tournament problems. Traveling tournament problem deals with the scheduling of double round-robin tournaments. In a double round-robin tournament, each participating team has its home venue and any pair of teams play two matches between them, once at each of their home venues. So from the structure of the tournament, it is evident that teams travel to different venues throughout the tournament. The goal is to minimize the total travel by all the teams throughout the tournament. This minimization depends on many factors. One of them is the upper bound on any team's consecutive home and away matches. This thesis concerns with the cases when this upper limit is two or any natural number greater than three. Traveling Tournament Problem is called TTP in short. In the work related to TTP-2, a betterment of the existing result available in the literature for practical cases has been shown. Also, in another work, the computational hardness of constrained TTP has been shown.

Firefighter problems can be classified into two major types: Max-save and Min-budget. In both cases, fire breaks out at a vertex of a given graph and at each time instance, it spreads to all the unprotected neighbors of a vertex on fire. The goal for the max-save case is to maximize the number of saved vertices by placing a given number of firefighters on different nodes at each time instance. In the min-budget case, the goal is to calculate the minimum number of firefighters required at each time step to save a given set of vertices of the graph. One of the problems related to the firefighter problem in this thesis is the firefighter min-budget problem on unit disk graphs. Unit disk graph is a special class of graphs

described later in this thesis.

A firefighter problem becomes a firebreak problem when the firefighters can be placed only once at the time of fire breakout and a given number of vertices has to be saved using the minimum number of firefighters. The other problem related to the firefighter problem in this thesis is the firebreak problem on split graphs.

In several chapters in this thesis, the terms NP-Hard and NP-Complete will be used repeatedly. These represent specific classes of computational complexities of the problems in Computer Science. Here NP stands for Non-deterministic Polynomial Time. A problem is in the NP class if given a solution to the problem, a deterministic algorithm can verify its correctness within a specified polynomial (of the input size) time. On the other hand, NP-Hard problems are computationally at least as hard as the hardest problems in the NP class. A problem in NP-Hard class can be reduced from a problem in the NP class in polynomial time. The problems which belong to both in NP and NP-Hard classes are said to be in NP-Complete class. So NP-Complete class is the intersection of NP class and NP-Hard class.

The main issues while working on the problems mentioned earlier are described in the following section.

1.3 Concluding Remarks

The survey so far has brought out certain issues regarding the problems that can be summarized as follows:

- Computational Complexity of TTP-2 is still not settled.
- Although many theoretical approximations have been done on TTP-2, practical results are less in number.
- There is a possibility of a better result of TTP-2 for less number of teams compared to the existing generalized solution.
- Although work on the computational complexity of TTP-3 and TTP-∞ is present in literature, a study on the computation complexity of general TTP-k for any natural number k has not been done yet.

- Unlike TTP-2, finding a lower bound for TTP-k is hard. Over that, synchronously fitting a schedule makes it even tougher.
- Computational complexity of firefighter problem on unit disk graphs is not settled. Hence there is no approximation result for this problem.
- Although NP-hardness of firebreak problem on split graphs has been shown, no approximation of the problem has been done yet.

1.4 Scope of the Present Work

The remaining chapters of this thesis attempt to address the issues mentioned in the previous section. The development is as follows:

- **Chapter 2** will attempt to find an improved approximation result of the Traveling Tournament Problem with Maximum Trip Length Two for real-world tournaments, specifically when the number of participating teams is divisible by 4 and less or equal to 32. This eventually includes all practical double round-robin tournaments played these days.
- **Chapter 3** will try to evaluate the computational complexity of TTP-k for $k > 3$ and $k \in \mathbb{N}$. TTP-1 has been proven to be impossible to schedule. TTP-2 is assumed to be NP-hard and approximation algorithms are attempted. TTP-3 and TTP-∞ have been proven to be NP-hard. The only gap in terms of the complexity of TTP is for the case of TTP-k where $k > 3$ and $k \in \mathbb{N}$. So it seems worthy to establish NP-hardness of general TTP-k.
- **Chapter 4** will aim to prove the NP-hardness of the Firefighter Problem on Unit Disk Graphs and also strive to present an approximation algorithm. Although the NP-hardness of the Firefighter Problem on general graphs and several other graph classes has been proved already, the question for the unit disk graphs is still open. Unit disk graphs are of great importance due to their similarity with real-world communication systems.
- **Chapter 5** will seek to present an approximation algorithm for the Firebreak Problem on Split Graphs. The firebreak Problem was introduced recently in

2020 and the NP-hardness of the problem on Split Graphs has been proved. The Firebreak Problem on Split Graphs structure resembles the current situation of protecting people in a social network from a harmful contagion.

CHAPTER 2

A Better Approximation Algorithm for Travelling Tournament Problem with Maximum Trip Length Two

2.1 Introduction

A double round-robin tournament is one of the most unbiased ways of evaluating teams participating in a competition. In these kind of tournaments, each participating team plays with every other team twice, i.e., one home match and one away match. This nullifies the effects of home advantage. So, in these kind of tournaments, each team is tested in all the venues and in all the conditions. If there are n teams participating, then each team will play $2(n-1)$ games and the total number of games played will be $n(n-1)$. After all the matches are played, the team with the highest point wins the tournament. The Traveling Tournament Problem (TTP) is a combinatorial optimization problem for a double round-robin tournament. In TTP, we have to provide a scheduling algorithm that minimizes the total distance traveled by all participating teams maintaining

certain constraints. Most of the instances of TTP with more than ten teams are still unsolved. The number of participating teams in a TTP has to be even. The distance between the participating teams' home venues is symmetric and satisfies the triangle inequality. Moreover, the distances are rational numbers and the time instances are discrete-time instances which can be expressed as integers. There are different variants of this problem depending on the constraints. TTP is inspired by *Major League Baseball* in USA. For given participating teams and all the mutual distances between their home grounds, the general form of constrained Traveling Tournament Problem, i.e., TTP-k for some natural number k, is defined as follows.

Definition 2.1 ***TTP-****k is the problem of scheduling a double round-robin tournament where the total travel distance by all the participating teams is minimized given the following constraints:*

1. *Each pair of participating teams play exactly two matches once in each of their home venues.*
2. *No pair of teams play consecutive matches with each other.*
3. *In an away tour, a visiting team travels directly from the home of one opponent to the home of the next opponent without returning to its own home. At the end of the tournament, all the teams return to their respective home venues.*
4. *The lengths of the home stands and away tours for any participating team are not more than k.*

For an odd number of teams, a solution to a Traveling Tournament Problem is impossible as every team should participate on a match day.

Despite of several benefits, Traveling Tournament Problem has some drawbacks also. The main drawbacks are: a huge number of matches and scheduling complexity. Although we can not decrease the number of matches, we can lower the complexity of the scheduling. But with imposed constraints on scheduling, the complexity increases. For a small number of teams, the scheduling is simpler and the complexity increases with the number of teams and imposed constraints.

TTP-∞ and TTP-3 has been proven to be NP-hard in [1] and [2] respectively. TTP-1 is impossible to schedule [3]. So, the only case where a complete solution may be possible is TTP-2. The complexity of TTP-2 is still not settled. Xiao and Kou gave the existing best result on approximating TTP-2 [4]. They gave an approximation factor of $(1 + \frac{2}{n} + \frac{2}{n-2})$ for TTP-2 on n teams with n divisible by 4. We also work on a similar setup, where we schedule a TTP-2 on n teams with n divisible by 4 and our schedule improves the result for $n \leq 32$.

A formal definition of the problem, other useful definitions, notations and well-known results related to the Traveling Tournament Problem are given in the following section.

2.1.1 Problem Definition

TTP-2: Traveling Tournament Problem-2 is the problem of scheduling a double round-robin tournament where the total travel distance by all the participating teams is minimized, maintaining the following constraints:

$\underline{C_1}$: Each pair of participating teams play exactly twice with each other i.e., once in each of their home venues.

$\underline{C_2}$: No pair of teams play consecutive matches with each other.

$\underline{C_3}$: In an away tour, a visiting team travels directly from one opponent's home to the next without returning to its home and at the end of the tournament, all the teams return to their respective home venues.

$\underline{C_4}$: The lengths of the home stands and away tours for any participating team are not more than 2.

2.1.2 Previous Work

Traveling Tournament Problem (TTP) is a special variant of the Traveling Salesman Problem (TSP). The Traveling Tournament Problem was first introduced by Easton, Nemhauser, and Trick [5]. In a TTP, when there is no constraint on home stands or away trip length, it becomes a problem of synchronously scheduling n

Traveling Salesman Problem. It has been shown that TTP-k is NP-Hard when $k = \infty$ by Bhattacharyya [1]. Thielen and Westphal showed the NP-Hardness of TTP-3 [2]. The relationship of some variants of round-robin tournaments with the planar three-index assignment problem has been analyzed and the complexity of scheduling a minimum cost round-robin tournament has been established using the same by Briskorn, Drexl and Spieksma[6]. They also showed the applicability of some techniques for the planar three-index assignment problem to solve a sub-problem of scheduling a minimum-cost round-robin tournament. Lots of work has been done towards the approximation algorithms of TTP [4, 7, 8, 9, 10, 11]. Works on heuristic algorithms of TTP can be found in [12, 13, 14, 15, 16]. Many offline and online set of benchmark data can be found for TTP-3 in [5, 17]. Van Hentenryck and Vergado showed that, for many benchmark results on improvements and complete solutions, even high-performance computers take more than a week [18]. But even then most of the instances of TTP-k on more than 10 teams are not completely solvable, as shown by Trick [17]. They worked on a basketball tournament with ten teams where the away trip for any team consisted of one or two matches. It has also been shown that TTP-1 is impossible to schedule by De Werra [3]. Rasmussen and Trick [19] conducted a survey on round-robin tournament scheduling. Work has also been done on complexity of TTP-k [20, 21, 22].

Our main focus is on TTP-2, which was introduced by Campbell and Chen [23]. Thielen and Westphal [24] has contributed towards approximation factor for TTP-2 and later gave an approximation factor of $(1 + \frac{16}{n}); \forall n \geq 12$ and n divisible by 4. Xiao and Kou have improved their result [4]. They gave an approximation factor of $(1 + \frac{2}{n-2} + \frac{2}{n})$ where n is divisible by 4. Our scheduling algorithm give better result than this for $n \leq 32$.

2.1.3 Our Result

We propose a scheduling algorithm for TTP-2 which yields an approximation factor of $\left(1 + \frac{\lceil \log_2 \frac{n}{4} \rceil + 4}{2(n-2)}\right)$. For number of participating teams less or equal to 32 and divisible by 4, this gives a better result than existing best result, with approximation factor of $(1 + \frac{2}{n-2} + \frac{2}{n})$ in [4].

2.2 Preliminaries

2.2.1 Definitions and Notations

In this chapter, the graph theoretic approach has been followed to get a better approximation factor for TTP-2. Here, teams are invariably referred to as vertices and distances between home locations of teams are referred to as weights of edges of the graph.

Definition 2.2 ***Matching Graph:*** *A matching graph $G(V, E)$ is a graph where no two edges have a common vertex. So, for a matching graph, $|V| = n \Rightarrow |E| \leq \frac{n}{2}$. The pair of vertices connected through an edge in a matching graph is called* ***matched vertices*** *of the matching graph.*

Definition 2.3 ***Maximal Matching of a Graph:*** *Maximal matching M of a graph $G(V, E)$ is a matching graph, where for any other matching M' of G, $M \nsubseteq M'$. It may not be unique for a given graph.*

Definition 2.4 ***Minimum Maximal Matching of an Undirected Weighted Graph:*** *Minimum Maximal Matching of an Undirected Weighted Graph $G(V, E)$ is a maximal matching of G with the sum of all the weights of its edges to be the smallest among all the maximal matching subgraphs of G. For a minimum maximal matching of an undirected weighted complete graph with n vertices, the number of edges of the matching will be $\frac{n}{2}$.*

In this work, we represents an edge between two vertices as a match between the teams corresponding to the vertices. Now a *super-match* is defined as follows:

Definition 2.5 ***Super-match:*** *A super-match between two pairs of matched vertices M_i and M_j is the set of edges $\{(u, w), (u, x), (v, w), (v, x)\}$ where $M_i = \{u, v\}$ and $M_j = \{w, x\}$.*

2.2.2 A Simple Lower Bound for TTP-2

Suppose there are n teams participating in TTP-2. Distances between the home locations of each pair of teams are given. Let, d_{ij} be the distance between home

locations of i^{th} and j^{th} team. Now we construct an undirected weighted complete graph $G(V, E)$ with all the n home locations as vertices and weights of the edges as the physical distances between the home locations of teams corresponding to the vertices connected through it. As G is a complete graph and $|V| = n$ (even), we get a minimum maximal matching in G and call it G_m. Let the sum of the weights of all the edges of G_m be W_m; the sum of the weights of all the edges in G be W_t and the sum of the weights of all the edges from a vertex i in G be W_i.

So, for an optimized schedule with the given constraints, it is natural for a team to travel to two matched teams in G_m in an away trip. But for the vertex matched with itself in G_m, it will make a to and fro journey. In that case, the total travel by i^{th} team is $W_i + W_m$. This gives a minimum travel by ith team given the constraints.

Now, if it is possible to synchronously fit this above-mentioned minimum travel by each participating team in the schedule, then the total traveled distance by all the teams in the tournament will be,

$$\sum_{i \in V} (W_i + W_m) = 2W_t + nW_m.$$

This gives a lower bound of TTP-2. But due to the imposed constraints on scheduling and the number of teams, it is not always possible to synchronously fit the minimum travel schedule of each participating team in the schedule. The hardness of the optimization makes this problem interesting.

There are $(\frac{n}{2} - 1)!$ number of minimum travel schedules for each participating team. For synchronously fitting one of these Minimum Travel Schedules for each team, we need an exhaustive search over a space of size exponential of the number of teams. This gives an intuitive idea of the NP-Hardness of the problem.

2.3 Design of Schedule

Suppose there are n teams participating in a double round-robin tournament where n is divisible by 4. We construct the undirected weighted graph $G(V, E)$ as described in **Section** 2 and also find the minimum maximal matching G_m in G. Now we number the vertices and the matched pairs such that the matched pair M_i con-

sist of vertices $2i-1$ and $2i$, $\forall i \in \{1,\ldots,\frac{n}{2}\}$. Now, we design the schedule in $\lceil \log_2 \frac{n}{2} \rceil$ rounds and $(\frac{n}{2}-1)$ levels such that i^{th} round consists of $\lceil \frac{1}{2}(\frac{n}{2^i}-1) \rceil$ levels and each level consists of $\frac{n}{4}$ *super-matches*. A *super-match* is played between two different matched pairs where both the teams in a matched pair play home and away matches with both the teams in the other matched pair. In every level each matched pair plays a *super-match*. We have designed three types of *super-matches* which are used in our schedule. Suppose two pairs of matched vertices are A_1, A_2 and B_1, B_2 in G_m, as described in the previous section. We give three types of *super-match* namely Type-1, Type-2, Type-3 which are the building blocks of our schedule.

Type-1: This consists of four match days namely T_1, T_2, T_3 and T_4 and the matches on this match days are given below:
$T_1 : A_1 \rightarrow B_1, A_2 \rightarrow B_2$.
$T_2 : A_1 \rightarrow B_2, A_2 \rightarrow B_1$.
$T_3 : B_1 \rightarrow A_1, B_2 \rightarrow A_2$.
$T_4 : B_1 \rightarrow A_2, B_2 \rightarrow A_1$.
where $u \rightarrow v$ means u is playing an away match with v in the home of v.
The home-away match sequence of the participating teams become the following:
$A_1 : Away - Away - Home - Home$
$A_2 : Away - Away - Home - Home$
$B_1 : Home - Home - Away - Away$
$B_2 : Home - Home - Away - Away$.

Type-1 *super-match* does not violate minimum travel of any of its participating teams.
This way, we can simultaneously schedule $\frac{n}{4}$ Type-1 *super-matches* in a level but then we can not schedule matches between the teams with the same home away match sequences due to **constraint:4** of the problem definition. So we need a different kind of *super-match* than Type-1 and hence comes the need of Type-2 *super-match*.

Type-2: This consists of four match days namely T_1, T_2, T_3 and T_4 and the matches on this match days are given below:

$T_1 : A_1 \to B_1, A_2 \to B_2.$
$T_2 : B_2 \to A_1, B_1 \to A_2.$
$T_3 : B_1 \to A_1, B_2 \to A_2.$
$T_4 : A_1 \to B_2, A_2 \to B_1.$

The home-away match sequence of the participating teams become the following:
$A_1 : Away - Home - Home - Away$
$A_2 : Away - Home - Home - Away$
$B_1 : Home - Away - Away - Home$
$B_2 : Home - Away - Away - Home.$

Type-2 *super-match* violates the minimum travel of all of its participating teams but helps to schedule matches of all the teams according to their minimum travel schedule in the next level. We may refer the Type-2 *super-match* as *flip* in future. But also after this modification, it is not possible to schedule home and away matches between two matched teams in G_m, maintaining their minimum travel schedule. So there comes the need for a Type-3 schedule block.

Type-3: This consists of six match days namely T_1, T_2, T_3, T_4, T_5 and T_6 and the matches on this match days are given below:

$T_1 : A_1 \to B_1, A_2 \to B_2.$
$T_2 : A_1 \to A_2, B_2 \to B_1.$
$T_3 : B_2 \to A_1, B_1 \to A_2.$
$T_4 : A_2 \to A_1, B_1 \to B_2.$
$T_5 : A_1 \to B_2, A_2 \to B_1.$
$T_6 : B_1 \to A_1, B_2 \to A_2.$

The home-away match sequence of the participating teams become the following:
$A_1 : Away - Away - Home - Home - Away - Home$
$A_2 : Away - Home - Home - Away - Away - Home$
$B_1 : Home - Home - Away - Away - Home - Away$

$B_2 : Home - Away - Away - Home - Home - Away.$

Although Type-1 *super-match* does not violate the minimum travel schedule for the teams, we can not schedule a double round robin tournament only with Type-1 *super-matches*. We need Type-2 and Type-3 *super-matches*. Now, $\frac{n}{4}$ number of Type-3 *super-matches* are unavoidable for any TTP-2 scheduling as each Type-3 *super-match* involves home and away matches between matched vertices for two pairs of matched vertices of G_m described in the previous section. So, for n participating teams at least $\frac{n}{4}$ number of Type-3 *super-matches* are required and our algorithm uses exactly $\frac{n}{4}$ numbers of Type-3 schedule blocks. Now, the only scope of improvement is reduction in numbers of Type-2 *super-matches*. So our main aim to keep the the number of Type-2 *super-matches* or *flips* as low as possible.

2.4 Our Algorithm

Following algorithm gives a improved schedule in terms of total distance traveled by all the teams than the existing best result [4] for TTP-2 when, $n \leq 32$ where the number of Type-2 super matches are bounded by $\left(\frac{n}{8} * \lceil \log_2 \frac{n}{4} \rceil\right)$.

In the Algorithm-1 for TTP-2 of n teams our technique is presented. We work on a euclidean plane where n teams are situated at their home venues. These teams at their home venues are the vertices of the graph and the distance between the home venues are the weights of the corresponding edge in the graph. First, we find the Minimum Maximal Matching on all the vertices or teams in this complete graph. Let the set of matched pair of vertices be $\{M_1, \ldots, M_{n/2}\}$. Then we consider each M_i's as a team situated at the midpoint of the physical locations of the constituent vertices in the euclidean plane. This midpoint serves as the location of the matched pair of vertices when considering super matches between different pairs of vertices. Then for a complete graph on these M_i's as vertices, we again find the minimum maximal matching and let the set of matched vertices be $\{N_1, \ldots, N_{n/4}\}$.

Now, we schedule the Type-2 super-matches in the different levels of different rounds according to the rule described in line 12 or line 14 of the algorithm,

Algorithm 1 Schedule TTP-2

INPUT: $G(V, E)$ with $|V| = n, |E| = \binom{n}{2}, W = \{w_e | e \in E\}$.
Identify the minimum maximal matching, $G_m(V, E_m)$, of G.
For all $i \in \{1, \ldots, \frac{n}{2}\}$, define $M_i = \{(u, v) | u, v \in V \ \& \ \text{Edge}(u, v) \in E_m\}$.
For all $v \in V$, allot a number to v such that $(u, v) \in M_i \implies \#u = (2i - 1) \ \& \ \#v = 2i \ \forall i \in \{1, \ldots, \frac{n}{2}\}$.
Define $X = \{x_i | \text{location of } x_i \text{ is in the midpoint of } u \ \& \ v \text{ where } (u, v) \in M_i \ \forall i \in \{1, \ldots, \frac{n}{2}\}\}$.
Define a complete graph $H(X, E') | \forall e \in E'$, weight of the edge $e, W_e = \delta(x_m, x_n)$ where e is the edge between x_m & x_n.
Identify the minimum maximal matching, $H_m(X, E'_m)$, of H.
For all $i \in \{1, \ldots, \frac{n}{4}\}$, define$N_i = \{(M_m, M_n) | x_m, x_n \in X \ \& \ \text{Edge}(x_m, x_n) \in E'_m\}$.
for $i = 1 : 1 : \lceil \log_2 \frac{n}{2} \rceil$ **do**
  while $2^{i+1} < n$ **do**
    if $2^{i+2} | n$ **then**
      Schedule first $\left\lceil \frac{1}{2} \times (\frac{n}{2^i} - 1) \right\rceil - 1$ levels of i^{th} round each with $\frac{n}{4}$ Type-1 *super-matches* and last level with $\frac{n}{8}$ Type-1 and $\frac{n}{8}$ Type-2 *super-matches*.
    else
      Schedule the $\left\lfloor \frac{n}{2}(1 - 2^{-i}) - 1 \right\rfloor^{th}$ match days with $\left\lfloor \frac{n}{8} \right\rfloor$ Type-2 *super-matches* for $i \in \{1, 2, \ldots, \log_2 n\}$ and rest of the super-matches as Type-1. For all other match days except the last one schedule all super-matches as Type-1.
    end if
    Schedule this last level of the tournament with $\frac{n}{4}$ Type-3 *super-matches* where $\forall i \in \{1, \ldots, \frac{n}{4}\}, M_p$ plays with $M_q | M_p, M_q \in N_i$.
  end while
end for

depending on the value of n. We schedule all the Type-3 super-matches in the last level of the last round of the tournament between matched pairs of M_i's, i.e., between the elements of N_i's to minimize the total travel distance. In next section, few schedules are given as examples using our algorithm.

2.5 Examples of Scheduling with Our Algorithm

For better understanding of our scheduling algorithm we give two examples of schedule for $n = 12, 16$ in the following section and for $n = 20, 24, 28$ in the

Appendix-A.1. An improved schedule of *Indian Premier League*, where n=8, is presented in Appendix-A.2. Let,

$$F_n = \frac{n}{8} * \left\lceil \log_2 \frac{n}{4} \right\rceil \quad \text{for } n \in \mathbb{N}. \tag{2.1}$$

2.5.1 Schedule for $n = 12$

For designing a schedule for Traveling Tournament Problem with 12 teams using our technique, first we number the teams or the vertices with natural numbers as follows.

Vertex Set={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}.

Then we find the Minimum Maximal Matching in the complete graph containing the vertices in the above-mentioned vertex set. Let the set of matched pair of vertices be $\{M_1, M_2, M_3, M_4, M_5, M_6\}$ and without loss of generality, we can say that,

M_1={1,2}, M_2={3,4}, M_3={5,6}, M_4={7,8}, M_5={9,10}, M_6={11,12}.

Then we consider each M_i's as a team situated at the mid point of the locations of its constituent vertices for $i \in \{1, 2, 3, 4, 5, 6\}$. Then for a complete graph on these M_i's as vertices, we find the Minimum Maximal Matching and let the set of matched vertices be

$\{N_1, N_2, N_3\}$ such that N_1=$\{M_1, M_5\}$, N_2=$\{M_2, M_3\}$, N_3=$\{M_4, M_6\}$.

Now, we describe below the super-matches to be scheduled in all the levels of all the rounds according to our scheduling technique in a tabular form. We can observe that the super-matches scheduled in the last level of the last round of the tournament are between matched pairs of M_i's, i.e., between the elements of N_i's.

Number of *Flips*= 3 = F_{12}.

2.5.2 Schedule for $n = 16$

For designing a schedule for Traveling Tournament Problem with 16 teams using our technique, first we number the teams or the vertices with natural numbers in a similar fashion as follows.

Vertex Set={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}.

Round:1, Level:1	**Round:1, Level:2**	**Round:1, Level:3**
$M_1 \xrightarrow{Type-1} M_2$	$M_1 \xrightarrow{Type-1} M_4$	$M_1 \xrightarrow{Type-1} M_3$
$M_3 \xrightarrow{Type-1} M_4$	$M_3 \xrightarrow{Type-2} M_6$	$M_6 \xrightarrow{Type-2} M_2$
$M_5 \xrightarrow{Type-1} M_6$	$M_5 \xrightarrow{Type-1} M_2$	$M_5 \xrightarrow{Type-1} M_4$

Round:2, Level:1	**Round:3, Level:1**
$M_1 \xrightarrow{Type-2} M_6$	$M_6 \xrightarrow{Type-3} M_4$
$M_2 \xrightarrow{Type-1} M_4$	$M_2 \xrightarrow{Type-3} M_3$
$M_5 \xrightarrow{Type-1} M_3$	$M_5 \xrightarrow{Type-3} M_1$

Then we find the Minimum Maximal Matching in the complete graph containing the vertices in the above-mentioned vertex set. Let the set of matched pair of vertices be $\{M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8\}$ and without loss of generality, we can say that

M_1={1,2}, M_2={3,4}, M_3={5,6}, M_4={7,8}, M_5={9,10}, M_6={11,12}, M_7={13,14}, M_8={15,16}.

Then we consider each M_i's as a team situated at the mid point of the locations of its constituent vertices for $i \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. Then for a complete graph on these M_i's as vertices, we find the Minimum Maximal Matching and let the set of matched vertices be

N_1=$\{M_1, M_5\}$, N_2=$\{M_2, M_6\}$, N_3=$\{M_3, M_7\}$, N_4=$\{M_4, M_8\}$.

Now, we describe below the super-matches to be scheduled in all the levels of all the rounds according to our scheduling technique in a tabular form. We can observe that the super-matches scheduled in the last level of the last round of the tournament are between matched pairs of M_i's, i.e., between the elements of N_i's. Also, as 8 is a power of 2, we know exactly the super-matches, which are *flips* in the different levels of all the tournament rounds according to our scheduling technique.

Number of *Flips*= $4 = F_{16}$.

Correctness of this algorithm is assured by the structures of Type-1, Type-2 and Type-3 *super-matches* as all three of these structures do not violate any of

Round:1, Level:1	**Round:1, Level:2**	**Round:1, Level:3**	**Round:1, Level:4**
$M_1 \xrightarrow{Type-1} M_2$	$M_1 \xrightarrow{Type-1} M_4$	$M_1 \xrightarrow{Type-1} M_6$	$M_1 \xrightarrow{Type-1} M_8$
$M_3 \xrightarrow{Type-1} M_4$	$M_3 \xrightarrow{Type-1} M_6$	$M_3 \xrightarrow{Type-1} M_8$	$M_3 \xrightarrow{Type-2} M_2$
$M_5 \xrightarrow{Type-1} M_6$	$M_5 \xrightarrow{Type-1} M_8$	$M_5 \xrightarrow{Type-1} M_2$	$M_5 \xrightarrow{Type-1} M_4$
$M_7 \xrightarrow{Type-1} M_8$	$M_7 \xrightarrow{Type-1} M_2$	$M_7 \xrightarrow{Type-1} M_4$	$M_7 \xrightarrow{Type-2} M_6$
Round:2, Level:1	**Round:2, Level:2**	**Round:3, Level:1**	
$M_1 \xrightarrow{Type-1} M_3$	$M_1 \xrightarrow{Type-1} M_7$	$M_1 \xrightarrow{Type-3} M_5$	
$M_5 \xrightarrow{Type-1} M_7$	$M_5 \xrightarrow{Type-2} M_3$	$M_3 \xrightarrow{Type-3} M_7$	
$M_2 \xrightarrow{Type-1} M_8$	$M_2 \xrightarrow{Type-1} M_4$	$M_2 \xrightarrow{Type-3} M_6$	
$M_6 \xrightarrow{Type-1} M_4$	$M_6 \xrightarrow{Type-2} M_8$	$M_8 \xrightarrow{Type-3} M_4$	

the constraints in the problem definition. So, our schedule also does not violate any of the constraints, which proves the correctness of our algorithm.

2.6 Proof of Results

Theorems related to the analysis of the proposed algorithm and their proofs are presented in this section.

Let $\delta(x, y)$ denote the distance between the teams' home venues x and y.

Theorem 2.1 *All the Type-3 schedule blocks introduce a relative error at most* $\frac{2}{n-2}$ *times the lower bound of TTP-2.*

Proof. Suppose for some $i \in \{1, \ldots, \frac{n}{4}\}$, N_i includes 4 vertices of G i.e., A_1, A_2, B_1, B_2 where A_1 and A_2 are matched pairs in G_m and so are B_1 and B_2. For a Type-3 schedule in between them, travel for each team are given below:

$A_1 : A_1 \to B_1 \to A_2 \to A_1 \to B_2 \to A_1$
$A_2 : A_2 \to B_2 \to A_2 \to A_1 \to B_1 \to A_2$
$B_1 : B_1 \to A_2 \to B_2 \to B_1 \to A_1 \to B_1$
$B_2 : B_2 \to B_1 \to A_1 \to B_2 \to A_2 \to B_2$.

So the total distance traveled is,

$$5\delta(A_1, B_1) + 3\delta(A_2, B_1) + 2\delta(A_1, A_2) + 3\delta(A_1, B_2) + 5\delta(A_2, B_2) + 2\delta(B_1, B_2).$$

For the minimum travel schedule the value is,

$$2\delta(A_1, B_1) + 2\delta(A_2, B_1) + 6\delta(A_1, A_2) + 2\delta(A_1, B_2) + 2\delta(A_2, B_2) + 6\delta(B_1, B_2).$$

So the extra amount of travel is,

$$3\delta(A_1, B_1) + 1\delta(A_2, B_1) - 4\delta(A_1, A_2) + 1\delta(A_1, B_2) + 3\delta(A_2, B_2) - 4\delta(B_1, B_2).$$

Using triangle inequality,the above expression is upper bounded by,

$$2\delta(A_1, B_1) + 2\delta(A_2, B_2) + 2\delta(A_1, B_2) + 2\delta(A_2, B_1).$$

Let us denote, *super-edge* D_{ij} between pairs A_1, A_2 and B_1, B_2 as,

$$\delta(A_1, B_1) + \delta(A_2, B_2) + \delta(A_1, B_2) + \delta(A_2, B_1).$$

where

$$A_1, A_2 \in M_i \text{ and } B_1, B_2 \in M_j \text{ for some } i, j \in \{1, \ldots, \tfrac{n}{2}\}.$$

Now, there are $\frac{n}{2}$ numbers of pair of vertices like A_1, A_2. If we consider all pairwise distances between all these $\frac{n}{2}$ pairs, then we get all the edges of the complete graph G but the edges of the matching G_m. But among all these $\binom{n/2}{2}$ pairwise distances, we are interested in $\frac{n}{4}$ matched pairwise distances as described in line 16 of Algorithm 1, while calculating the error due to all Type-3 schedule blocks. So, the total error due to Type-3 schedule blocks is bounded by

$$2 * \frac{n/4}{\binom{n/2}{2}} * (W_t - W_m) < \frac{2}{n-2} * (\text{Lower Bound of TTP-2}).$$

■

Theorem 2.2 *All the Type-2 and Type-3 schedule blocks together introduces relative error at most* $\frac{\lceil \log_2 \frac{n}{4} \rceil + 4}{2(n-2)}$ *times of the Lower Bound of TTP-2.*

Proof. Suppose a Type-2 schedule block is designed among 4 vertices of G i.e., A_1, A_2, B_1, B_2 where A_1 and A_2 are matched pairs in G_m and so are B_1 and B_2. For a Type-2 schedule in between them, travel for each team are given below:

$A_1 : A_1 \to B_1 \to A_1 \to B_2 \to A_1$
$A_2 : A_2 \to B_2 \to A_2 \to B_1 \to A_2$
$B_1 : B_1 \to A_2 \to A_1 \to B_1$
$B_2 : B_2 \to A_1 \to A_2 \to B_2$.

So the total distance traveled is,

$$3\delta(A_1, B_1) + 3\delta(A_2, B_1) + 2\delta(A_1, A_2) + 3\delta(A_1, B_2) + 3\delta(A_2, B_2).$$

For the minimum travel schedule the value is,

$$2\delta(A_1, B_1) + 2\delta(A_2, B_1) + 2\delta(A_1, A_2) + 2\delta(A_1, B_2) + 2\delta(A_2, B_2) + 2\delta(B_1, B_2).$$

So the extra amount of travel is,

$$\delta(A_1, B_1) + \delta(A_2, B_1) + \delta(A_1, B_2) + \delta(A_2, B_2) - 2\delta(B_1, B_2).$$

Which is upper bounded by,

$$\delta(A_1, B_1) + \delta(A_2, B_2) + \delta(A_1, B_2) + \delta(A_2, B_1).$$

Let us denote the pairwise distance $D_P(A, B)$ between pairs A_1, A_2 and B_1, B_2 as,

$$\delta(A_1, B_1) + \delta(A_2, B_2) + \delta(A_1, B_2) + \delta(A_2, B_1).$$

Now, there are $\frac{n}{2}$ numbers of pair of vertices like A_1, A_2. If we consider all pairwise distances between all these $\frac{n}{2}$ pairs, then we get all the edges of the complete graph G but the edges of the matching G_m. But among all these $\binom{n/2}{2}$ pairwise distances, we have already selected $\frac{n}{4}$ pairwise distances as described in the proof of Theorem 2.1 and now we are interested in at most F_n, given in equation 2.1, pairwise distances as per line 12 or 14 of Algorithm 1, while calculating the error due to all Type-2 schedule blocks. So, the total error due to Type-2 and Type-3

schedule blocks is bounded by,

$$\frac{\frac{n}{8} * \lceil \log_2 \frac{n}{4} \rceil + \frac{n}{2}}{\binom{n/2}{2}} * (W_t - W_m) < \frac{\lceil \log_2 \frac{n}{4} \rceil + 4}{2(n-2)} * (\text{Lower Bound of TTP-2}).$$

■

Theorem 2.3 *Our algorithm gives better approximation than existing best result for number of participating teams less than or equal to* 32*.*

Proof. From the last two theorems we can see that the approximation factor in our algorithm is $1 + \frac{\lceil \log_2 \frac{n}{4} \rceil + 4}{2(n-2)}$ and in the existing best result the approximation factor is $1 + \frac{2}{n-2} + \frac{2}{n}$ [4]. So for $n \leq 32$,

$$\frac{8}{n} \leq \left\lfloor \log_2 \frac{64}{n} \right\rfloor \iff \frac{8}{n} \leq 4 - \left\lceil \log_2 \frac{n}{4} \right\rceil \iff \left\lceil \log_2 \frac{n}{4} \right\rceil \leq 4 - \frac{8}{n}$$
$$\iff \left\lceil \log_2 \frac{n}{4} \right\rceil \leq \frac{4}{n}(n-2) \iff \frac{\lceil \log_2 \frac{n}{4} \rceil}{(n-2)} \leq \frac{4}{n}$$
$$\iff \frac{\lceil \log_2 \frac{n}{4} \rceil}{2(n-2)} \leq \frac{2}{n} \iff \frac{4 + \lceil \log_2 \frac{n}{4} \rceil}{2(n-2)} \leq \frac{2}{n-2} + \frac{2}{n}.$$

This proves the theorem. ■

2.7 Concluding Remarks

- In this chapter, a better approximation factor than the existing best result has been achieved for the Traveling Tournament Problem with a maximum trip length of two with our scheduling algorithm when the number of participating teams is less or equal to 32.

- Due to time constraints and other factors, most of the tournaments involving a number of teams more than 32 are not round-robin tournaments. For example, a round-robin tournament with 40 teams will require 78 match days, 1560 matches and 40 grounds, which demand lots of time, human support and a very long season. That is why most round-robin tournaments are conducted with less than 32 teams.

- Therefore, it can be said that for almost all practical cases, the proposed scheduling algorithm would produce a better result than the existing best result.

- One of the popular double round-robin tournament in India is **Indian Premier League (IPL)** and the number of teams involved in this tournament was 8 till the last year. This tournament is not in the TTP-2 structure now. But, if it is scheduled in the TTP-2 structure, the proposed algorithm will significantly lower the total travel distance. An improved schedule of **IPL** using the proposed scheduling algorithm is presented in Appendix-A.2. It shows a 15% decrease in total travel distance compared to the actual **IPL-2019** schedule.

- As described in our algorithm, we know the specific match days of the schedule where the Type-2 super matches or *Flips* are to be incorporated. But as we have specified the pairs of teams between whom the Type-3 super matches are to be played to minimize the total travel distance due to the Type-3 super matches, nothing of this kind is done for the *Flips*. So, a revisit in this topic can give some idea about the specific pairs of teams for minimizing the distance due to the *Flips*.

CHAPTER 3

Complexity of Traveling Tournament Problem with Trip Length More Than Three

3.1 Introduction

Sports tournaments are very popular events all over the world. A huge amount of money is involved in organizing these tournaments, and lots of revenue is generated by selling the tickets and broadcasting rights. Scheduling the matches is a very important aspect of these tournaments. Scheduling specifies the sequence of matches played by each participating team along with the venues. In a Double Round-robin Tournament, every pair of participating teams play exactly two matches between them once in both of their home venues. The Traveling Tournament Problem (TTP) asks for a double round robin schedule minimizing the total travel distance of the participating teams. The fairness condition imposes an upper bound k to the maximum number of consecutive home or away matches played by any team.

TTP was first introduced by Easton, Nemhauser, and Trick [5]. The problem bears some similarities with Traveling Salesman Problem (TSP). In fact, a reduc-

tion of the unconstrained version of TTP (or $k = \infty$) from TSP has been shown by Bhattacharyya [1] proving the basic problem to be NP-hard. When the maximum permissible length of consecutive home or away matches is set to 3, TTP is also proven to be NP-Hard by Thielen and Westphal [2]. In this chapter, the natural question has been asked, *Is TTP NP-Hard for any fixed $k > 3$?*

3.1.1 Problem Definition

Let T be the set of teams with $|T| = n$. Let Δ be a square matrix of dimension $n \times n$ whose element at i^{th} row and j^{th} column corresponds to the distance between the home venues of i^{th} and j^{th} team in T for all $i, j \leq |T|$. Δ is symmetric with diagonal terms 0 and all the distances in Δ satisfy triangle inequality.

Definition 3.1 ***Decision Version of TTP-k:*** *For a fixed natural number k, an even integer n, a given set of teams T with $|T| = n$, mutual distances matrix (Δ) and a rational number δ, is it possible to schedule a double round-robin tournament such that the total travel distance of the tournament is less or equal to δ, where no team can have more than k consecutive away matches or consecutive home matches and no team plays its consecutive matches with the same opponent.*

3.1.2 Previous Work

The Traveling Tournament Problem was first introduced by Easton, Nemhauser, and Trick [5]. Since then most of works focused on finding approximation algorithms or heuristics when $k = 3$ [25, 14, 12, 13, 9]. Thielen and Westphal [2] proved NP-hardness for TTP-3. In a series of two papers [26, 7] Imahori, Matsui, and Miyashiro showed approximation algorithms for the unconstrained TTP. Bhattacharyya [1] complemented this by proving the NP-hardness of unconstrained TTP. For other values of k, only upper bound results are known. Thielen and Westphal approximated TTP-2 [27] and improved by Xiao and Kou [4]. For TTP-k with $k > 3$, approximation algorithms are given by Yamaguchi, Imahori and Miyashiro [11] and Westphal and Noparlik [10].

Many different problems related to sports scheduling have been thoroughly analyzed in the past. For detailed surveys and studies on round-robin tournament

scheduling, the readers are referred to [22, 19, 17]. Graph Theoretic approach for solving scheduling problems can be found in [28, 3] by De Werra.

3.1.3 Approach towards the Problem

In this chapter, a generalization of the approach by Thielen and Westphal [2] has been done, which showed the NP-Hardness of TTP-3. Like them, a reduction from the satisfiability problem has been shown. While [2] showed a reduction from 3-SAT, a reduction from k-SAT is shown here. However, the reduction shown here is different in a few crucial aspects. Firstly, the construction of the reduced TTP instance graph is different from that in [2]. To accommodate trips of length k, a new graph in terms of vertices and edge weights is required. In addition, the trip structures and their synchronous assembly to get the tournament schedule differ from that of TTP-3. The reconstruction of k-SAT with specific properties of clauses requires a different technique.

3.1.4 Result

Here the main theorem is the following.

Theorem 3.1 *TTP-k for a fixed $k > 3$ is strongly NP-Complete.*

3.2 Proof of Theorem 3.1

This reduction requires that the input instance of the satisfiability problem satisfies certain properties. The first step is to show that *any* input instance can be transformed into an instance with properties required for the reduction. Following notations are used throughout the chapter. If $x \in \{0, 1\}$ is a boolean variable, $\bar{x}$ denotes its complement,i.e., $\bar{x} = 1 \oplus x$.

Lemma 3.1 *For a k-SAT instance F_k with t variables and p clauses there exists another k-SAT instance F'_k with t' variable and p' clauses such that a satisfying assignment of F_k implies a satisfying assignment of the variables of F'_k and vice-versa. Moreover, the number of occurrence of all the t' variables and there compliments are equal in F'_k, $t' \leq \left(t + \frac{k+1}{2}\right)$ and $k \mid p'$ with $k, t, t', p, p' \in \mathbb{N}$.*

Proof. Let x_i and $\bar{x}_i$ ($i \in \{1, \ldots, t\}$) be the variables in F_k. Suppose n_i and $\bar{n}_i$ are the number of occurrence of x_i and $\bar{x}_i$ in F_k respectively. Without loss of generality it can be assume that $n_i \leq \bar{n}_i$. To make $n_i = \bar{n}_i$, few clauses has been added to F_k depending on the value of k is even or odd.

1. When k is odd, clauses of the form $(x_i \vee x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k-1}{2}} \vee \bar{x}_{t+\frac{k-1}{2}})$ are added until $n_i = \bar{n}_i \ \forall i \in \{1, \ldots, t\}$ by introducing $(k-1)/2$ new variables. After adding these clauses number of clauses is even due to the *Handshaking Lemma*. Then by adding at most $(k-1)/2$ pairs of clauses of the form $(x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k-1}{2}} \vee \bar{x}_{t+\frac{k-1}{2}} \vee x_{t+\frac{k+1}{2}})$ and $(x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k-1}{2}} \vee \bar{x}_{t+\frac{k-1}{2}} \vee \bar{x}_{t+\frac{k+1}{2}})$, number of clauses can be made divisible by k keeping $n_i = \bar{n}_i, \forall i \in \{1, \ldots, t + \frac{k+1}{2}\}$.

2. When k is even, then using the *Handshaking Lemma* it can be said that there exist even number of indices $j \in \{1, \ldots, t\}$ such that $(n_j + \bar{n}_j)$ is odd. So, $\|n_j - \bar{n}_j\|$ is odd. Without loss of generality, it can be assumed that $\bar{n}_j > n_j$. By identifying two indices of this kind, namely m and n, clauses of the form $(x_m \vee x_n \vee x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k}{2}-1} \vee \bar{x}_{t+\frac{k}{2}-1})$ are added until $\|n_i - \bar{n}_i\|$ is even $\forall i \in \{1, \ldots, t\}$ by introducing $(k/2-1)$ new variables. Now $\forall j \in \{1, \ldots, t\}$ such that $n_j \neq \bar{n}_j$, pair of clauses of the forms $(x_j \vee x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k}{2}-1} \vee \bar{x}_{t+\frac{k}{2}-1} \vee x_{t+\frac{k}{2}})$ and $(x_j \vee x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k}{2}-1} \vee \bar{x}_{t+\frac{k}{2}-1} \vee \bar{x}_{t+\frac{k}{2}})$ are added until $n_j = \bar{n}_j \ \forall j \in \{1, \ldots, t\}$. Then by adding at most $(k-1)$ clauses of the form $(x_{t+1} \vee \bar{x}_{t+1} \vee \cdots \vee x_{t+\frac{k}{2}} \vee \bar{x}_{t+\frac{k}{2}})$, the number of clauses can be made divisible by k keeping $(n_i = \bar{n}_i) \ \forall i \in \{1, \ldots, t + \frac{k}{2}\}$.

Let the resulting *k-SAT* problem expression be F'_k with t' variables and p' clauses where, $k|p'$. In both the cases explained above, all the additional clauses always give truth values. So, F'_k will have a truth assignment of variables x_i for all $i \in \{1, \ldots, t'\}$ if and only if F_k have a truth assignment of variables x_i $\forall i \in \{1, \ldots, t\}$. This proves the lemma. ■

3.2.1 TTP-k is NP-Complete

The first step is to show that TTP-k is indeed in NP.

Lemma 3.2 *TTP-k is in NP.*

Proof. In a decision version of *TTP-k* with given set T of n teams, and the constraint k it is verifiable in $O(n^2)$ whether a schedule is valid and gives a total travel distance less than δ or not. This ensures the membership of *TTP-k* in *NP*. ■

Next, a reduction from *k-SAT* to *TTP-k* has been shown. For this, a specially weighted graph $G = (V, E)$ has been constructed with one or more teams situated at each vertex of G and a predefined value δ of total travel distance such that there is a satisfying assignment of variables for *k-SAT* if and only if there is a *TTP-k* schedule between the teams in G with total travel distance less than δ. First, the input *k-SAT* problem is modified using Lemma 3.1 such that the resulting formula has the following properties.

(a) There are t variables $x_1, x_2, \ldots, x_t$.

(b) Number of occurrence of x_i is equal to the number of occurrence of $\bar{x}_i$, i.e., $n_i = \overline{n}_i$.

(c) Number of clauses p is divisible by k, i.e., $k|p$.

3.2.1.1 The Construction

We start with the construction of the reduced instance graph G. Recall that k is the upper bound of the number of consecutive home or away matches and n_i is the number of occurrences of the variable x_i. The main part of the graph is the union of t many sub-graphs $G_1, G_2, \cdots, G_t$, where t is the number of variables in the (modified) input SAT instance. Each G_i consists of $(k+1)n_i$ vertices, where,

(a) n_i many vertices are denoted by $x_{i,j}$ and n_i many vertices are denoted by $\bar{x}_{i,j}$, where $j \in \{1, \ldots, n_i\}$.

(b) For $j \in \{1, \ldots, n_i\}$, the vertices $y_{i,j}$ denote n_i many vertices.

(c) For $j \in \{1, \ldots, n_i\}$ and $l \in \{1, \cdots, k-2\}$, the vertices $w^l_{i,j}$ are the remaining $(k-2)n_i$ many vertices.

In addition, there are $(k-1)p + 1$ many vertices in the graph where p is the number of clauses of the (modified) input SAT instance. There is a central vertex

v. Then there are p vertices of the form C_m and $(k-2)p$ vertices of the form z_m^l $\forall l \in \{1,\ldots,k-2\}$ and $\forall m \in \{1,\ldots,p\}$.

For ease of explanation, we summarize the important parameters related to G. The total number of vertices in G is $\left[\left(\sum_{i=1}^{t}(k+1)n_i\right)+(k-1)p+1\right]$. We also know that, $\sum_{i=1}^{t} 2n_i = kp$. So, The total number of vertices other than v in G is $\left(\frac{k(k+1)}{2}+k-1\right)p$, which we denote by a, i.e.,

$$a = \left(\frac{k(k+1)}{2}+k-1\right)p = p\left(\frac{k^2+3k-2}{2}\right).$$

WEIGHTS OF THE EDGES. Let weight $M = \theta(a^5)$ is assigned to the edges from v to the vertices $x_{i,j}$ and $\bar{x}_{i,j}$, $(M-2)$ to the edges from v to $y_{i,j}$ and $(M-2k+4)$ to edges from v to $w_{i,j}^{k-2}$ of G_i for every $j \in \{1,2,\ldots,n_i\}$. Then $x_{i,j}$ is connected with $w_{i,j}^{k-2}$ through $k-3$ vertices serially namely $w_{i,j}^1, w_{i,j}^2, \ldots, w_{i,j}^{k-3}$, where each of these consecutive vertices in this serial connection is connected with each other with an edge of weight 2. Then $\bar{x}_{i,j}$ is connected to $w_{i,q}^1$ with an edge of weight 2 where $q = j(mod\ n_i)+1$ and also $y_{i,j}$ is connected to both $x_{i,j}$ and $\bar{x}_{i,j}$ with edges of weight 2 each as described in Figure 3.1.

For the connection between the remaining vertices in G, first $x_{i,j}$ or $\bar{x}_{i,j}$ are connected to c_m with an edge of weight 2 if x_i or $\bar{x}_{i,j}$ has its j^{th} occurrence in the m^{th} clause of the modified *k-SAT*. Then c_m is connected with z_m^{k-2} through $k-3$ vertices serially namely $z_m^1, z_m^2, \ldots, z_m^{k-3}$, where each of these consecutive vertices in this serial connection is connected with each other with an edge of weight 2. At last z_m^{k-2} and c_m is connected to v by edges of weights $(M-2k+6)$ and $(M+2)$ respectively as described in Figure 3.2.

Formally, weights of all the edges of G are listed as follows:

- Weight$(w_{i,j}^r, w_{i,j}^s)$=2$\|r-s\|$ for all $i \in \{1,\ldots,t\}$, for all $j \in \{1,\ldots,n_i\}$, for all $r,s \in \{1,\ldots,k\text{-}2\}$.

- Weight$(x_{i,j}, w_{i,j}^s)$=2s for all $i \in \{1,\ldots,t\}$, for all $j \in \{1,\ldots,n_i\}$, for all $s \in \{1,\ldots,k\text{-}2\}$.

- Weight$(\bar{x}_{i,j}, w_{i,q}^s)$=2s for all $i \in \{1,\ldots,t\}$, for all $j \in \{1,\ldots,n_i\}$, for all $s \in \{1,\ldots,k\text{-}2\}$ and $q = j(mod\ n_i)+1$.

- Weight$(x_{i,j}, y_{i,j})$=Weight$(\bar{x}_{i,j}, y_{i,j})$=2 for all $i \in \{1, \ldots, t\}$, for all $j \in \{1, \ldots, n_i\}$.
- Weight(c_m, z_m^r)=2r for all $r \in \{1, \ldots, k-2\}$ and for all $m \in \{1, \ldots, p\}$.
- Weight(z_m^r, z_m^s)=2$\|r - s\|$ for all $r, s \in \{1, \ldots, k-2\}$ and for all $m \in \{1, \ldots, p\}$.
- Weight$(x_{i,j}, c_m)$=2, if j^{th} occurrence of x_i is present in m^{th} clause of the given k-SAT expression.
- Weight$(\bar{x}_{i,j}, c_m)$=2, if j^{th} occurrence of $\bar{x}_i$ is present in m^{th} clause of the given k-SAT expression.
- Weight$(v, w_{i,j}^{k-2})$=M-2k+4 for all $i \in \{1, \ldots, t\}$, for all $j \in \{1, \ldots, n_i\}$.
- Weight$(v, y_{i,j})$=M-2 for all $i \in \{1, \ldots, t\}$, for all $j \in \{1, \ldots, n_i\}$.
- Weight$(v, x_{i,j})$=M for all $i \in \{1, \ldots, t\}$, for all $j \in \{1, \ldots, n_i\}$.
- Weight$(v, \bar{x}_{i,j})$=M for all $i \in \{1, \ldots, t\}$, for all $j \in \{1, \ldots, n_i\}$.
- Weight(v, z_m^{k-2})=M-2k+6 for all $m \in \{1, \ldots, p\}$.
- Weight(v, c_m)=M+2 for all $m \in \{1, \ldots, p\}$.

All other edges in the complete graph G are given the maximum possible weights without violating triangle inequality.

Creating TTP-k instance. Now, the teams are placed on the vertices of G to construct the reduced instance.

- Total number of teams is equal to $a^3 + a$.
- At each vertex of G except v, only one team is placed. This set of vertices or teams is denoted as U.
- a^3 teams are situated at v and distance between them is 0 and call this set of vertices or teams T.

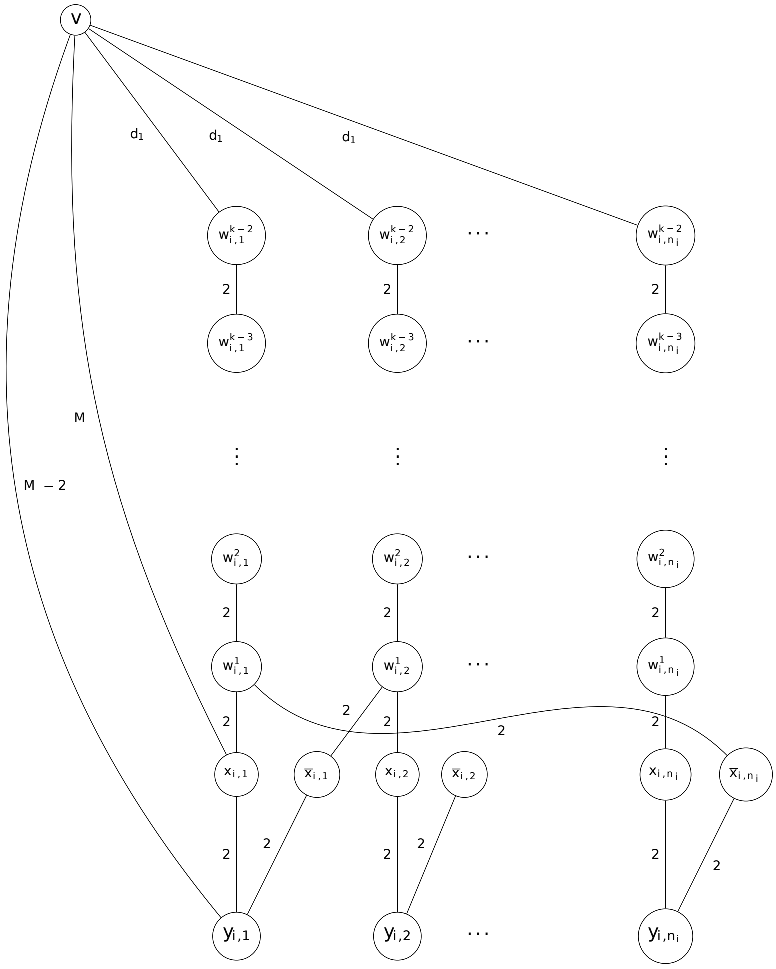


Figure 3.1: Sub-graph of G for the i^{th} variable, where $d_1 = M - 2k + 4$

We fix,

$$\delta = M\left[\frac{4a^4}{k}\right] + pa^3(k^2 - 3k + 6) + 2a(a-1)k + \frac{4a^4}{k}.$$

Lemma 3.3 *Weights of the edges of G preserve the Triangle Inequality.*

Proof. For a tuple $(v, x_{i,j}, y_{i,j})$ or $(v, \bar{x}_{i,j}, y_{i,j})$, for all $i \in \{1, \ldots, t\}$ and $j \in$

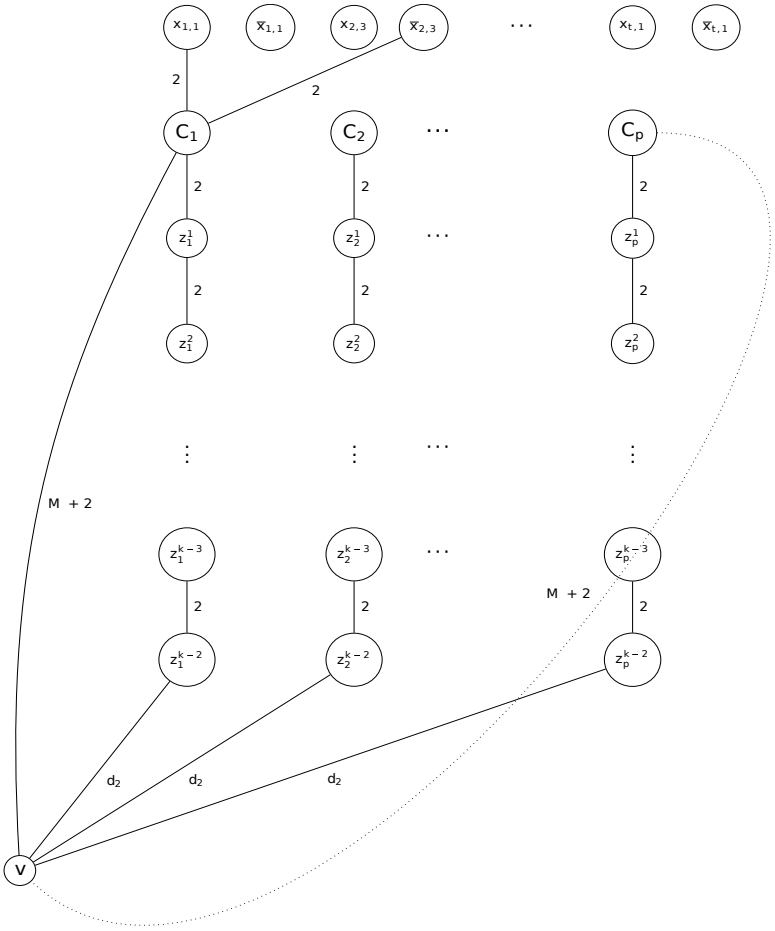


Figure 3.2: Sub-graph Corresponding to First Clause of the form $C_1 = (x_1 \vee \bar{x}_2 \vee \dots)$, where $d_2 = M - 2k + 6$

$\{1, \dots, n_i\}$, the triangle inequality is preserved as,

$$Weight(v, x_{i,j}) = M = Weight(x_{i,j}, y_{i,j}) + Weight(v, y_{i,j}) = 2 + (M - 2),$$
$$Weight(v, \bar{x}_{i,j}) = M = Weight(\bar{x}_{i,j}, y_{i,j}) + Weight(v, y_{i,j}) = 2 + (M - 2).$$

For a tuple $(v, x_{i,j}, w^1_{i,j})$ or $(v, \bar{x}_{i,j}, w^1_{i,j})$, for all $i \in \{1, \dots, t\}$ and $j \in \{1, \dots, n_i\}$, the triangle inequality is preserved as,

$$Weight(v, x_{i,j}) = M = Weight(x_{i,j}, w^1_{i,j}) + Weight(v, w^1_{i,j}) = 2 + (M - 2),$$
$$Weight(v, \bar{x}_{i,j}) = M = Weight(\bar{x}_{i,j}, w^1_{i,j}) + Weight(v, w^1_{i,j}) = 2 + (M - 2).$$

For a tuple $(v, x_{i,j}, c_m)$ or $(v, \bar{x}_{i,j}, c_m)$, where j^{th} occurrence of x_i or $\bar{x}_i$ is present in the m^{th} clause of the given *k-SAT* expression, the triangle inequality is preserved as,

$$Weight(v, c_m) = M + 2 = Weight(x_{i,j}, c_m) + Weight(v, x_{i,j}) = 2 + M,$$
$$Weight(v, c_m) = M + 2 = Weight(\bar{x}_{i,j}, c_m) + Weight(v, \bar{x}_{i,j}) = 2 + M.$$

The triangle inequality for all other tuples of three vertices in G is followed from these three cases as the weights are given maximum possible values without violating triangle inequality while assigned. ■

3.2.2 The Reduction

As a desired value of δ has been derived, now the only remaining part is the reduction. First, we show that a given satisfying assignment of variables in a *k-SAT* $\Rightarrow$ a *TTP-k* schedule of total travel distance less than δ. The tours for the a^3 vertices situated at v are constructed and we then show that, these tours are so cheap in terms of travel distance that tours of similar structure must be there in a tournament where the total travel distance is desired to be as low as possible. So, $\frac{a}{k}$ node disjoint tours are constructed for a team at v where, all the vertices $x_{i,j}, \bar{x}_{i,j}, y_{i,j}, w^r_{i,j}, c_m, z^r_m$ are visited. It is given that there is a satisfying assignment of variables in the *k-SAT*. Let us define two conditions denoting with the value of a variable b_m in the following manner,

$b_m = 1 \implies \exists\, i \in \{1, \ldots, t\}$ such that $x_i = 1$ & x_i appears in the m^{th}clause.

$b_m = 0 \implies \exists\, i \in \{1, \ldots, t\}$ such that $x_i = 0$ & $\bar{x}_i$ appears in the m^{th}clause.

For all $m \in \{1, \ldots, p\}$, if $b_m = 1$ then Weight$(x_{i,j}, c_m) = 2$. Now, to visit a team at c_m only, a team in v has to travel $2(M + 2)$ distance. But if it travels to a vertex $x_{i,j}$, then to c_m and travel through z^1_m to z^{k-2}_m and return to v, it travels the same distance, $2(M + 2)$, and also visits k vertices in a single trip which is a desired situation here. Afterwards in another trip to a vertex $\bar{x}_{i,j}$, in spite of a to and fro journey to $\bar{x}_{i,j}$ only, i.e., $2M$, if it travels to $w^{k-2}_{i,q}$ first then through all $w^s_{i,q}$

for $q = j(mod\ n_i) + 1$ and $s \in \{1, \ldots, k-3\}$ to $\bar{x}_{i,j}$ and $y_{i,j}$ and returns back to v, the travel distance is same as $2M$ and also k vertices will be visited in a single trip. Multiple trip to these extra $(k-1)$ vertices would cost much more in comparison. Similar tours are taken when $b_m = 0$ only interchanging $x_{i,j}$ and $y_{i,j}$ with $\bar{x}_{i,q}$ and $y_{i,q}$ respectively. This leaves $(k-2)p/2$ number of $x_{i,j}$ type vertices to visit. As $k|p$, this can be done in $p(k-2)/2k$ trips each of length k and travel distance less than $2(M + k(k-1))$. So the total distance traveled by a team situated at v is upper bounded by δ_1, where,

$$\begin{aligned}\delta_1 &= 2(M+2)p + Mkp + M(k-2)p/k + (k-1)(k-2)p \\ &= p\left[M\left(k+3-\frac{2}{k}\right) + k^2 - 3k + 6\right].\end{aligned}$$

With tours involving distance M is minimized by covering exactly k vertices in each of the tours. Now these tours can be numbered from 1 to $\frac{a}{k}$ and vertices in each tour can be numbered from 1 to k. So, all the a vertices of U are named as $u_{i,j}$ for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, \frac{a}{k}\}$ such that $u_{i,j}$ is the i^{th} visited team of the j^{th} tour. Also the vertices in T are partitioned in a^2 disjoint sets $T_1, T_2, \ldots, T_{a^2}$ each of size a. Moreover, $T_q = \{t_{r,q}$ such that $r \in \{1, \ldots, a\}\}$. The tours by the teams in T are now designed in such a way that, $t_{1,1}$ will take tour number 1, i.e., travel through $u_{1,1}, u_{2,1}, \ldots, u_{k,1}$. Then, $u_{1,1}, u_{2,1}, \ldots, u_{k,1}$ visit $t_{1,1}$ in the same order. Similarly, for all $i \in \{2, \ldots, k\}, t_{i,1}$ follows the same tour and visited by the same teams as $t_{1,1}$ but with a time delay of $(i-1)$. This way all the teams in T_1 first complete visits to all the teams in tour 1 and then get visited by them. Then T_1 starts tour 2. This way teams in T_1 plays with the teams in tour 1 to tour $\frac{a}{2k}$ in such a way that the teams in T_1 visit first and then they get visited. Similarly, teams in T_q for all $q \in \{1, \ldots, \frac{a^2}{2}\}$ follow a similar travel like T_1. The matches between teams in T_q for all $q \in \{\frac{a^2}{2} + 1, \ldots, a^2\}$ and the teams in U that are in tour j, for all $j \in \{\frac{a}{2k} + 1, \ldots, \frac{a}{k}\}$ are also played in a similar fashion with the change that the teams in U visit the teams in T first and then they get visited by the teams in T. This way the sets T and U both are divided in two

parts according to the teams they completed playing. Formally,

$$T_a = \bigcup_{q=1}^{a^2/2} T_q, \qquad T_b = \bigcup_{q=a^2/2+1}^{a^2} T_q, \qquad U_a = \bigcup_{j=1}^{a/2k} S_j, \qquad U_b = \bigcup_{j=a/2k+1}^{a/k} S_j.$$

Afterwards by changing the roles of T_a and T_b, matches between teams in T_a and U_b are arranged and similarly between teams in T_b and U_a. So, the main remaining part is schedule of matches between the teams in U and that of the teams in T.

For scheduling matches between teams in U, the teams in U are categorized in k categories depending on their occurrence in the tours, i.e., for all $i \in \{1, \ldots, k\}, U_i = \{u_{i,j}$ such that $j \in \{1, \ldots, \frac{a}{k}\}\}$. For all i, the teams in U_i play against the teams in T at the same slots but half of them play at home and the other half play away. More specifically the teams in U_i play with the teams in T at exactly on the slots $(2i-1)$ to $(2a^3+2i-2)$. Keeping these busy slots in mind, the schedule of a single round-robin tournament of the teams in U is designed using the canonical tournament introduced by de Werra[3]. This is done by assigning a vertex to each of the teams in a U_i in a special graph as done in the canonical tournament design. This tournament structure gives assurance that each team plays every other team exactly once and no team has a long sequence of home and away matches. Here a match between i and j signifies a match between a team in U_i and a team in U_j. In the end, the same tournament is repeated with changed match venues, i.e., nullification of home field advantage.

Now the only remaining part is scheduling of the matches between the teams in T. Let $d(t)$, $\forall\, t \in T$ denote the first slot on which team t plays a team in U and let $T_i = \{t \in T$ such that $((d(t)-1) \bmod k) + 1 = i\}$, for all $i \in \{1, \ldots, k\}$. Then every T_i is partitioned into $2a^2$ groups of cardinality $\frac{a}{2k}$ such that $d(t_1) = d(t_2)$ for every two members t_1, t_2 of the same group. For every T_i, the matches between teams in different groups in T_i are scheduled. Among the teams in T_i, $\frac{a}{k}$ will always be busy playing with some teams of U. To handle this fact, two dummy groups U_1 and U_2, as defined before, are introduced that play with these busy $\frac{a}{k}$ teams of T_i. Then each group is treated as a team and again canonical tournament structure is applied only skipping the day at which the two dummy groups meet.

For scheduling the matches between the members of two groups g and h of T_i

in l rounds, where $g = \{g_1, g_2, \ldots, g_l\}$ and $h = \{h_1, h_2, \ldots, h_l\}$ the following steps are done. The i^{th} round contains the matches between h_j and $g_{((i+j) \mod l)+1}$ for all $i, j \in \{1, 2, \ldots, l\}$ with game taking place at the home of h_j. This restricts the lengths of away trips and home stands for all the teams from being long. Then the same schedule is repeated with altered venues. Matches between the teams of some group of T_i with the teams of a dummy group has already been taken care when the matches between U and T were designed.

Now, for the scenario where the two dummy groups meet, two kinds of matches are there which differ in length. The encounter between two groups consists of $\frac{a}{k}$ slots, while the encounter between two dummy groups, i.e., U_1 and U_2 consists of a slots. The encounters between the groups of T_i are scheduled in the usual way using $\frac{a}{k}$ slots and the extra $\frac{(k-1)a}{k}$ slots are used to schedule matches between the teams in different T_i's.

To schedule matches between the teams in different T_i's, first each T_i is partitioned in two equal size, namely $T_{i,1}$ and $T_{i,2}$. Now, considering each $T_{i,j}$ as a single team for all $i \in \{1, 2, \ldots, k\}$ and all $j \in \{1, 2\}$, a canonical tournament is again applied on these teams skipping the day at which $T_{i,1}$ encounters $T_{i,2}$ for all i. This scenario can be achieved by properly initializing the canonical tournament. Then the same schedule is repeated with altered venues to nullify home advantage as done in all earlier canonical tournament structures.

For a team situated in one of the vertex in $E \setminus v$ to visit the teams in v, it has to travel at most $\frac{2(M+2)a^3}{k}$ using $\frac{a^3}{k}$ trips of length k. So, the total travel distance of all the teams in $E \setminus \{v\}$ to the teams in v can be bounded by δ_2, where,

$$\delta_2 = \frac{2(M+2)a^4}{k}.$$

As all the distances between the vertices in $E \setminus \{v\}$ in G is less or equal to $2k$, the travel between all the teams in $E \setminus \{v\}$ can be bounded above by δ_3, where,

$$\delta_3 = 2a(a-1)k.$$

The teams in T visit each other at zero cost as they are situated at the same point. So, the total travel distance of the tournament is bounded by,

$$\begin{aligned}
&\delta_1 \cdot a^3 + \delta_2 + \delta_3 \\
&= a^3 p \left[M \left(k + 3 - \frac{2}{k} \right) + k^2 - 3k + 6 \right] + \frac{2(M+2)a^4}{k} + 2a(a-1)k \\
&= \delta.
\end{aligned}$$

In another way,

$$\begin{aligned}
\delta \quad &= Ma^3 \left[p \left(k + 3 - \frac{2}{k} \right) + \frac{2a}{k} \right] + pa^3(k^2 - 3k + 6) + 2a(a-1)k + \frac{4a^4}{k} \\
&= M \left[\frac{4a^4}{k} \right] + pa^3(k^2 - 3k + 6) + 2a(a-1)k + \frac{4a^4}{k}.
\end{aligned}$$

This completes the first direction of the proof.

3.2.3 Proof of the other direction

For the other direction of the proof, it has to be shown that *k-SAT* is not satisfiable implies that a *TTP-k* schedule of total travel distance less than δ is not possible. Here we are given that there is no satisfying assignment of variables of the *k-SAT*. In the forward direction of the proof explained above, it is shown that the proposed schedule is compact and gives an optimized travel distance value for the teams' tours in T to the teams in U. But the travels are designed depending on a truth assignment of x variables. The travel to c_m vertex goes through a $x_{i,j}$ or $\bar{x}_{i,j}$ which is there in the m^{th} clause of the *k-SAT* and assigned with value 1. Also, the other variable among these two covers $y_{i,j}$ and $w^s_{i,j}$ for all i, j together in another tour. Let's assume that there exist optimum tours similar to the forward direction of the proof, although there is no satisfying assignment of variables of the *k-SAT*. So, there is an optimum path through each c_m for all $m \in \{1, \ldots, p\}$ that includes a vertex $x_{i,j}$ or $\bar{x}_{i,j}$ which is present in the m^{th} clause of the *k-SAT* expression. Now, if value 1 is assigned to each of these variables, then that must end in a wrong assignment of variables as it contradicts the assumption otherwise. This implies that $x_{i,j} = \bar{x}_{i,j} = 1$ has been assigned for some $i \in \{1, \ldots, t\}$ and $j \in \{1, \ldots, n_i\}$. That means both $x_{i,j}$ and $\bar{x}_{i,j}$ are included in optimized tours from v to c_i and

c_j, where $i, j \in \{1, \ldots, p\}$. Now, it will not be possible to design a trips from v, that includes $y_{i,j}, w^1_{i,j}, \ldots, w^{k-2}_{i,j}$, for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, \frac{a}{k}\}$ using the optimized trips. So, it is not possible to cover all the $c_m, y_{i,j}, w^s_{i,j}$ for all i, j along with all $x_{i,j}$ for all i, j using these optimized tours. To cover all the vertices in U, each of the a^3 vertices of T has to tour at least once more to the vertices of U. These extra tours will increase the total travel distance by at least $2 \cdot M \cdot a^3$. As the total travel distance by the teams in U for matches among them is $O(a^2)$, the tours of the teams of T to those of U dictate the total travel distance of the tournament. So, an increase in this part will increase the total distance significantly and for M being $\theta(a^5)$, the total travel distance will be more than δ. This completes the other direction of the proof and the reduction.

3.3 Concluding Remarks

- In this chapter, the general Traveling Tournament Problem-k for any natural number $k > 3$ has been proven to be NP-Hard.
- In literature, TTP-3 and TTP-∞ has been proven to be NP-Hard. So the result in this chapter closes the complexity analysis of the Traveling Tournament Problem.

CHAPTER 4

Firefighter Problem with Minimum Budget: Hardness and Approximation Algorithm for Unit Disk Graphs

4.1 Introduction

Firefighter Problem on a graph is broadly described by saving vertices of a graph from fire by placing firefighters on them. Suppose there is a graph $G(V, E)$. Fire breaks out at one of the vertex $s \in V$. The vertex s is called the source of the fire. Now at each time instance, the fire spreads to the neighbors of all the vertices on fire unless a firefighter is placed on them to stop it. Firefighter Problems can be classified into two main categories depending on the objective, i.e., Max-Save and Min-Budget. In the Max-Save version, the goal is to save the maximum number of vertices possible. In contrast, in the Min-Budget version, the goal is to save a given set of vertices using the minimum number of firefighters at each time instance. The formal definitions are given below.

Definition 4.1 ***Firefighter Max-Save Problem***
Input: *Graph $G(V, E)$, source vertex of fire $s \in V$ and maximum number of firefighters allowed to be placed at each time instance $d \in \mathbb{N}$.*
Question: *At most how many vertices can be saved from fire and what is the strategy?*

Definition 4.2 ***Firefighter Min-Budget Problem***
Input: *Graph $G(V, E)$, source vertex of fire $s \in V$ and a set of vertices to be saved $T \subset V$.*
Question: *What is the minimum number of firefighter required at each time instance to save the vertices in T?*

The firefighter problem has very important relevance to many real-life situations. Suppose a malicious program or virus is spreading through a communication network and our goal is to prevent some nodes in the network from this virus by using some preventive measures for some nodes. This real-life scenario can be seen as a firefighter problem on a graph where the virus is like a spreading fire, the preventative measure can be thought of as firefighters and the graph has the same topology as the communication network. In another case, suppose sensitive data or information has been leaked in a communication network and the flow of the information to some vulnerable or less secured nodes is needed to be restricted. This scenario also can be modeled as a firefighter problem, where the information can be seen as the spreading fire and the vulnerable nodes as the vertices to be saved using firefighters. Both the versions of the firefighter problem can be solved in polynomial time for interval graphs, split graphs and permutation graphs. The case of unit disk graphs is considered here.

Unit Disk Graph represents the intersection of the disk graph and interval graph. For each vertex u in the graph, there is a disk of radius 1 in the Euclidean plane. There is an edge between vertex u and v if v is inside the unit disk centered at u.

Definition 4.3 *A **Unit Disk Graph** is an undirected graph formed by correspondence with a set of disks of unit radius on a euclidean plane, where each disk corresponds to a vertex in the graph and two vertices have an edge in between them if their corresponding disks intersect in the euclidean plane.*

Unit Disk Graphs are relevant to communication networks. In almost all practical wireless communication networks, the routers have the same communication range, which can be modeled as disks of coverage area similar to a unit disk graph. Also, measures have to be taken to prevent any virus or sensitive information from reaching a node, which is the same as placing firefighters on some vertices in the firefighter problem. In a practical scenario of communication networks, the goal is to keep the sensitive data away from some specific nodes in the network, given the network topology. This case resembles the MIN-BUDGET version of the Firefighter Problem, where the source vertex of fire and a set of vertices to be saved from the fire are given. At each instance of time, the minimum number of firefighters is to be placed to meet the objective of saving the given set of vertices.

4.1.1 Our Results

In this chapter, NP-Hardness of Firefighter Min-Budget Problem on Unit Disk Graphs has been shown and also a 2-approximation algorithm for the problem has been presented by proving the following two theorems.

Theorem 4.1 *The Firefighter MIN-BUDGET problem on unit disk graph is NP hard.*

Theorem 4.2 *There exists an approximation algorithm of Firefighter MIN-BUDGET Problem on unit disk graph with a constant approximation factor of 2.*

4.1.2 Previous Works

The Firefighter problem has a rich history of work. Some results which are most relevant to our work are mentioned. For a detailed account, the readers are referred to the survey on results, directions and open questions related to firefighter problems by Finbow and MacGillivray [29]. The firefighter problem was introduced by Hartnell [30] who studied the way to protect an infinite graph from a spreading virus. Hartnell [31] showed that a simple 2-approximation algorithm exists for Firefighter Max-Save Problem. Wang and Moeller [32] showed 2 firefighters are required to control the fire in a finite 2-dimensional grid with a minimum of 8

steps and (r-1) firefighters are required to save an r-regular graph. Develin and Hartke [33] further added that at least 18 vertices would burn to do so and (2n-1) firefighters can hold a fire in an n-dimensional grid if a fire breaks out in a single vertex. MacGillivray and Wang [34] proved the NP-hardness of firefighter problem on Bipartite graphs and gave a polynomial time algorithm for the same problem on P-trees and also gave an integer programming to determine an optimal sequence of vaccinating the vertices. This work was followed by the work by Hartke [35] to narrow the integrality gap between the integer programming optimal and the optimal of the linear programming relaxation. Finbow, King, MacGillivray and Rizzi [36] showed that the Max-Save version of the firefighter problem is NP-hard on trees with a maximum degree 3. However, suppose a fire breaks out at a vertex with degree 2. In that case, the problem is in P. Chlebikova and Chopin [37] showed the complexity of firefighter problem is governed by path width and maximum degree of a graph. Work on parameterized complexity of firefighter problems can be found in the works by Bazgan, Chopin and Fellows [38] and Cygan, Fomin and Van Leeuwen [39]. Anshelevich, Chakrabarty, Hate and Swamy [40] considered both spreading and non-spreading vaccination models of firefighter problem to show the NP-hardness of approximation of the problem within $n^{1-\epsilon}$ for any ϵ. Fomin, Heggernes and Van Leeuwen [41, 42] considered the Max-Save Firefighter problem and gave a polynomial time algorithm for interval graphs, split graphs, permutation graphs and proved NP-hardness of the problem on Unit Disk Graphs. Online firefighter problem and fractional firefighter problem were introduced by Coupechoux, Demange, Ellison and Jouve [43] and they also gave the optimal competitive ratio and asymptotic comparison of the number of firefighters and the size of the tree levels. In another work, Clark, Colbourn and Johnson showed the NP-completeness of different graph problems on unit disk graphs [44].

4.1.2.1 Previous Works on the Min-Budget version

Resource Minimization for Fire Containment is similar to the Min-Budget version of the Firefighter Problem. King and MacGillivray [45] showed that Resource Minimization for Fire Containment (RMFC) is NP-hard on full trees of degree

3. They also showed that the 2-approximation is hard as well, but on graphs of maximum degree 3, if a fire breaks out at a vertex of degree 2, the problem is in P. In a follow-up work of Chalermsook and Chuzhoy [46], the authors showed an $O(\log n)$-approximation LP-rounding algorithm for RMFC on trees.

4.2 Notation and Preliminaries

The following notations are used throughout this chapter.

- $|S|$ denotes the size of the set S.
- In the Euclidean space, $d_{\ell_2}(u, v)$ denotes the ℓ_2 norm (i.e., Euclidean distance) of two points u and v.
- For an undirected, connected, simple graph G, (G, s) denotes an instance of the firefighter problem with s as the source vertex of fire.
- V_G and E_G denote the vertex set and the edge set of G respectively.

Definition 4.4 ***Decision Version of Min-Budget Firefighter Problem in Unit Disk Graph***
Input*: A unit disk graph $G(V, E)$, a vertex $s \in V$, a subset $T \subseteq V$ and $B \in \mathbb{N}$.*
Question*: Can all the vertices of T be saved by placing at most B number of firefighter at each time instance when the fire starts from s?*

Definition 4.5 ***Interval Graph***
An interval Graph is an undirected graph formed by correspondence with a set of intervals on real line, where each interval corresponds to a vertex in the graph and two vertices have an edge in between them if their corresponding intervals intersect.

In the field of economics, interval graphs and unit disk graphs represent the resource allocation networks. One use case of firefighter problem on interval graph or unit disk graph is when we need to stop the distribution of some malicious product through a resource allocation network.

Our approximation algorithm for Firefighter Min-Budget Problem on Unit Disk Graph uses the algorithm for Firefighter Min-Budget Problem on Interval Graphs.

4.3 Proof of Theorem 4.1: the Reduction

A reduction is shown from the Firefighter Problem for Trees with a maximum degree three, which is NP-hard. The proof idea is similar to the proof of the MAX-SAVE version by Fomin et al. [42]. The first step is to transform an instance of the MIN-BUDGET Firefighter problem on Trees with a maximum degree three to an instance of the MIN-BUDGET firefighter problem on a unit disk graph. To do that, an embedding of any full rooted tree of degree three into a unit disk graph is shown in following section.

4.3.1 The Construction

Let (T, Γ, B) be the input instance of the firefighter problem on Trees with a maximum degree three, where Γ denotes the set of vertices to be saved and B denotes the budget. The first step is to embed T into a unit disk graph T_G.

Rectilinear Embedding. Let T be a full rooted tree of maximum degree three and r be the root of T. Let m be the number of vertices in T. A new vertex s is added to the tree and connected to r. Let us call the new graph T' rooted at s with number of vertices, $N = m + 1$ vertices.

The leaves of the new graph T' are numbered according to their appearance in the pre-order traversal starting from s. Then, for each non-leaf vertex, number it with the median of the numbers corresponding to its children. If any of them has two children, then number it with the maximum number belonging to its children. After numbering all the nodes in this fashion, the vertex $\mathbf{v}$ gets the number $\mathbf{n(v)}$.

The embedding works in the following way. Put the vertex s at the coordinate $(2n(s), 0)$. Let v be any other vertex. Suppose the parent of v is u and u has been placed at (X_u, Y_u). The vertex v is placed at (X_v, Y_v), where $X_v = X_u + 2\,(n(u) - n(v))$, and $Y_v = Y_u + 2\,(N - |n(u) - n(v)|)$.

The edges are drawn as paths parallel to the axes. The edge between u and v is drawn as follows. Draw a line of length $2\,(N - |n(u) - n(v)|)$ in the positive Y direction from u and then take a 90 degree turn and draw a line of length $2|n(u) - n(v)|$ in the positive or negative direction of X axis, according to the positive or negative values of $n(u) - n(v)$ respectively. By this construction method the embedding is indeed a rectilinear embedding. Moreover, in this embedding

each edge is of length exactly $2N$ and has at most one **bend**. A 90 degree shift or rotation in the path has been termed as **bend** here.

The final step of our transformation is the following. On each edge starting from a parent u to a child v in T', $2N-1$ new vertices are placed equal distance apart in the path joining u to v. These new vertices are denoted by $W_v^1, \ldots, W_v^{2N-1}$, where W_v^1 is adjacent to u and W_v^{2N-1} is adjacent to v. Finally, for each vertex $v \in T'$, W_v^1 is split $2N-1$ times and $W_v^2, W_v^3, ..., W_v^{2N-1}, v$ are split $4N-1$ times. This graph is called T_G.

Lemma 4.1 *T_G is a unit disk graph.*

Proof. Consider any edge (w_1, w_2) in T_G. By construction, w_1, w_2 are two adjacent vertices in the path joining some vertices u and v of T. As all the paths are drawn parallel to one of the axis, $d_{\ell_2}(w_1, w_2) = 1$.
For the other direction, consider any non-adjacent pair of vertices w_1, w_2 of T_G. If the vertices are in a path joining two vertices of T in the embedding, then there is at least one vertex between w_1 and w_2 in the path. As the vertices are placed at distance 1, and each path has only one bend, $d_{\ell_2}(w_1, w_2) \geq \sqrt{2} > 1$.

Suppose, w_1 and w_2 are in different paths. By construction, distance between two completely different paths in T_G is at least 2. If some part of the paths are same, then $d_{\ell_2}(w_1, w_2) \geq \sqrt{2} > 1$. Hence, all the non adjacent vertices in T_G are more than unit distance away from each other. ■

For saving a vertex of the tree T, only one of its predecessors had to be vaccinated at some time instance. In another way, it can be said that at most, one firefighter is used at each time step towards saving a vertex in T. As the topology of the unit disk graph, T_G is kept similar to that of the tree T; the same rule applies here also.

The last step of the above construction produces $4N$ images of every vertex $v \in T$. Let Γ_G be the set of all $4n$ images for each of the vertices Γ. The constructed instance of the firefighter problem is (T_G, Γ_G, B). The following lemma proves the correctness of the reduction.

Lemma 4.2 *(T_G, Γ_G) is a YES instance for the Decision version of the Minimum Budget Firefighter Problem on the unit disk graph if and only if (T, Γ) is a yes*

instance of the Minimum Budget Firefighter problem on tree with maximum degree 3.

4.3.1.1 Proof of Lemma 4.2

The proof of Lemma 4.2 follows from the following two lemmas.

Lemma 4.3 *There is a strategy to save all $4N$ images of some vertex v of T_G in firefighter game if and only if there is a strategy to save all $2N$ images of W_v^1 of T_G.*

Proof. If in the graph T_G all the $4N$ images of some vertex v are saved, then it means at least one of the vertices $W_v^2, W_v^3, ..., W_v^{2N-1}, v$ had all its $4N$ images vaccinated till fire came to that level or all the $2N$ images of W_v^1 were vaccinated before the fire came to it. Now, if the first case is true for v or some W_v^k where $k \in \{2, 3, \ldots, 2N-1\}$ and at each time instance only one vertex can be vaccinated towards saving v, then at least $2N$ images of W_v^k were vaccinated when the fire reached to W_v^1. Then, in spite of vaccinating $2N$ images of W_v^k, all the images of W_v^1 could be vaccinated.
Also, in a contra-positive sense, if by any optimal strategy all the $4N$ images of v can't be saved before it reaches that level, then at least one vertex of W_v^1 must be infected when the fire reaches its level. So in T_G, saving v means saving W_v^1 and vice-versa.
To make this argument valid for W_t^1, where t is the child of r in T, the extra vertex s to T is added in the construction. ■

Lemma 4.4 *There is a strategy to save W_v^1 where W_u^1 can't be saved in T_G in firefighter game if and only if there is a strategy to save v where u can't be saved in T given v is the child of u in T.*

Proof. Now to correspond the unit disk graph problem to the case of a full rooted tree of maximum degree three problem, it is sufficient to prove that,

1. If it is possible to save vertex v while its parent u can't be saved in a full rooted tree, then it is possible to save W_v^1 when W_u^1 can't be saved in T_G.

2. In a contra-positive sense, if there is no strategy to save v while its parent u is infected in T, then there is no strategy in T_G to save W_v^1 when W_u^1 is infected and fire came ahead of it.

Now, for case 1, when u just gets infected and the fire is about to come towards v, v is vaccinated. A similar case in T_G will be when the fire reaches W_u^1 and affects some of its images, vaccination of the images of W_v^1 is started and it takes $2N$ time steps to reach W_v^1 by then all the images of it are vaccinated and save W_v^1.

For case 2, when u is infected, fire started to move towards v and by no strategy, v can be vaccinated. The similar case in T_G will be when W_u^1 is already infected, fire moved towards u and affected W_u^2, still it is not possible to start vaccinating the copies of W_v^1. Then by the time fire reaches W_v^1, all the images of it can't be saved as it will take $2N-1$ time steps and $2N$ vertices have to be vaccinated. So, W_v^1 can't be saved. ■

In **lemma 4.3**, the equivalence in terms of burning or saving between vertices v and W_v^1 of graph T_G has been shown and in **lemma 4.4**, the equivalence of saving strategies of vertex W_v^1 in graph T_G and vertex v in graph T has been shown. These two lemmas together prove that saving or burning of vertex v in graphs T_G and T are equivalent and thus prove **lemma 4.2**.

Lemma 4.1 and **Lemma 4.2** together proves **Theorem 4.1**.

4.4 Approximation Algorithm for MIN-BUDGET problem

Firefighter problems on interval graphs are polynomial-time solvable. Keeping this fact in mind, the underlying geometry of the unit disk graph in the Euclidean space can be modified a bit to get another geometry with a polynomial time solution.

Following is the polynomial time algorithm for the firefighter problem on interval graphs, which will be extended later for an approximation algorithm for the same problem on the unit disk graph.

4.4.1 Exact Algorithm for Interval Graph

Let $(G(V, E), v, T)$ be an instance of the MIN-BUDGET Firefighter Problem on the Interval graph, where v is the source vertex and $T \subset V$ is the set of vertices to be saved. The algorithm for Interval Graphs is described in **Algorithm 2**.

Algorithm 2 Algorithm for Firefighter Min-Budget Problem on Interval Graph

INPUT: $G(V, E), v, T|$ where G is an Interval Graph, $v \in V$ be the source vertex of fire and $T \subseteq V$ be the set of vertices to save.
$T_1 = T \cap N(v)$
$T' = T \setminus T_1$
$V_1 = V$
$A_1 = |N(v)|$
$B_1 = |T_1|$
for $i = 2 : n : 1$ **do**
  if $|T'| \neq 0$ **then**
    $V_i = V_{i-1} \setminus T_{i-1}$.
    $G_i(V_i, E_i)|\forall u, v \in V_i, (u, v) \in E \Rightarrow (u, v) \in E_i$
    $T_i = \{t \in T'|Dist_{G_i}(v, t) = i\}$
    $B_i = \left\lceil \frac{\sum_{k=1}^{i} |T_k|}{i} \right\rceil$
    $A_i = \left\lceil \frac{\sum_{k=1}^{i-1} |T_k|+|K_i|}{i} \right\rceil$
  end if
end for
if $\max_i B_i \leq \min_i A_i$ **then**
  The Budget is $\max_i B_i$ and firefighters are to be placed on T
else
  Let $\min_i A_i = A_m$, The Budget is $\max(A_m, \max_k B_k|k < m)$ and firefighters are to be placed on $\cup_{i=1}^{m-1} T_i \cup K_m$
end if

4.4.2 Description of Algorithm 2

Algorithm 2 deals with the problem in two ways:

1. At each step, it identifies the vertices of T which are vulnerable to fire using the variable B_i.

2. It also identifies the cut-set of the remaining vertices in T with the vertices on fire at each time instance using the variable A_i.

After these calculation it determines the optimum result. In the process it requires some other parameters namely V_i, G_i, T_i where,

- V_i is the set of vertices that are not vaccinated till the $(i-1)^{th}$ time instance.
- G_i is the induced graph of G on V_i.
- T_i is the set of vertices to save which are at distance i from the source vertex in G_i.

The correctness of the algorithm has been proven below.

4.4.3 Correctness of Algorithm 2

Lemma 4.5 *In the optimal strategy, Firefighters have to be placed on T_i, if not placed on all the vertices of K_i at the i^{th} time instance.*

Proof. Consider the i^{th} time instance. Suppose, there are some unprotected vertices of T_i. Moreover, suppose there are some vertices in K_i where no firefighter is placed. Then $\exists\ u_i \in T_i$, where fire reaches to the unprotected vertices of T_i at the i^{th} or the $(i+1)^{th}$ time instance, which is not permitted. So, firefighters have to be placed on the unprotected vertices of T_i at the i^{th} or the $(i+1)^{th}$ time instance. So, placing the firefighters on $K_i \setminus T_i$ at the i^{th} time instance will be waste of firefighters and will also increase the budget. So, it will be wise to place the firefighters either on K_i or on T_i at the i^{th} time instance. ■

Lemma 4.6 *If at the i^{th} time instance firefighters are placed on K_i then fire will never reach to T_k, $\forall\ k \geq i$.*

Proof. Suppose, at the i^{th} time instance firefighters are placed on all vertices of K_i, then fire can never reach to any of the vertices of $W_i : W_i = T_i \setminus K_i$. The span of $W_i \geq 1$ as the intervals are of unit length. So, fire will never reach to T_k for all $k \geq i$. ■

If at some level i, the better strategy is to put firefighters on K_i then the fire fighting process will stop at that time instance. So, either firefighters have to be placed on K_i at the i^{th} time instance and on T_k's at corresponding time instance for $k < i$ or firefighters have to be placed on T_i's $\forall i$. Now, if the optimum strategy is to place fire fighters on K_i at the i^{th} step then the budget must be greater than both A_m and $\max_k B_k | k < m$, where $\min_i A_i = A_m$ as both are the necessary conditions. But, if $\max_i B_i \leq \min_i A_i$, then $\max_i B_i$ gives the optimum budget as a necessary condition. This proves the correctness of the algorithm.

4.5 Approximation Algorithm for Unit Disk Graph

4.5.1 Description of Algorithm 3

The algorithm for minimum budget firefighter problem on unit disk graph is based on the algorithm for minimum budget firefighter problem on unit interval graph described in Algorithm 2. The problem on unit disk graph has been considered as a combination of four simultaneous interval graph problems in four different directions, i.e., left(l), right(r), up(u), down(d). For a given graph $G(V, E)$ and a fire break-out point $v \in V$, the minimum budget of firefighters to be placed at each time instance has to be found to save a given set of vertices $T \subseteq V$.

As a first step, all the unit disks corresponding to the vertices of G are replaced by their circumscribing squares with sides parallel to the rectilinear axes of the plane. Then the vertices of G are sorted in two different arrays, one according to the X-coordinates of the centers of their corresponding squares and the other according to the Y-coordinates of the centers of their corresponding squares. These two sorted arrays are for later use in different algorithm stages.

Afterwards, squares corresponding to the disk v i.e., (S^v) and its boundaries (S^v_m), boundaries of the burning rectangle (R^i_m) at the i^{th} iteration and direction m, Set of vertices saved (T^i_m) at the i^{th} iteration and direction m, vertices saved in the i^{th} iteration (T^i) and in each direction (T^i_m), vertices at the boundary of the burning rectangle at the i^{th} iteration in each direction (C^i_m), Budget required at each direction $(\mathfrak{B})_m$ and set of vertices on which firefighters to be placed ($\mathcal{V}_m$)

Algorithm 3 Firefighter Min-Budget Problem on Unit Disk Graph

INPUT: $G(V,E), v, T|$ where G is an Unit Disk Graph, $v \in V$ be the source vertex of fire and $T \subseteq V$ be the set of vertices to save.
Replace all the disks by their circumscribing squares with sides parallel to coordinate axes and $G(V,E)$ becomes an Unit Square Graph.
$\forall v_i \in V$ sort them according to the X_l^i values of their corresponding Square S_i in increasing order. If $X_l^i = X_l^j$ then sort them according to their increasing order of Y_d^i and Y_d^j.
$\forall v_i \in V$ sort them according to the Y_d^i values of their corresponding Square S_i in increasing order. If $Y_d^i = Y_d^j$ then sort them according to their increasing order of X_l^i and X_l^j.
The boundaries of the burning rectangle (R) be, $R_l^0 = X_l^v, R_r^0 = X_r^v, R_u^0 = Y_u^v, R_d^0 = Y_d^v$.
$T^0 = \phi$.
$T' = T \setminus T^0$.
$V_0 = V$.
$C_m^0 = \{v\} \forall m \in \{l,r,u,d\}$.
for $i = 1 : n : 1$ **do**
  if $T' \neq \phi$ **then**
    $V_r^i = W \mid \forall w \in W \subset V, R_r^{i-1} - 1 < S_l^w < R_r^{i-1}$ & $R_d^{i-1} - 1 < S_d^w < R_u^{i-1}$ & $\exists v \in C_r^{i-1} | (S_l^w - S_l^v)^2 + (S_d^w - S_d^v)^2 \leq 1$.
    $V_l^i = W \mid \forall w \in W \subset V, R_l^{i-1} + 1 > S_r^w > R_l^{i-1}$ & $R_d^{i-1} - 1 < S_d^w < R_u^{i-1}$ & $\exists v \in C_l^{i-1} | (S_r^w - S_r^v)^2 + (S_d^w - S_d^v)^2 \leq 1$.
    $V_u^i = W \mid \forall w \in W \subset V, R_u^{i-1} > S_d^w > R_u^{i-1} - 1$ & $R_l^{i-1} - 1 < S_l^w < R_r^{i-1}$ & $\exists v \in C_u^{i-1} | (S_l^w - S_l^v)^2 + (S_d^w - S_d^v)^2 \leq 1$.
    $V_d^i = W \mid \forall w \in W \subset V, R_d^{i-1} < S_u^w < R_d^{i-1} + 1$ & $R_l^{i-1} - 1 < S_l^w < R_r^{i-1}$ & $\exists v \in C_d^{i-1} | (S_l^w - S_l^v)^2 + (S_u^w - S_u^v)^2 \leq 1$.
    for $m \in \{l,r,u,d\}$ **do**
      $T_m^i = T \cap V_m^i$.
      if $V_m^i \setminus T_m^i \neq \phi$ **then**
        $C_m^i = V_m^i \setminus T_m^i$
      else
        $C_m^i = C_m^{i-1}$
      end if
      $R_m^i = X_m^v || X_m^v | = \max_w |X_m^w| \forall w \in C_m^i$
    end for
    $T^i = T_r^i \cup T_l^i \cup T_u^i \cup T_d^i$.
    $T' = T' \setminus T^i$.
    $V_i = V_{i-1} \setminus T^i$.
    $G_i(V_i, E_i) | \forall u, v \in V_i, (u,v) \in E \Rightarrow (u,v) \in E_i$.
    K_m^i = Minimum Cut-set of V_m^i in G_i where, $m \in \{r,l,u,d\}$.
    $B_m^i = \left\lceil \frac{\sum_{k=1}^{i} |T_m^k|}{i} \right\rceil$ where, $m \in \{r,l,u,d\}$.
    $A_m^i = \left\lceil \frac{\sum_{k=1}^{i-1} |T_m^k| + |K_m^i|}{i} \right\rceil$ where, $m \in \{r,l,u,d\}$.
  end if
end for
$\mathfrak{B} = 0, \mathcal{V} = \phi$.
for each $m \in \{r,l,u,d\}$ **do**
  $T_m = \cup_{i=1}^{n} T_m^i$.
  $\mathfrak{B}_m = 0, \mathcal{V}_m = \phi$.
  if $\max_i B_m^i \leq \min_i A_m^i$ **then**
    Then Budget $\mathfrak{B}_m$ is $\max_i B_m^i$ and firefighters are to be placed on $\mathcal{V}_m = T_m$
  else
    Let $\min_i A_m^i = A_m^t$, Then Budget $\mathfrak{B}_m$ is $\max(A_m^t, \max_k B_m^k | k < t)$ and firefighters are to be placed on $\mathcal{V}_m = \cup_{i=1}^{t-1} T_m^i \cup K_m^t$.
  end if
  $\mathfrak{B} = \mathfrak{B} + \mathfrak{B}_m$.
  $\mathcal{V} = \mathcal{V} \cup \mathcal{V}_m$.
end for

are defined.

Now at each iteration, the new set of vertices whose corresponding squares have an intersection with the burning rectangle is identified and also filtered depending on whether their connected disks have an intersection with any of the disks corresponding to the set of vertices constituting the boundary of the burning rectangle at the previous iteration. Then the burning rectangle and its boundary, vertices to be saved at the current iteration, the set of vertices constituting the new boundary of the burning rectangle and the minimum cut set of the vertices that completely hold fire at the current iteration is updated. Depending on the cut set and the saved vertices at each iteration, the budget for each direction, along with the final budget and the set of vertices to be placed firefighter on, are calculated.

Lemma 4.7 ***Algorithm 3*** *produces an approximation solution for the Firefighter Min- Budget Problem on Unit Disk Graph with approximation factor 2.*

Proof.

Let there be n vertices in the unit disk graph $G(V, E)$, where v be the source vertex of fire and $T \subset V$ be the set of vertices to be saved. First, the disks are replaced with circumscribing squares with sides parallel to the coordinate axes. This problem can be seen as four firefighter MIN-BUDGET problems on Unit interval graphs by observing that fire can spread in the up, down, left or right direction accordingly starting from v. So, our algorithm for the MIN-BUDGET problem on **Interval Graph** can be used simultaneously in the four directions around v. Due to this modification, the shape of the region of influence of fire at any time becomes a rectangle called the **Burning Rectangle**. The algorithm accounts for the boundary vertices of this burning rectangle at each time instance. Due to this geometric assumption, two kinds of errors are introduced, namely: **Topological Error** and **Time Dependent Error**.

The **Topological Error** can be observed in terms of getting a denser graph and, as a result, vaccinating more vertices than required in the actual scenario. But this error can be eliminated by checking at each time instance whether a vertex to save in consideration is indeed within the unit distance of any of the boundary vertices of the burning rectangle.

But the **Time Dependent Error** can not be eliminated. Due to the burning rectangle assumption, a vertex corresponding to a unit disk centered at (X, Y) can burn as early as at $\text{Max}(X, Y)$ time instant, while in actual case, it can take at most $2(X + Y)$ time. As the fire has to travel at most $2k$ squares to reach $(k, 0)$ point starting from $(0, 0)$. The limiting case arises when every square has a neighbor square slightly shifted in the X direction and the Y direction. Due to this error, the partial average of the budget can be as big as $\frac{2(X+Y)}{\max(X,Y)}$ times the optimum. As $\frac{2(X+Y)}{\max(X,Y)} \leq 2$, this gives an approximation factor of 2.

■

4.5.2 Correctness of the Algorithm

In the algorithm for Minimum Budget Firefighter Problem on Unit Disk Graph, the algorithm for Minimum Budget Firefighter Problem on Interval Graph has been applied simultaneously in four directions, namely, upwards, downwards, left and right, using the subscript m. So the correctness of the algorithm in each of these four directions is guaranteed by the correctness of the algorithm for the interval graph that is already proved in subsection 4.4.3.

It should also be noted that due to the geometric assumption of the burning rectangle, an approximation factor of 2 has been introduced by this algorithm as described in **Lemma 4.7**.

4.6 Concluding Remarks

- In this chapter, the Firefighter Min-Budget Problem on Unit Disk Graphs has been proven to be NP-Hard.
- Also, an approximation algorithm of Firefighter Min-Budget Problem on Unit Disk Graphs with an approximation factor of 2 has been presented in this chapter.

CHAPTER 5

An Approximation Algorithm of Firebreak Problem on Split Graphs

5.1 Introduction

In many real-life communication, distribution and social networks, it is seen that a set of nodes or elements are very strongly connected and the other nodes are only connected to this strongly connected part of the network. This kind of network structure resembles split graphs where the set of vertices can be split into two sets, namely a clique or complete graph and an independent set.

When a malicious program or contagion starts spreading through a network, then to contain it or to save some vertices of the network firefighter problem is introduced. In a firefighter problem, the malicious program or contagion is considered as fire and the goal is to save vertices from the fire. The firefighter problem has two main variants, namely **max-save** and **min-budget**. In the Max-save variant, the goal is to save maximum number of nodes by placing firefighters. In the Min-budget variant, the goal is to save a given set of vertices by placing the minimum number of firefighters on the nodes at each time instance. But in

both variants, the common part is that the firefighters can be placed on different vertices over time, making it a game between the firefighters and the fire.

Now, if the scenario changes in such a way that the firefighters must be placed at once and the certain number of vertices of the graph to be saved, then the firefighter problem becomes a firebreak problem. Like the max-save version of the firefighter problem, maximizing the number of saved vertices is unnecessary. Also, unlike the min-budget version of the firefighter problem, only the desired number of vertices has to be saved instead of any specific set of vertices. This difference from the firefighter problem makes it interesting. In a firebreak problem, we need to place firefighters on some nodes of the network at once to save a given number of nodes of the network. In another way, this problem is similar to the vertex cut problem. But the difference is no specific set of saved vertices is given here. So the firebreak problem on different graph classes has different hardness levels. Due to its similarity with the current pandemic situation, the firebreak problem on split graphs is a very relevant problem today.

A use case of the firebreak problem on split graphs can be seen in the privacy policy on sharing content on Facebook with friends of friends. Suppose a Facebook user wants to share a content with friends of friends. Then the content will be seen by all the users within 2 distance from the source user of the content and can not be seen outside this set of users. This can be seen as a firebreak problem where the content can not reach the next level of users.

The firebreak problem on trees and intersection graphs is solvable in polynomial time, but in the case of split graphs, the problem has been proven to be NP-hard by Barnetson et al.[47]. This chapter presents an approximation algorithm with an approximation factor of 2.

5.1.1 Previous Work

The firebreak problem is a special version of the firefighter problem and was introduced by Barnetson, Burgess, Enright, Howell, Pike and Ryan [47]. The firefighter problem was first introduced by Hartnell [30]. Harnell also showed a simple 2-approximation algorithm for Firefighter Max-Save Problem[31]. The firefighter problem has a rich history of work. Some results relevant to this chapter are

mentioned here. For a detailed account, readers are referred to the survey on results, directions and open questions related to firefighter problems by Finbow and MacGillivray [29]. MacGillivray and Wang proved NP-hardness of the firefighter problem on Bipartite graphs [34]. Fomin, Heggernes and Van Leeuwen considered the max-save firefighter problem and gave a polynomial time algorithm for interval graphs, split graphs and permutation graphs and also proved NP-hardness of the problem on Unit Disk Graphs in [41, 42]. Borgatti first introduced a dissemination problem related to the Firebreak Problem, where the goal is to select a set S of k vertices of a graph G from where reaching $G \setminus S$ is easy, along with a similar separation problem in 2002 [48]. Borgatti also connected these two problems with the key player problem in [49]. Applicability of the firebreak problem to the robustness of networks can be found in [50, 51, 52]. k-way vertex cut problem has lots of similarities with the firebreak problem. Computational complexity of k-way vertex cut problem on different types of graphs can be found in [53, 54, 55, 56]. A survey by Lalou, Tahraoui and Kheddouci gives an insight into detecting critical nodes of a network[57]. Works on edge-based version of the firebreak problem can be found in [58, 59, 60].

5.1.2 Some Definitions

Definition 5.1 (*Firebreak Problem*) *For a graph $G(V, E)$ and a source of fire s where $s \in V$, the firebreak problem asks for the minimum size of the subset S of $V \setminus \{s\}$ such that, placing firefighters on the vertices of S at least t number of vertices of $V \setminus (S \cup \{s\})$ can be saved from fire.*

Definition 5.2 (*Split Graph*) *A graph $G(V, E)$ is called a split graph if its' set of vertices V can be split into two disjoint subsets A and B where A induces a clique in G and B induces an independent set in G.*

Definition 5.3 (*t-Subset Cover*) *Let N=$\{1, 2, \ldots, n-1, n\}$ and $A, B, \cdots \subset N$. The **t-Subset Cover** of N is the minimum sized subset T of N where at least t among the given subsets of N are subsets of T.*

5.1.3 Problem Definition

Definition 5.4 (*Firebreak Problem on Split Graph*) *For a given split graph $G(V, E)$, a natural number t and a source vertex of fire $s \in V$, the firebreak problem on G asks for the minimum size of the set $T \subset (V \setminus \{s\})$ such that in the induced graph of G on $V \setminus T$ at least t vertices are disconnected with s.*

5.1.4 Our Result

Theorem 5.1 *There is an approximation algorithm for the Firebreak Problem on a split graph with an approximation factor of 2.*

5.1.5 Correspondence of Firebreak Problem on Split Graphs with t-Subset Cover

In this section, the correspondence between the Firebreak Problem on Split Graphs and Subset Cover Problem has been established by proving the following theorem.

Let a given split graph $G(X \cup Y, E)$ has partitions of vertices X and Y where X is the set of vertices of the clique and Y is the set of vertices of the independent set. Suppose a solution to the firebreak problem on G to save t vertices is desired. The vertices of X are denoted with numbers $1, 2, \ldots$ and the vertices of Y are denoted with alphabets $a, b, \ldots$. Let $|X| = n$ and $|Y| \geq t$ as otherwise, the firebreak problem has no solution. There are no edges between the vertices in Y as, by definition, they induce an independent set in G. Now each of the vertices of Y i.e., $a, b, \ldots$ correspond to subsets $\gamma_a, \gamma_b, \ldots$ of X where $\gamma_y = N(y)$ for all $y \in Y$ where $N(y)$ is the set of neighbors of y in G.

Theorem 5.2 *Firebreak Problem on Split Graph G to save t vertices is equivalent to t-Subset Cover Problem where the vertices of X are elements and γ_y for all $y \in Y$ are the subsets.*

Proof. The following lemmas are used to prove this theorem.

Lemma 5.1 *All the firefighters has to be placed on the vertices in X.*

Proof. If fire breaks at a vertex s in X, then there is no point in placing firefighters on the vertices of Y as saving them will not save any further vertices of the graph. So all the firefighters must be placed on vertices of X. If fire breaks at a vertex $y \in Y$, then either firefighters have to be placed on all the elements of γ_y or the fire will reach some of the vertices of X as $\gamma_y \subseteq X$ and the scenario will merge with the first case. Now, if firefighters are placed on all the elements of γ_y, then the only remaining job will be to check whether $n - |\gamma_y| - 1 \geq t$ as in this case, fire can not spread further. If $n - |\gamma_y| - 1 < t$ then firebreak problem has no solution. But, if $n - |\gamma_y| - 1 \geq t$ then it is to be checked that whether there are t vertices $y_1, y_2, \ldots, y_t \in Y \setminus \{y\}$ such that,

$$\bigcup_{i=1}^{t} \gamma_{y_i} < \gamma_y$$

which is also a Firebreak Problem on an induced sub-graph of G with vertex set $X \cup Y \setminus \{y\}$. So in all the cases, firefighters are placed on the vertices in X. This proves the lemma. ■

Lemma 5.2 *All the saved vertices can be assumed to belong to the set Y.*

Proof. If a fire starts from a vertex in X, then the fire will reach all the other vertices of X unless placed firefighters on, as the vertices of X induces a clique in G. So they can not come in the set of saved vertices. Now, if the fire starts from a vertex $y \in Y$, then either firefighters will be placed on all the elements of γ_y and fire will be contained or some of the vertices in X will be open to fire. In the first case, some vertices in X may be saved but to verify the correctness of the result, the firebreak problem on an induced sub-graph of G by the vertex set $X \cup Y \setminus \{y\}$ has to be examined, which makes the saving of vertices of X of no interest. In the latter case, it is the same as the source of fire in X with delayed by an instance of time. So in all the cases, the vertices of Y are considered for inclusion in the set of saved vertices. This proves the lemma. ■

Lemma 5.3 *Firefighters are placed on the smallest set of vertices in X that covers at least t number of subsets of X corresponding to the vertices of Y.*

Proof. Here the goal should be finding the minimum number of vertices in X for placing the firefighters on to save at least t number of vertices in Y. To save a vertex $y \in Y$, the firefighters must be placed on all the elements of γ_y. Now to save t such vertices in Y, it is needed to find a set of vertices $y_1, y_2, \ldots, y_t \in Y$ such that $\bigcup_{i=1}^{t} \gamma_{y_i}$ is minimum. So saving t vertices in a firebreak problem in a split graph G becomes the same as covering t subsets of X. This proves the lemma. ■

These three lemmas together prove the theorem 5.2. ■

5.1.6 Some Measures for t-Subset Cover Problem

Let d_y be the degree of a vertex $y \in Y$. So $|N_y| = |\gamma_y| = d_y$. We define $f(y) = \frac{1}{d_y}$ for all $y \in Y$. It can be observed that by including a neighbor of a vertex $y \in Y$ in T we are actually covering $\frac{1}{d_y}$ part of the set y. Now for each vertex $x \in X$ we define,

$$F(x) = \sum_{y \in (Y \cap N(x))} f(y). \tag{5.1}$$

$$C(y) = \sum_{x \in N(y)} F(x). \tag{5.2}$$

Definition 5.5 ***The density of a vertex*** $y \in Y$ *is defined as*

$$D_y = \frac{|N(y)|}{C(y)}. \tag{5.3}$$

5.2 Our Technique

Our technique involves a greedy algorithm based on density, as defined in the previous section. We mainly follow a ***Density Based*** algorithm and in the end do a little modification to get an approximation factor for our algorithm.

First, the set of neighbors for all the vertices in X and Y are formed. Let T_{den} be our resulting subset of X that covers at least t vertices of Y. So we initialize $T_{den} = \phi$. All the neighbors of $y_1 \in Y$ are included in T_{den} where y_1 has the minimum density among all the vertices in Y. If two or more vertices in Y

have minimum density, then we break the tie using the partial coverage of other vertices in Y by their neighbors and select the one which covers the greater part of another vertex in Y. Then we modify the set of neighbors for all vertices that are not selected in both the sides X and Y and recalculate the densities of the remaining vertices of Y.

We repeatedly apply the steps mentioned above on the modified settings till a position where including neighbors of any remaining $y \in Y$ makes the number of covered vertices in Y more than t. If the inclusion makes the number of covered vertices exactly t, then we give that T_{den} as our output. Otherwise, we go to the next step where we look for the vertex in Y that is not covered by T_{den} till now and has the minimum degree in the modified scenario. Then we include all its neighbors in X that are not included in T_{den} and give the $|T_{den}|$ as our result.

When the density-based part of the algorithm stops after covering $(t - k)$ vertices of Y by the elements of T_{den} where $t \geq k \geq 1$, it can be proved that the size of T_{den} at that step is optimum if the firebreak problem is posed for $(t-k)$ in spite of t. Intuitively it may seem that a sequence of selected subsets may exist other than that selected by our density-based algorithm, which keeps the cumulative density of the selected subsets even smaller. Also, it can be proved that in that case, one subset among the selected subsets will have an even smaller density than their cumulative density. The cumulative density of these subsets is lesser than that of the selected subsets using our density-based algorithm. Now, one of the selected subsets in our algorithm will have more density than the cumulative density of the subsets in our density-based algorithm. So there will be a contradiction that why this subset with higher density was chosen over an existing less dense subset. This shows that there can not be a situation like this. A detailed proof of this is given in the proof section of the chapter.

5.2.1 Our Algorithm

The algorithm for t-Subset Cover Problem is presented in Algorithm-4.

Algorithm 4 Algorithm for t-Subset Cover Problem

Input: Split graph $G(V, E)$.
Determine the set of vertices $X \subset V$ that induces ***clique*** in G.
Determine the set of vertices $Y \subset V$ that induces ***independent set*** in G.
Determine $f(y)$ for all $y \in Y$.
Calculate $F(x)$ for all $x \in X$.
Calculate D_y for all $y \in Y$. {Calculation of initial densities }
Initialize $T_{den} = \phi$. {Initialization of resulting set}
Initialize $Z = \phi$. {Initialization of dummy set resulting subsets}
for $|Z| < t$ **do**
  Find $y \in Y$ such that D_y is minimum among all D_i's for $i \in Y$.
  Modify $Z = Z \cup y$. {Inclusion of minimum density subset}
  Modify $Y = Y \setminus y$. {Modification of set of subsets}
  Modify $X = X \setminus N(y)$.{Modification of set of elements}
  Modify $F(x)$ for all $x \in X$.
  Modify D_y for all $y \in Y$. {Modification of densities}
  if $D_y = 0$ for some $y \in Y$ **then**
    Modify $Z = Z \cup y$. {Considering the automatically included subsets}
    Modify $Y = Y \setminus y$.
  end if
  if $|Z| \leq t$ **then**
    $T_{den} = T_{den} \cup N(y)$. {Including the elements of covered subset}
  end if
end for {Condition checking for repeating the steps}
if $|Z| > t$ **then**
  Find $y \in Y$ such that $N(y) \setminus T_{den}$ is minimum.
  $T_{den} = T_{den} \cup N(y)$. {Inclusion of elements in the final step}
end if
Output: $|T_{den}|$.

5.3 Proof of Results

In this section, all the proofs supporting the methods, techniques and results of this chapter have been discussed. First, some conditions for the firebreak problem on split graphs have been proved. Then the correspondence of these conditions with the t-subset cover problem is shown. As the algorithm described in this chapter gives an approximate solution to the t-subset cover problem, the application of the algorithm in this case and its correctness are shown. At last, the correctness of the approximation factor has been proved.

Theorem 5.3 *It is sufficient to assume that the source vertex of the firebreak problem on a split graph belongs to the clique and we need to save some of the vertices of the independent set by placing firefighters on some of the vertices of the clique.*

Proof. To prove the theorem, the following lemmas are used.

Lemma 5.4 *For the firebreak problem on split graphs, firefighters need not be placed on the vertices in the independent set of the split graph.*

Proof. In a split graph, any vertex in the independent set has edges only with some of the vertices of the clique. Now in a firebreak problem, we are trying to save its neighbors in the clique by placing firefighters on this vertex. But if this vertex is the source of fire, then we can not place a firefighter on it by definition. Also if the source of fire is any other vertex in the clique or the independent set, then the fire reaches the vertices that we are trying to save first and then to the vertex that we are placing firefighter on. So there is no point in placing a firefighter on the vertex. This proves the lemma. ■

Lemma 5.5 *In a firebreak problem on split graphs, it is enough to consider only the vertices in the independent set for inclusion in the set of vertices to be saved and a vertex in the clique as the source of the fire.*

Proof. In a firebreak problem, the set of saved vertices consist of those vertices where the fire can not reach due to placing firefighters. Now, if the source vertex belongs to the set of the clique of the split graph, then each of the other vertices in

the clique is directly connected to the source. So these vertices will either be brunt or will be placed firefighters upon to save some of the vertices of the independent set; thus can not be included in the set of saved vertices. If the source vertex belongs to the independent set, placing firefighters on all its neighbors, all other vertices of the graph can be saved but that may not be the minimum number of vertices to place firefighters on to save the given minimum number of vertices. So we need to check whether the set of neighbors of the source is indeed the minimum sized set of vertices to place fighters on. For that, if we do not place firefighters on all its neighbors, then one of the vertices in the clique gets fire and the case turns out to be the same with a source of fire in the clique. Thus again, with the same logic, none of the vertices in the clique can be put in the set of the saved vertices and the source of the fire can always be considered as one of the vertices of the clique. These two cases, depending on the location of the source of fire, cover all the possibilities and proves the lemma. ■

These two lemmas together prove the theorem 5.3. ■

From the theorem above, it can be observed that some of the vertices of the clique will be burnt and on others, firefighters will be placed to save some of the vertices of the independent set as the vertices of the clique are connected with each other directly. So the problem comes down to finding a t sized subset of the independent set where the size of the union of the neighbors of its elements is minimum. The problem can be seen from another angle. If all the edges of the clique are removed from the graph, it will become a bipartite graph with the vertices of the clique and the independent set of the main graph as the two partitions. Suppose each vertex of the independent set of the main graph is considered as sets and their neighbors in the bipartite graph as their elements. In that case, the problem turns into ***t-Subset Cover*** problem on the set of vertices in the clique of the main graph, with the sets corresponding to the vertices in the independent set of the main graph being the subsets.

The algorithm described in the chapter is used to get an approximate result of this ***t-Subset Cover*** problem. So an approximate solution to the ***Firebreak Problem on Split Graphs*** can also be found from this. Now the only part remaining is to prove the correctness of the algorithm and the approximation ratio.

According to the technique described in the previous section, suppose the density-based greedy algorithm stops when the number of subsets covered is $(t-k)$ where $k \in \{0, 1, 2, \ldots, t\}$. In this scenario, it can be shown that this result is optimum for the ***(t-k)-Subset Cover Problem***.

Theorem 5.4 *If the density based greedy algorithm part of the approximation algorithm for* ***t-Subset Cover Problem*** *halts after covering* $(t-k)$ *subsets then that is the optimal result for* ***(t-k)-Subset Cover Problem*** *on the same graph.*

Proof. The theorem will be proved by contradiction. Suppose the density-based greedy part of the algorithm halts after covering $(t-k)$ subsets. Let $t-k = m$ and these subsets are covered by p elements. Suppose the optimum result for covering be n (where $n \leq p$) and in the optimum case the covered subsets are $\{y_1, y_2, \ldots, y_m\}$. Let the elements that cover these m subsets are given as $\{x_1, x_2, \ldots, x_n\}$. Now the following lemmas will help to prove the theorem.

Lemma 5.6 *There is a subset of vertices* y_i *for* $i \in \{1, 2, \ldots, m\}$ *such that its density is less or equal to* $\frac{n}{m}$.

Proof. Let the size of each subset y_i be d_i for $i \in \{1, 2, \ldots, m\}$ and each x_j occurs in D_j number of y_i's for $j \in \{1, 2, \ldots, n\}$. This y_i's are vertices of G and subsets in t-Subset Cover Problem. So the density of these subsets can be calculated using the definition of density of a vertex in section 5.1.6. Density of each subset y_i will be,

$$\text{Density of } y_i = \frac{d_i}{C(y_i)}.$$

Lemma 5.7 $\sum_{i=1}^{m} d_i = \sum_{j=1}^{n} D_j$.

Proof. The m number of subsets y_i for $i \in \{1, 2, \ldots, m\}$ are completely covered by the n elements x_j for $j = \{1, 2, \ldots, n\}$. Also these n elements occur in these m subsets exactly D_j times each respectively. Now if the sum over all d_i's of the m subsets is calculated, it gives exactly the total number of occurrences of the n elements in these m subsets which is the sum over all D_j. This proves the lemma. ■

Lemma 5.8 *For two positive real sequences* $\{a\}_n$ *and* $\{b\}_n$ *if* $b_i \geq b_j \implies \frac{a_i}{b_i} \leq \frac{a_j}{b_j}$ *for all* $i, j \in \{1, 2, \ldots, n\}$ *then* $\frac{\sum_{i=1}^{n} \frac{a_i}{b_i}}{n} \geq \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$.

Proof. Let there be b_i number of positive real elements of value $\frac{a_i}{b_i}$ for all $i \in \{1, 2, \ldots, n\}$ and without loss of generality it can be assumed that,

$$\frac{a_1}{b_1} \leq \frac{a_2}{b_2} \leq \frac{a_3}{b_3} \leq \cdots \leq \frac{a_{n-1}}{b_{n-1}} \leq \frac{a_n}{b_n}.$$

So it is evident that in this set of values, the lower values are high in number and higher values are low in number. Also, the numerator of the RHS of the inequality is the sum of all the elements and the denominator is the total number of elements as, sum of the elements

$$= \sum_{i=1}^{n} \frac{a_i \times b_i}{b_i} = \sum_{i=1}^{n} a_i.$$

So the RHS is the arithmetic mean or average of all the elements. Also the LHS of the inequality is the arithmetic mean of some of the elements of the set leaving some of the smaller elements i.e., average of nb_n of elements where each $\frac{a_i}{b_i}$ has been taken into account b_n times as, Average of these elements

$$= \frac{b_n \sum_{i=1}^{n} \frac{a_i}{b_i}}{nb_n} = \frac{\sum_{i=1}^{n} \frac{a_i}{b_i}}{n}.$$

So as some of the smaller elements are not considered in the LHS for averaging, the LHS will be bigger than the RHS This proves the lemma. ■

Lemma 5.9 *For two positive real sequences* $\{a\}_n$ *and* $\{b\}_n$ *there exists an* i *for which* $\frac{a_i}{b_i} \leq \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$.

Proof. In the proof of the previous lemma, it is shown that if there are b_i number of positive real elements of value $\frac{a_i}{b_i}$ for all $i \in \{1, 2, \ldots, n\}$ then RHS of the inequality to prove becomes the arithmetic mean or the average of all these elements. So if all the elements are not equal to this average, there is an element

that is less than this average. Otherwise, they are all equal to the average itself. This proves the lemma. ■

Lemma 5.10 $\sum_{i=1}^{m} C(y_i) \geq \frac{m}{n} \sum_{j=1}^{n} D_j$.

Proof. By definition,

$$\sum_{i=1}^{m} C(y_i) = \sum_{i=1}^{m} \sum_{x_j \in y_i} F(x_j) = \sum_{i=1}^{m} \sum_{x_j \in y_i} \sum_{y_p \in N(x_j)} f(y_p).$$

It can be observed that due to the double integration each $f(y_i)$ is summed $\sum_{x_j \in y_i} D_j$ times for $i \in \{1, 2, \ldots, m\}$. So,

$$\sum_{i=1}^{m} C(y_i) = \sum_{i=1}^{m} f(y_i) \sum_{x_j \in y_i} D_j = \sum_{i=1}^{m} \frac{\sum_{x_j \in y_i} Dj}{d_i}.$$

Now it is assumed that,

$$d_i \geq d_p \implies \frac{\sum_{x_j \in y_i} Dj}{d_i} \leq \frac{\sum_{x_j \in y_p} Dj}{d_p}$$

for all $i, p \in \{1, 2, \ldots, m\}$. The validity and supporting statements are given in the next lemma.

So

$$\sum_{i=1}^{m} C(y_i) \geq \frac{m \sum_{j=1}^{m} D_j^2}{\sum_{j=1}^{m} D_j} \geq \frac{m \sum_{j=1}^{m} D_j}{n} = \frac{m}{n} \sum_{j=1}^{n} D_j.$$

■

Lemma 5.11 $d_i \geq d_p \implies \frac{\sum_{x_j \in y_i} Dj}{d_i} \leq \frac{\sum_{x_j \in y_p} Dj}{d_p}$ *for all* $i, p \in \{1, 2, \ldots, m\}$.

Proof. Suppose in the optimum solution of the $(t-k)$-***subset cover*** problem two subsets y_i and y_p are selected where $d_i \geq d_p$ and

$$\frac{\sum_{x_j \in y_i} Dj}{d_i} \geq \frac{\sum_{x_j \in y_p} Dj}{d_p}.$$

This implies that the subset with more elements has a higher average degree of its elements. But with this higher average degree, its elements cover fewer subsets, which implies that the subsets they are part of are comparatively bigger and tend to make the optimum result bigger. So this contradicts the result being optimum and proves the lemma. ■

So

$$\frac{\sum_{i=1}^{m} d_i}{\sum_{i=1}^{m} C(y_i)} \leq \frac{n}{m}$$

which implies that there exists an $i \in \{1, 2, \ldots, m\}$ such that $\frac{d_i}{C(y_i)} \leq \frac{n}{m}$ and also proves the lemma. ■

Now, the algorithm described in this chapter starts by including the subset with a lower density. So it is evident that in the latter part of the algorithm, the included subsets have a higher density than its final average density, i.e., $\frac{p}{m} > \frac{n}{m}$. Now, if there is a subset in the optimum result with a density less than $\frac{n}{m}$, then it could have been added in the later part of our algorithm if not already included earlier. It can be concluded that all the subsets with a density less than $\frac{p}{m}$ are already included in the algorithm described in this chapter. So if all these common subsets are excluded from both sides, then the average density of the remaining subsets in the optimum result must be higher than that of the proposed algorithm, which contradicts the optimity of the optimum result. This proves our theorem. ■

As the density based part of the proposed algorithm gives the optimum result for $(t-k)$-***Subset Cover Problem*** for some $k \in \{0, 1, 2, \ldots, t\}$, this result will be less than the optimum result for t-***Subset Cover Problem***. Otherwise, the optimity of the results will be contradicted. Now, if $k = 0$, the optimum result is obtained by the density-based part of the proposed algorithm. If $k \neq 0$, then according to the proposed algorithm, those new elements are included in T_{den} that increases its size the least.

Lemma 5.12 *The number of the new elements added to T_{den} in the last step of the proposed algorithm is less than the optimum result of the t-**Subset Cover Problem**.*

Proof. Let the number of new elements added in the last step of the proposed algorithm be l. Now, if it is greater than the optimum result of ***t-Subset Cover Problem*** then there must be a subset with a size less than l whose all the elements are not included in the T_{den}. Otherwise, the density-based part of the proposed algorithm would have given the optimum result for ***t-Subset Cover Problem***. In that case, this subset will have fewer new elements that can be added to get the optimum result for the problem. But this contradicts our assumption of l being the smallest number of elements. So by contradiction, it can be proved that l is less than the optimum result of the ***t-Subset Cover Problem***. ■

Now both part of the proposed algorithm, i.e., the density-based part and the inclusion of new elements at the last step, includes less number of elements than the optimum result of the ***t-Subset Cover Problem***. So the proposed algorithm will provide a result within 2 times the optimum result of the ***t-Subset Cover Problem***. This proves Theorem 5.1.

5.4 Concluding Remarks

- The firebreak problem in a network is similar to the current pandemic situation in the world and also fits in a case of the spread of malicious program through a communication network. To contain the spread, it is necessary to know that on which nodes precautions have to be taken.

- When the network is a split graph, then an optimal solution in polynomial time is not possible. But this chapter presents an approximation algorithm that restricts the number of vertices where the precaution must be taken within 2 times the optimal result.

CHAPTER 6

Conclusions and Scope of Future Work

With the goal to study the hardness and provide an approximate result of some graph theoretic problems, **this thesis has achieved the following:**

- Studied the complexity of the Traveling Tournament Problem with a constraint on the maximum length of home stands and away trips of the participating teams and proved the NP-Hardness of the same when the value of the upper bound of the length is any natural number greater than 3.
- Closed the complexity analysis gap of constrained Traveling Tournament Problem.
- Presented an approximation algorithm of Traveling Tournament Problem-2 that give better result in practical cases than the existing best result when the number of participating teams is less than 32 and a multiple of 4.
- Studied the computational complexity of Firefighter Problem on Unit Disk Graphs and proved that the Min-Budget version of the problem is NP-Hard.
- Presented an approximation algorithm of Firefighter Problem on Unit Disk Graphs with an approximation factor of 2.

- Studied Firebreak Problem and gave an approximation algorithm of the problem on Split Graphs with an approximation factor of 2.

6.1 Future Scopes

The process of inquiry carried out in this thesis leads one to the **following questions worthy for further study:**

1. The Traveling Tournament Problem with Maximum Trip Length Two is assumed to be NP-Hard and approximation algorithms are attempted. There is no proof of the problem being NP-Hard. Can it be proved to be NP-Hard?

2. As the Traveling Tournament Problem-k has been proven to be NP-Hard for $k > 3$ and $k \in \mathbb{N}$, Can there be any efficient approximation algorithm for TTP-k?

3. Although the Firefighter Problem has been to be NP-Hard on general graphs and many different graph classes, the complexity analysis of the problem on several other graph classes can be attempted.

4. Can the approximation factor for the Firefighter Problem on Unit Disk Graphs be further improved than the one presented in this thesis?

5. Complexity analysis of the Firebreak problem on different classes can be tried.

6. Does there exist an approximation algorithm of the Firebreak Problem on Split Graph with an approximation factor less than 2?

APPENDIX A

Appendix

A.1 More Examples of Schedule

Schedules for $n = 20, 24, 28$ are given below for a better insight of our algorithm.

A.1.1 Schedule for $n = 20$

Vertex Set = $\{1, 2, \ldots, 19, 20\}$. Set of Pair of vertices = $\{M_1, M_2, \ldots, M_{10}\}$; where M_1={1,2}, M_2={3,4}, M_3={5,6}, M_4={7,8}, M_5={9,10}, M_6={11,12}, M_7={13,14}, M_8={15,16}, M_9={17,18}, M_{10}={19,20}; and N_1=$\{M_1, M_5\}$, N_2=$\{M_2, M_{10}\}$, N_3=$\{M_3, M_9\}$, N_4=$\{M_4, M_7\}$, N_5=$\{M_6, M_8\}$.

Table A.1: ***Schedule for Tournament with 20 Teams***

Round:1, Level:1	**Round:1, Level:2**	**Round:1, Level:3**
$M_1 \xrightarrow{Type-1} M_2$	$M_1 \xrightarrow{Type-1} M_4$	$M_1 \xrightarrow{Type-1} M_6$
$M_3 \xrightarrow{Type-1} M_4$	$M_3 \xrightarrow{Type-1} M_6$	$M_3 \xrightarrow{Type-2} M_8$
$M_5 \xrightarrow{Type-1} M_6$	$M_5 \xrightarrow{Type-1} M_8$	$M_5 \xrightarrow{Type-1} M_{10}$
$M_7 \xrightarrow{Type-1} M_8$	$M_7 \xrightarrow{Type-1} M_{10}$	$M_7 \xrightarrow{Type-2} M_2$
$M_9 \xrightarrow{Type-1} M_{10}$	$M_9 \xrightarrow{Type-1} M_2$	$M_9 \xrightarrow{Type-1} M_4$

Round:1, Level:4	**Round:1, Level:5**	**Round:2, Level:1**
$M_1 \xrightarrow{Type-1} M_3$	$M_1 \xrightarrow{Type-1} M_8$	$M_1 \xrightarrow{Type-2} M_7$
$M_8 \xrightarrow{Type-2} M_{10}$	$M_{10} \xrightarrow{Type-1} M_4$	$M_{10} \xrightarrow{Type-1} M_6$
$M_5 \xrightarrow{Type-1} M_7$	$M_5 \xrightarrow{Type-1} M_3$	$M_5 \xrightarrow{Type-2} M_4$
$M_2 \xrightarrow{Type-1} M_4$	$M_2 \xrightarrow{Type-1} M_6$	$M_2 \xrightarrow{Type-1} M_3$
$M_9 \xrightarrow{Type-1} M_6$	$M_9 \xrightarrow{Type-1} M_7$	$M_9 \xrightarrow{Type-1} M_8$

Round:2, Level:2	**Round:3, Level:1**	**Round:4, Level:1**
$M_7 \xrightarrow{Type-1} M_3$	$M_7 \xrightarrow{Type-1} M_6$	$M_7 \xrightarrow{Type-3} M_4$
$M_{10} \xrightarrow{Type-1} M_1$	$M_{10} \xrightarrow{Type-1} M_3$	$M_{10} \xrightarrow{Type-3} M_2$
$M_4 \xrightarrow{Type-1} M_6$	$M_4 \xrightarrow{Type-2} M_8$	$M_8 \xrightarrow{Type-3} M_6$
$M_2 \xrightarrow{Type-1} M_8$	$M_2 \xrightarrow{Type-2} M_5$	$M_5 \xrightarrow{Type-3} M_1$
$M_9 \xrightarrow{Type-1} M_5$	$M_9 \xrightarrow{Type-1} M_1$	$M_9 \xrightarrow{Type-3} M_3$

Number of *Flips*= $7 = \lfloor F_{20} \rfloor$.

A.1.2 Schedule for $n = 24$

Vertex Set =$\{1, 2, \ldots, 23, 24\}$. Set of Pair of vertices = $\{M_1, M_2, \ldots, M_{10}, M_{11}, M_{12}\}$; where M_1=$\{1,2\}$, M_2=$\{3,4\}$, M_3=$\{5,6\}$, M_4=$\{7,8\}$, M_5=$\{9,10\}$, M_6=$\{11,12\}$, M_7=$\{13,14\}$, M_8=$\{15,16\}$, M_9=$\{17,18\}$, M_{10}=$\{19,20\}$, M_{11}=$\{21,22\}$, M_{12}=$\{23,24\}$; and N_1=$\{M_9, M_5\}$, N_2=$\{M_1, M_7\}$, N_3=$\{M_{11}, M_3\}$, N_4=$\{M_{10}, M_6\}$, N_5=$\{M_2, M_4\}$, N_6=$\{M_8, M_{12}\}$.

Table A.2: ***Schedule for Tournament with 24 Teams***

Round:1, Level:1	**Round:1, Level:2**	**Round:1, Level:3**	**Round:1, Level:4**
$M_1 \xrightarrow{Type-1} M_2$	$M_1 \xrightarrow{Type-1} M_4$	$M_1 \xrightarrow{Type-1} M_6$	$M_1 \xrightarrow{Type-1} M_8$
$M_3 \xrightarrow{Type-1} M_4$	$M_3 \xrightarrow{Type-1} M_6$	$M_3 \xrightarrow{Type-1} M_8$	$M_3 \xrightarrow{Type-1} M_{10}$
$M_5 \xrightarrow{Type-1} M_6$	$M_5 \xrightarrow{Type-1} M_8$	$M_5 \xrightarrow{Type-1} M_{10}$	$M_5 \xrightarrow{Type-1} M_{12}$
$M_7 \xrightarrow{Type-1} M_8$	$M_7 \xrightarrow{Type-1} M_{10}$	$M_7 \xrightarrow{Type-1} M_{12}$	$M_7 \xrightarrow{Type-1} M_2$
$M_9 \xrightarrow{Type-1} M_{10}$	$M_9 \xrightarrow{Type-1} M_{12}$	$M_9 \xrightarrow{Type-1} M_2$	$M_9 \xrightarrow{Type-1} M_4$
$M_{11} \xrightarrow{Type-1} M_{12}$	$M_{11} \xrightarrow{Type-1} M_2$	$M_{11} \xrightarrow{Type-1} M_4$	$M_{11} \xrightarrow{Type-1} M_6$

Round:1, Level:5	Round:1, Level:6	Round:2, Level:1	Round:2, Level:2
$M_1 \xrightarrow{Type-1} M_{10}$	$M_1 \xrightarrow{Type-2} M_{12}$	$M_{12} \xrightarrow{Type-1} M_2$	$M_{12} \xrightarrow{Type-1} M_6$
$M_3 \xrightarrow{Type-1} M_{12}$	$M_3 \xrightarrow{Type-1} M_2$	$M_3 \xrightarrow{Type-1} M_1$	$M_3 \xrightarrow{Type-1} M_5$
$M_5 \xrightarrow{Type-1} M_2$	$M_5 \xrightarrow{Type-2} M_4$	$M_4 \xrightarrow{Type-1} M_6$	$M_4 \xrightarrow{Type-2} M_{10}$
$M_7 \xrightarrow{Type-1} M_4$	$M_7 \xrightarrow{Type-1} M_6$	$M_7 \xrightarrow{Type-1} M_5$	$M_7 \xrightarrow{Type-2} M_9$
$M_9 \xrightarrow{Type-1} M_6$	$M_9 \xrightarrow{Type-2} M_8$	$M_8 \xrightarrow{Type-1} M_{10}$	$M_8 \xrightarrow{Type-1} M_2$
$M_{11} \xrightarrow{Type-1} M_8$	$M_{11} \xrightarrow{Type-1} M_{10}$	$M_{11} \xrightarrow{Type-1} M_9$	$M_{11} \xrightarrow{Type-1} M_1$

Round:2, Level:3	Round:3, Level:1	Round:4, Level:1
$M_{12} \xrightarrow{Type-1} M_4$	$M_{12} \xrightarrow{Type-2} M_{10}$	$M_9 \xrightarrow{Type-3} M_5$
$M_3 \xrightarrow{Type-1} M_7$	$M_3 \xrightarrow{Type-2} M_9$	$M_1 \xrightarrow{Type-3} M_7$
$M_{10} \xrightarrow{Type-2} M_2$	$M_2 \xrightarrow{Type-1} M_6$	$M_{11} \xrightarrow{Type-3} M_3$
$M_9 \xrightarrow{Type-2} M_1$	$M_1 \xrightarrow{Type-1} M_5$	$M_{10} \xrightarrow{Type-3} M_6$
$M_8 \xrightarrow{Type-1} M_6$	$M_8 \xrightarrow{Type-1} M_4$	$M_2 \xrightarrow{Type-3} M_4$
$M_{11} \xrightarrow{Type-1} M_5$	$M_{11} \xrightarrow{Type-1} M_7$	$M_8 \xrightarrow{Type-3} M_{12}$

Number of *Flips*= $9 = F_{24}$.

A.1.3 Schedule for $n = 28$

Vertex Set ={1, 2,. . . , 27, 28}. Set of Pair of vertices ={M_1, M_2,. . . , M_{14}}; where M_1={1,2}, M_2={3,4}, M_3={5,6}, M_4={7,8}, M_5={9,10}, M_6={11,12}, M_7={13,14}, M_8={15,16}, M_9={17,18}, M_{10}={19,20}, M_{11}={21,22}, M_{12}={23,24}, M_{13}={25,26}, M_{14}={27,28}; and N_1={M_{14}, M_4}, N_2={M_{12}, M_6}, N_3={M_3, M_2}, N_4={M_8, M_{10}}, N_5={M_9, M_5}, N_6={M_7, M_{11}}, N_7={M_1, M_{13}}.

Table A.3: ***Schedule for Tournament with 28 Teams***

Round:1, Level:1	Round:1, Level:2	Round:1, Level:3	Round:1, Level:4
$M_1 \xrightarrow{Type-1} M_2$	$M_1 \xrightarrow{Type-1} M_4$	$M_1 \xrightarrow{Type-1} M_6$	$M_1 \xrightarrow{Type-1} M_8$
$M_3 \xrightarrow{Type-1} M_4$	$M_3 \xrightarrow{Type-1} M_6$	$M_3 \xrightarrow{Type-1} M_8$	$M_3 \xrightarrow{Type-1} M_{10}$
$M_5 \xrightarrow{Type-1} M_6$	$M_5 \xrightarrow{Type-1} M_8$	$M_5 \xrightarrow{Type-1} M_{10}$	$M_5 \xrightarrow{Type-1} M_{12}$
$M_7 \xrightarrow{Type-1} M_8$	$M_7 \xrightarrow{Type-1} M_{10}$	$M_7 \xrightarrow{Type-1} M_{12}$	$M_7 \xrightarrow{Type-1} M_{14}$
$M_9 \xrightarrow{Type-1} M_{10}$	$M_9 \xrightarrow{Type-1} M_{12}$	$M_9 \xrightarrow{Type-1} M_{14}$	$M_9 \xrightarrow{Type-1} M_2$
$M_{11} \xrightarrow{Type-1} M_{12}$	$M_{11} \xrightarrow{Type-1} M_{14}$	$M_{11} \xrightarrow{Type-1} M_2$	$M_{11} \xrightarrow{Type-1} M_4$
$M_{13} \xrightarrow{Type-1} M_{14}$	$M_{13} \xrightarrow{Type-1} M_2$	$M_{13} \xrightarrow{Type-1} M_4$	$M_{13} \xrightarrow{Type-1} M_6$

Round:1, Level:5	**Round:1, Level:6**	**Round:1, Level:7**
$M_1 \xrightarrow{Type-1} M_{12}$	$M_1 \xrightarrow{Type-1} M_{10}$	$M_1 \xrightarrow{Type-1} M_{14}$
$M_3 \xrightarrow{Type-1} M_{14}$	$M_3 \xrightarrow{Type-2} M_{12}$	$M_{12} \xrightarrow{Type-1} M_{10}$
$M_5 \xrightarrow{Type-1} M_2$	$M_5 \xrightarrow{Type-1} M_{14}$	$M_5 \xrightarrow{Type-1} M_7$
$M_7 \xrightarrow{Type-1} M_4$	$M_7 \xrightarrow{Type-2} M_2$	$M_2 \xrightarrow{Type-1} M_4$
$M_9 \xrightarrow{Type-1} M_6$	$M_9 \xrightarrow{Type-1} M_4$	$M_9 \xrightarrow{Type-1} M_{11}$
$M_{11} \xrightarrow{Type-1} M_8$	$M_{11} \xrightarrow{Type-2} M_6$	$M_6 \xrightarrow{Type-1} M_8$
$M_{13} \xrightarrow{Type-1} M_{10}$	$M_{13} \xrightarrow{Type-1} M_8$	$M_{13} \xrightarrow{Type-1} M_3$
Round:2, Level:1	**Round:2, Level:2**	**Round:2, Level:3**
$M_1 \xrightarrow{Type-1} M_7$	$M_1 \xrightarrow{Type-2} M_{11}$	$M_{11} \xrightarrow{Type-2} M_{10}$
$M_{12} \xrightarrow{Type-1} M_8$	$M_{12} \xrightarrow{Type-1} M_4$	$M_{12} \xrightarrow{Type-1} M_{14}$
$M_5 \xrightarrow{Type-1} M_4$	$M_5 \xrightarrow{Type-2} M_3$	$M_3 \xrightarrow{Type-1} M_1$
$M_2 \xrightarrow{Type-1} M_{10}$	$M_2 \xrightarrow{Type-1} M_{14}$	$M_2 \xrightarrow{Type-2} M_8$
$M_9 \xrightarrow{Type-1} M_3$	$M_9 \xrightarrow{Type-1} M_8$	$M_9 \xrightarrow{Type-1} M_7$
$M_6 \xrightarrow{Type-1} M_{14}$	$M_6 \xrightarrow{Type-1} M_{10}$	$M_6 \xrightarrow{Type-1} M_4$
$M_{13} \xrightarrow{Type-1} M_{11}$	$M_{13} \xrightarrow{Type-1} M_7$	$M_{13} \xrightarrow{Type-2} M_5$
Round:3, Level:1	**Round:3, Level:2**	**Round:4, Level:1**
$M_{10} \xrightarrow{Type-1} M_4$	$M_{10} \xrightarrow{Type-2} M_{14}$	$M_{14} \xrightarrow{Type-3} M_4$
$M_{12} \xrightarrow{Type-1} M_{13}$	$M_{12} \xrightarrow{Type-1} M_2$	$M_{12} \xrightarrow{Type-3} M_6$
$M_3 \xrightarrow{Type-1} M_7$	$M_3 \xrightarrow{Type-1} M_{11}$	$M_3 \xrightarrow{Type-3} M_2$
$M_8 \xrightarrow{Type-1} M_{14}$	$M_8 \xrightarrow{Type-1} M_4$	$M_8 \xrightarrow{Type-3} M_{10}$
$M_9 \xrightarrow{Type-1} M_1$	$M_9 \xrightarrow{Type-1} M_{13}$	$M_9 \xrightarrow{Type-3} M_5$
$M_6 \xrightarrow{Type-1} M_2$	$M_6 \xrightarrow{Type-2} M_7$	$M_7 \xrightarrow{Type-3} M_{11}$
$M_5 \xrightarrow{Type-1} M_1$	$M_5 \xrightarrow{Type-2} M_1$	$M_1 \xrightarrow{Type-3} M_{13}$

Number of *Flips*= $11 = \lceil F_{28} \rceil$.

A.2 Tabular IPL Schedule

In this section, we present a schedule of **Indian Premier League(IPL)** using our algorithm1. IPL is a Double Round-robin Tournament of eight teams. The proposed schedule is presented in the following table.

Table A.4: ***Proposed Indian Premier League Schedule***

Match Day	**1**
Away	*Home*
MUM	KOL
HYD	RAJ
CHE	DEL
BANG	PUN

Match Day	**2**
Away	*Home*
MUM	RAJ
HYD	KOL
CHE	PUN
BANG	DEL

Match Day	**3**
Away	*Home*
RAJ	HYD
KOL	MUM
DEL	CHE
PUN	BANG

Match Day	**4**
Away	*Home*
RAJ	MUM
KOL	HYD
DEL	BANG
PUN	CHE

Match Day	**5**
Away	*Home*
MUM	DEL
HYD	PUN
CHE	KOL
BANG	RAJ

Match Day	**6**
Away	*Home*
MUM	PUN
HYD	DEL
RAJ	CHE
KOL	BANG

Match Day	**7**
Away	*Home*
PUN	HYD
DEL	MUM
RAJ	BANG
KOL	CHE

Match Day	**8**
Away	*Home*
DEL	HYD
PUN	MUM
CHE	RAJ
BANG	KOL

Match Day	**9**
Away	*Home*
MUM	CHE
HYD	BANG
KOL	DEL
RAJ	PUN

Match Day	**10**
Away	*Home*
MUM	HYD
KOL	RAJ
BAG	CHE
PUN	DEL

Match Day	**11**
Away	*Home*
BANG	MUM
CHE	HYD
PUN	KOL
DEL	RAJ

Match Day	**12**
Away	*Home*
HYD	MUM
RAJ	KOL
CHE	BANG
DEL	PUN

Match Day	**13**
Away	*Home*
MUM	BANG
HYD	CHE
KOL	PUN
RAJ	DEL

Match Day	**14**
Away	*Home*
CHE	MUM
BANG	HYD
DEL	KOL
PUN	RAJ

KOL → Kolkata Knight Riders	**MUM** → Mumbai Indians
CHE → Chennai Super Kings	**BANG** → Royal Challengers Bangalore
RAJ → Rajasthan Royals	**DEL** → Delhi Capitals
HYD → Sunrisers Hyderabad	**PUN** → Kings XI Punjab

This schedule gives 15% better result than the actual **IPL-2019** schedule in terms of total distance traveled by all the teams.

References

[1] R. Bhattacharyya, "Complexity of the unconstrained traveling tournament problem," *Operations Research Letters*, vol. 44, no. 5, pp. 649–654, 2016.

[2] C. Thielen and S. Westphal, "Complexity of the traveling tournament problem," *Theoretical Computer Science*, vol. 412, no. 4-5, pp. 345–351, 2011.

[3] D. De Werra, "Some models of graphs for scheduling sports competitions," *Discrete Applied Mathematics*, vol. 21, no. 1, pp. 47–65, 1988.

[4] M. Xiao and S. Kou, "An improved approximation algorithm for the traveling tournament problem with maximum trip length two," in *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[5] K. Easton, G. Nemhauser, and M. Trick, "The traveling tournament problem description and benchmarks," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2001, pp. 580–584.

[6] D. Briskorn, A. Drexl, and F. C. Spieksma, "Round robin tournaments and three index assignments," *4OR*, vol. 8, no. 4, pp. 365–374, 2010.

[7] S. Imahori, T. Matsui, and R. Miyashiro, "A 2.75-approximation algorithm for the unconstrained traveling tournament problem," *Annals of Operations Research*, vol. 218, no. 1, pp. 237–247, 2014.

[8] R. Hoshino and K.-i. Kawarabayashi, "An approximation algorithm for the bipartite traveling tournament problem," *Mathematics of Operations Research*, vol. 38, no. 4, pp. 720–728, 2013.

[9] R. Miyashiro, T. Matsui, and S. Imahori, "An approximation algorithm for the traveling tournament problem," *Annals of Operations Research*, vol. 194, no. 1, pp. 317–324, 2012.

[10] S. Westphal and K. Noparlik, "A 5.875-approximation for the traveling tournament problem," *Annals of Operations Research*, vol. 218, no. 1, pp. 347–360, 2014.

[11] D. Yamaguchi, S. Imahori, R. Miyashiro, and T. Matsui, "An improved approximation algorithm for the traveling tournament problem," *Algorithmica*, vol. 61, no. 4, p. 1077, 2011.

[12] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados, "A simulated annealing approach to the traveling tournament problem," *Journal of Scheduling*, vol. 9, no. 2, pp. 177–193, 2006.

[13] L. Di Gaspero and A. Schaerf, "A composite-neighborhood tabu search approach to the traveling tournament problem," *Journal of Heuristics*, vol. 13, no. 2, pp. 189–207, 2007.

[14] K. Easton, G. Nemhauser, and M. Trick, "Solving the travelling tournament problem: A combined integer programming and constraint programming approach," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2002, pp. 100–109.

[15] M. Goerigk, R. Hoshino, K.-i. Kawarabayashi, and S. Westphal, "Solving the traveling tournament problem by packing three-vertex paths," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[16] A. Lim, B. Rodrigues, and X. Zhang, "A simulated annealing and hill-climbing algorithm for the traveling tournament problem," *European Journal of Operational Research*, vol. 174, no. 3, pp. 1459–1478, 2006.

[17] M. Trick, "Challenge traveling tournament instances," *http://mat. tepper. cmu. edu/TOURN/*, 1999.

[18] P. Van Hentenryck and Y. Vergados, "Population-based simulated annealing for traveling tournaments," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007, p. 267.

[19] R. V. Rasmussen and M. A. Trick, "Round robin scheduling–a survey," *European Journal of Operational Research*, vol. 188, no. 3, pp. 617–636, 2008.

[20] N. Fujiwara, S. Imahori, T. Matsui, and R. Miyashiro, "Constructive algorithms for the constant distance traveling tournament problem," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, 2006, pp. 135–146.

[21] R. Hoshino and K.-i. Kawarabayashi, "The linear distance traveling tournament problem," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[22] G. Kendall, S. Knust, C. C. Ribeiro, and S. Urrutia, "Scheduling in sports: An annotated bibliography," *Computers & Operations Research*, vol. 37, no. 1, pp. 1–19, 2010.

[23] R. T. Campbell and D. Chen, "A minimum distance basketball scheduling problem," *Management science in sports*, vol. 4, pp. 15–26, 1976.

[24] C. Thielen and S. Westphal, "Approximation algorithms for ttp (2)," *Mathematical Methods of Operations Research*, vol. 76, no. 1, pp. 1–20, 2012.

[25] T. Benoist, F. Laburthe, and B. Rottembourg, "Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems," in *Proceedings CPAIOR*, vol. 1. Citeseer, 2001, pp. 15–26.

[26] S. Imahori, T. Matsui, and R. Miyashiro, "An approximation algorithm for the unconstrained traveling tournament problem," *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, PATAT*, 2010.

[27] C. Thielen and S. Westphal, "Approximating the traveling tournament problem with maximum tour length 2," in *International Symposium on Algorithms and Computation*. Springer, 2010, pp. 303–314.

[28] D. De Werra, "Scheduling in sports," *Studies on graphs and discrete programming*, vol. 11, pp. 381–395, 1981.

[29] S. Finbow and G. MacGillivray, "The firefighter problem: a survey of results, directions and questions." *Australasian J. Combinatorics*, vol. 43, pp. 57–78, 2009.

[30] B. Hartnell, "Firefighter! an application of domination," in *the 24th Manitoba Conference on Combinatorial Mathematics and Computing, University of Minitoba, Winnipeg, Cadada, 1995*, 1995.

[31] ——, "Firefighting on trees: how bad is the greedy algorithm?" *Congressus Numerantium*, vol. 145, pp. 187–192, 2000.

[32] P. Wang and S. A. Moeller, "Fire control on graphs," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 41, pp. 19–34, 2002.

[33] M. Develin and S. G. Hartke, "Fire containment in grids of dimension three and higher," *Discrete Applied Mathematics*, vol. 155, no. 17, pp. 2257–2268, 2007.

[34] G. MacGillivray and P. Wang, "On the firefighter problem," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 47, pp. 83–96, 2003.

[35] S. G. Hartke, "Attempting to narrow the integrality gap for the firefighter problem on trees." in *Discrete Methods in Epidemiology*, 2004, pp. 225–232.

[36] S. Finbow, A. King, G. MacGillivray, and R. Rizzi, "The firefighter problem for graphs of maximum degree three," *Discrete Mathematics*, vol. 307, no. 16, pp. 2094–2105, 2007.

[37] J. Chlebíková and M. Chopin, "The firefighter problem: A structural analysis," in *International Symposium on Parameterized and Exact Computation*, 2014, pp. 172–183.

[38] C. Bazgan, M. Chopin, and M. R. Fellows, "Parameterized complexity of the firefighter problem," in *International Symposium on Algorithms and Computation*. Springer, 2011, pp. 643–652.

[39] M. Cygan, F. V. Fomin, and E. J. van Leeuwen, "Parameterized complexity of firefighting revisited," in *International Symposium on Parameterized and Exact Computation*. Springer, 2011, pp. 13–26.

[40] E. Anshelevich, D. Chakrabarty, A. Hate, and C. Swamy, "Approximation algorithms for the firefighter problem: Cuts over time and submodularity," in *International Symposium on Algorithms and Computation*. Springer, 2009, pp. 974–983.

[41] F. V. Fomin, P. Heggernes, and E. J. van Leeuwen, "Making life easier for firefighters," in *International Conference on Fun with Algorithms*. Springer, 2012, pp. 177–188.

[42] ——, "The firefighter problem on graph classes," *Theoretical Computer Science*, vol. 613, pp. 38–50, 2016.

[43] P. Coupechoux, M. Demange, D. Ellison, and B. Jouve, "Firefighting on trees," *Theoretical Computer Science*, vol. 794, pp. 69–84, 2019.

[44] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete mathematics*, vol. 86, no. 1-3, pp. 165–177, 1990.

[45] A. King and G. MacGillivray, "The firefighter problem for cubic graphs," *Discrete Mathematics*, vol. 310, no. 3, pp. 614–621, 2010.

[46] P. Chalermsook and J. Chuzhoy, "Resource minimization for fire containment," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2010, pp. 1334–1349.

[47] K. D. Barnetson, A. C. Burgess, J. Enright, J. Howell, D. A. Pike, and B. Ryan, "The firebreak problem," *Networks*, vol. 77, no. 3, pp. 372–382, 2021.

[48] S. P. Borgatti, *The key player problem.* na, 2003.

[49] ——, "Identifying sets of key players in a social network," *Computational & Mathematical Organization Theory*, vol. 12, no. 1, pp. 21–34, 2006.

[50] C. Ballester, A. Calvó-Armengol, and Y. Zenou, "Who's who in networks. wanted: The key player," *Econometrica*, vol. 74, no. 5, pp. 1403–1417, 2006.

[51] D. Ortiz-Arroyo and D. Hussain, "An information theory approach to identify sets of key players," in *European Conference on Intelligence and Security Informatics*. Springer, 2008, pp. 15–26.

[52] M. M. Sathik and A. A. Rasheed, "A centrality approach to identify sets of key players in an online weblog," *International Journal of Recent Trends in Engineering*, vol. 2, no. 3, p. 85, 2009.

[53] B. Addis, M. Di Summa, and A. Grosso, "Identifying critical nodes in undirected graphs: Complexity results and polynomial algorithms for the case of bounded treewidth," *Discrete Applied Mathematics*, vol. 161, no. 16-17, pp. 2349–2360, 2013.

[54] A. Berger, A. Grigoriev, and R. van der Zwaan, "Complexity and approximability of the k-way vertex cut," *Networks*, vol. 63, no. 2, pp. 170–178, 2014.

[55] D. Marx, "Parameterized graph separation problems," *Theoretical Computer Science*, vol. 351, no. 3, pp. 394–406, 2006.

[56] S. Shen and J. C. Smith, "Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs," *Networks*, vol. 60, no. 2, pp. 103–119, 2012.

[57] M. Lalou, M. A. Tahraoui, and H. Kheddouci, "The critical node detection problem in networks: A survey," *Computer Science Review*, vol. 28, pp. 92–117, 2018.

[58] N. Baruah and A. K. Baruah, "On a traffic control problem using cut-set of graph," *International Journal of Advanced Networking and Applications*, vol. 3, no. 4, p. 1240, 2012.

[59] A. V. Karzanov and E. A. Timofeev, "Efficient algorithm for finding all minimal edge cuts of a nonoriented graph," *Cybernetics*, vol. 22, no. 2, pp. 156–162, 1986.

[60] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.