# The Traveling Tournament Problem

by

Salomon Bendayan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics & Optimization

Waterloo, Ontario, Canada, 2022

## Abstract

In this thesis we study the Traveling Tournament problem (TTP) which asks to generate a feasible schedule for a sports league such that the total travel distance incurred by all teams throughout the season is minimized. Throughout our three technical chapters a wide range of topics connected to the TTP are explored.

We begin by considering the computational complexity of the problem. Despite existing results on the NP-hardness of TTP, the question of whether or not TTP is also APX-hard was an unexplored area in the literature. We prove the affirmative by constructing an $L$-reduction from $(1, 2)$-TSP to TTP. To reach the desired result, we show that given an instance of TSP with a solution of cost $K$, we can construct an instance of TTP with a solution of cost at most $20m(m + 1)cK$ where $m = c(n - 1) + 1$, $n$ is the number of teams, and $c > 5, c \in \mathbb{Z}$ is fixed. On the other hand, we show that given a feasible schedule to the constructed TTP instance, we can recover a tour on the original TSP instance.

The next chapter delves into a popular variation of the problem, the mirrored TTP, which has the added stipulation that the first and second half of the schedule have the same order of match-ups. Building upon previous techniques, we present an approximation algorithm for constructing a mirrored double round-robin schedule under the constraint that the number of consecutive home or away games is at most two. We achieve an approximation ratio on the order of $3/2 + O(1)/n$.

Lastly, we present a survey of local search methods for solving TTP and discuss the performance of these techniques on benchmark instances.

**Acknowledgements**

I would like to thank my supervisor Joseph Cheriyan for his guidance throughout my degree. I would also like to thank Professor Kevin Cheung of Carleton University for graciously lending his time and expertise on this project.

I am grateful to Chahira, Cami, Enrique, Vanessa, Amy, Abbey and all the friends I made along the way for making my time in Waterloo a special and memorable experience. Lastly, I am forever indebted to my parents for their continued support and unwavering belief in me.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Introduction

The Traveling Tournament Problem (TTP) was introduced by Easton, Nemhauser, and Trick [13] and inspired by Trick's work done for Major League Baseball (USA). Trick and his partner, baseball executive Doug Bureman, worked on creating better schedules for the MLB since 1996. The problem quickly grew in popularity as other sports leagues realized adopting a mathematical method of generating season schedules could greatly decrease operational costs. Satisfying leagues requirements has increased in difficulty as leagues are increasing in number of teams, number of games played by each team, and geographic dispersion of teams. As professional sports leagues have more specific demands ranging from maximizing television viewership, to minimizing operating costs for teams, it has become in a sports league's best interest to invest in generating optimal season schedules. Moreover, as sports science evolved and the importance of proper rest for top athletes became more apparent, professional sports leagues seek to generate season schedules that minimize travel time for players.

We have described a broad problem of generating optimal schedules, however, the notion of optimality may vary considerably depending on our objectives. As we will see in the following section, our goal in this thesis will be to minimize the traveling distance incurred by all teams over a full season. Despite the inherently applied nature of the problem, we do not concern ourselves with real-world examples. While we do discuss some computational results, we focus on the theoretical aspects of the problem which spark our interest as purely mathematical research.

In this chapter we begin by introducing the problem as it was proposed originally and some solution methods are discussed. Variants of the problem and the progress made on these versions is presented. Finally, we discuss some interesting problems related to TTP.

## 1.2 The Traveling Tournament Problem

Given $n$ teams, with $n$ being even, a *round-robin tournament* is a series of games played between the teams such that every team plays every other team. A $m$ round-robin tournament is defined the same way with the added stipulation that every team plays every other team $m$ times. Each game is designated to be played in one team's home venue, making them the *home team*, while their opponent is the *away team*.

A series of consecutive away games is called a *road trip*, while consecutive home games is a *home stand*. At the beginning of the tournament, all teams are at home, and must return home at the conclusion of the tournament.

The distances between all team venues are given in an $n \times n$ matrix $D$. When playing an away game, a team travels from their home city to their opponent's city. When playing consecutive away games, they travel directly from one opponent's city to the next. At the conclusion of their road trip, they return to their home city.

Note that the length of a road trip (or home stand) is given by the number of opponents played and not the distance travelled by the team.

The *Traveling Tournament Problem (TTP)* can be succinctly captured as follows:

**Input:** An even number of teams, $n$; an $n$ by $n$ integral distance matrix, $D$; $L, U$ integer parameters.

**Output:** A double round-robin tournament on the $n$ teams such that:

- every team plays every other team once at their home venue and once at the venue of the opponent
- the length of every home stand and road trip is between $L$ and $U$, and
- the total distance travelled by the teams is minimized.

A schedule satisfying the first constraint is a feasible double round-robin tournament. The second constraint is called the *at-most* constraint. The final constraint ensures optimality. There may be an additional constraint placed on the tournament:

- *No repeaters:* there are no teams $i, j$ such that in the tournament schedule, $i$ plays at $j$, followed immediately by $j$ plays at $i$.

This last constraint is quite natural to have in the interest of fans who want to see different match-ups on consecutive days.

Note that for $L = 1$ and $U = n - 1$ a team may take a road trip visiting all other teams, which equates to finding a traveling salesman tour. On the other hand, for small $U$, teams must return home often thus the distance travelled will increase. The parameter $L$ is taken to be 1 throughout this thesis thus leaving us with the ability to choose the parameter $U$. The TTP was originally introduced with $U = 3$ but we will explore the problem for different values. We rename the parameter $k$ and the version of the problem being discussed is distinguished by using the notation TTP($k$).

Over the years, a growing number of variants and instances of TTP were introduced. Here are several classes of instances which have become popular benchmarks:

The circle instance is one where $n$ cities correspond to the nodes of a circle graph of size $n$. Nodes $i$ and $i + 1$, and nodes $n - 1$ and 0 are joined by an edge of length 1. Then the shortest distance from $i$ to $j$ ($i > j$) is the minimum of $i - j$ and $j - i + n$. A circle instance with $n$ teams is denoted CIRC$n$.

The National League instance with $n$ teams is denoted NL$n$, and is given by $n$ teams of the Major League Baseball league and distances are given by the "air distance" from city centers. Optimal solutions to NL4 and NL6 are given in [13]. NL8 was solved optimally for the first time in [14] using a branch and price algorithm in parallel with constraint programming although the no-repeaters condition was neglected.

The constant distance instance (denoted CON$n$) was introduced by Ribeiro and Urrutia [32] and the problem is formulated just as we would expect: the distance between all pairs of cities is constant, typically equal to one. Fujiwara, Imahori, Matsui, and Miyashiro [15] proposed a lower bound for the constant distance TTP and devised two approximation algorithms that produce feasible solutions with values close to their lower bound.

A description of these instances as well as the most current progress on them can be found at the web-page maintained by Trick [31].

### 1.2.1 Unconstrained TTP

The unconstrained TTP (uTTP) is formulated identically to the original TTP except we do not enforce the at-most constraint. Thus we do not impose a bound on the maximum length of home stands and road trips. In this formulation, a team can take an arbitrarily long road trip, possibly visiting all other teams, before returning home. The unconstrained version of the problem was shown to be NP-hard [2], and when the number of consecutive home/away games is fixed to three, the problem is strongly NP-complete [29]. NP-completeness was later shown for all fixed values of $k > 3$ [6].

### 1.2.2 Mirrored TTP

The mirrored TTP (mTTP) is a widely studied variation of the problem. The first paper on mTTP is due to Ribeiro and Urrutia [27]. This version includes the additional constraint that the schedule of the first half of the season is the same as that of the second half of the season, but with the venues reversed. If team $A$ plays at city $B$ on the first day of the first half of the season, then team $B$ would play at city $A$ on the first day of the second half of the season. This version of the problem is of interest to the sports scheduling community due to being a common tournament structure in Latin America.

Cheung [8] used a two-phase method based on generating timetables from 1-factorizations and finding optimal home/away assignments to solve NL8 and CIRC8 to optimality. Moreover, Cheung used a Benders approach to compute lower bounds for the mirrored TTP [9]. These bounds remain the current best-known lower bounds for the mirrored problem.

Incidentally, several key results for the mirrored TTP were not intended to specifically solve the mirrored problem. The fact that the resulting schedule was mirrored was merely a byproduct of the method used to construct it. For that reason, there are a number of important results on mTTP which are discussed in other sections.

## 1.3 Approximation Algorithms

TTP was originally introduced with a bound $k$ on the number consecutive home or away games a team can play. Due to the influence of Major League Baseball on the formulation of the problem, $k$ was taken to be 3. However, the problem for other values of $k$ has been studied as well. We now present the current best approximations for the problem for different values of $k$. We assume the distances between team venues satisfy the triangle inequality thus the distances are metric.

Imahori, Matsui, and Miyashiro [19] proposed a constant factor approximation algorithm for the unconstrained TTP (i.e. $k = n - 1$, $n$ is the number of teams) with approximation ratio 2.75. If a shortest Hamilton cycle on the cities is known, the approximation ratio is improved to 2.25. The solution returned by the algorithm is a mirrored schedule thus it is also a feasible solution to the unconstrained mTTP.

There is no feasible schedule when $k = 1$ [11].

Campbell and Chen [5] studied the problem for $k = 2$ in 1976, preceding the formal definition of the TTP. Since then there have been many attempts to get a good approximation ratio, but the result depends on the parity of $n/2$.

Thielen and Westphal [30] devised an algorithm for each case and found a $3/2+O(1)/n$ bound for $n/2$ odd and a $1+O(1)/n$ bound for $n/2$ even. Imahori [18] built on this method and proved a $1 + O(1)/n$ bound when $n/2$ is odd. Zhao and Xiao [36] also achieved a $1 + O(1)/n$ ratio for $n/2$ odd which has a slightly better ratio ($1 + 12/n$ versus $1 + 24/n$) than found by Imahori.

In the case where $n/2$ is even, Xiao and Kou [34] gave an approximation algorithm with approximation ratio of $(1 + 1/4n)$. Chaterjee and Roy [7] found an approximation ratio of $1 + \frac{\lceil \log_2 n/4 \rceil + 4}{2(n-2)}$ which yields slightly better results than that of Xiao and Kou when $n \leq 32$.

For $k = 3$, the most popular version of the problem, Imahori, Matsui, and Miyashiro [20] proposed a randomized approximation algorithm which yields a feasible solution whose approximation ratio is less than $2 + (9/4)/(n - 1)$. They then used a derandomization technique to find a deterministic approximation algorithm with the same approximation ratio. These algorithms were the first with constant approximation ratio, which is at most 2.75. The result relies on a new lower bound that is not as tight as previously known lower bounds, but easier to calculate. This new bound implies that any feasible solution to a TTP(3) instance is at most three times the optimal value of the instance. In fact, their lower bound yields a trivial approximation ratio of $k$ for any feasible TTP($k$) solution.

When $k \geq 4$, Westphal and Noparlik [33] proposed an approximation algorithm which produces a solution no more than 5.875 times the optimal for any choice of $n \geq 6$.

Yamaguchi, Imahori, Matsui, and Miyashiro [35] considered TTP($k$) for all fixed $k \geq 3$. When $k \leq 5$, the approximation ratio of the proposed algorithm is bounded by $(2k-1)/k+O(k/n)$, when $k > 5$, the ratio is bounded by $(5k - 7)/(2k) + O(k/n)$. For $k = 3$, the approximation ratio of the proposed algorithm is $5/3+O(1)/n$. The schedule produced by their algorithm is a mirrored schedule.

The Bipartite Traveling Tournament Problem (BTTP) was introduced by Hoshino and Kawarabayashi [16] where $2n$ teams are split into two groups of $n$ teams. Each team plays one home and one away game against all other teams in the other group. This variant was introduced to model the scheduling of games for the Nippon Professional Baseball (NPB) league, Japan's largest sports league. Each NPB team plays 24 inter-league games each season thus the bipartite structure of games between two groups was relevant. The problem is NP-complete. Despite its computational hardness, an optimal schedule for inter-league games was found for the NPB which is made up of only 12 teams.

Hoshino and Kawarabayashi [17] produced a polynomial-time algorithm to solve the BTTP with an approximation ratio of $1 + 2c/3 + (3 - c)/3n$, where $c$ is the approximation factor of the TSP.

## 1.4 Computational Methods for TTP

### 1.4.1 Lower Bound & Solution Methods

It is possible to generate a lower bound on the optimal solution to TTP by finding the minimum distance teams must travel regardless of other constraints. Solving for the minimum distance travelled by a single team $i$ requires finding a collection of tours starting from $i$'s home city, each with length between $L$ and $U$, such that all other cities are visited. Summing up the minimum distance for all teams yields a lower bound on TTP which is referred to as the *Independent Lower Bound (ILB)*. Note that finding the ILB involves solving a vehicle routing problem.

### 1.4.2 Integer Programming

Two methods were proposed in Easton et al. [13] to produce feasible solutions and prove optimality: constraint programming, and integer programming. We will not cover the constraint programming approach in this thesis.

By adding additional constraints to the sub-problem we solve to find the ILB, we can fully formulate the TTP. Trick proposes generating all possible road trips of appropriate length and assigning a decision variable $x_j$ to each, which determines if the road trip $j$ is selected or not.

The constraints of the problem are given as:

1. Each team must play every other team twice.
2. Each team can play only one game per day.
3. Teams cannot have consecutive road trips.

Given the ILB, we can also require that a team travels at least its minimum distance. This cuts down on the number of road trips we need to explore, thus reducing the computation time.

### 1.4.3 Local Search Methods

Given an optimization problem that is computationally hard to solve, local search methods may be used to find an optimal solution. For the local search methods studied in this thesis, we assume that each instance of the problem has a discrete set of feasible solutions. A local search algorithm starts from a candidate solution and then iteratively moves to a neighboring solution until a stopping criterion is met. Each local search algorithm is therefore defined by how the neighborhood of solutions is chosen, as well as its stopping criterion.

Several heuristic methods were implemented to achieve good results on benchmark instances of TTP. The first example of these methods being applied to TTP was a simulated annealing technique used by Anagnostopoulos et al. [1] to solve TTP(3). A greedy iterative search method was proposed by Ribeiro and Urrutia [27] to solve the mirrored TTP. More recently, Khelifa and Boughaci [22] proposed a Variable Neighborhood search technique to solve TTP(3). They then went on to achieve even better results for the unconstrained TTP using an Evolutionary Harmony search algorithm [23]. Further discussion of local search methods as well as the papers mentioned above can be found in Chapter 4.

## 1.5 Related Problems

**Vehicle Routing Problem (VRP) :** In the Vehicle Routing Problem (sometimes called the multiple TSP), there is a set of depots, clients, and vehicles and the goal is to optimally design routes for the vehicles from the depots to the clients. The problem is defined by a complete graph $G = (V, E)$, and a cost function $c : E \to \mathbb{R}_+$. We choose the depot vertex without loss of generality to be $v_0$, while the remaining vertices are client vertices. An optimal solution to VRP is a minimum cost collection of tours such that each client is visited exactly once by exactly one vehicle where vehicles start and end their tours at a depot, see Laporte [24].

**Referee assignment :** Duarte, Ribeiro, Urrutia, and Haeusler [12] consider the Referee Assignment problem which is the problem of assigning referees to oversee all games in an already scheduled tournament. Referees provide a max number of games they are willing to oversee, and a target number of games they hope to be assigned to. The goal is then to assign a referee to each game of the tournament such that, over all referees, the difference between their target number and the actual number of games they are assigned is minimized. Additionally, the total idle time of all referees should be minimized. Idle time is the time a referee must wait between consecutive games they are assigned to. A commercial solver (CPLEX 9.0) was used to compute the complete minimum Pareto set for medium-sized problems, and give results for an instance with 50 games and 100 referees.

**Break minimization with broadcast objectives :** Ribeiro and Urrutia [26] consider the following problem with two objectives: minimizing the number of breaks (which ensures fairness) and maximizing the revenue due to broadcasting of popular games. A break occurs whenever a team plays two consecutive games at home or away.

The problem is considered in the context of the Brazilian soccer league. Because of this, there are particular constraints that are specific to the league's requirements, for example, the soccer league has a mirrored schedule. One way to generalize the broadcast objective is to assign a popularity score to each team. The popularity of a game is then the sum of the scores of the participating teams. A new variable that determines if a game is televised or not is introduced. Given a fixed number of games to be televised, the new additional objective is then to maximize the popularity of televised games.

To tackle the bi-objective problem, an extra constraint is added stipulating that the number of breaks reaches its minimum. The broadcast objective is then maximized subject to this restriction. If the maximum objective of the restricted problem is equal to the unconstrained maximum, this solution is called an ideal point, meaning all objectives are simultaneously at their individual optimal values. If the maximum value of the unconstrained problem is greater than the maximum value of the restricted problem, this solution is not an ideal point, however it is still a non-dominated solution, meaning we cannot find a solution which improves an objective without trading a decrease in the other objective. Note that the authors assume that there exists some solution such that the number of breaks reaches its minimum. This is a fair assumption to make in this problem as this fact was experimentally verified on all test instances and their variations.

An integer programming formulation of the problem could not be solved in reasonable time by CPLEX, so a multi-phase decomposition algorithm was proposed. This process involves generating all home-away patterns that can feasibly be mirrored then solving an integer program to find the maximum objective value of each home-away pattern.

**Scheduling with absences :** Schauz [28] considers the problem of scheduling a tournament where each team has a pre-specified number of allowable absences, $t$. The goal is then to determine how many rounds are necessary to schedule the whole tournament in the worst case. The problem is approached as a graph coloring problem. When scheduling a round-robin tournament, we only consider complete multigraphs where the number of edges between two nodes is the number of times they must compete against each other. Each color represents a round of the schedule, thus it follows that a legal coloring of the multigraph defines a feasible schedule since each team is assigned to play exactly one game per round. A vertex labeling $c : V \rightarrow \mathbb{Z}^+$ is used to denote single absences of a team, i.e. what round they are unavailable, so $c(v) = 5$ means team $v$ is absent at round 5. An edge coloring $c' : E \rightarrow Z^+$ determines the round a match-up $e = (u, v)$ is played. The goal is therefore to find a proper coloring $c'$ such that $c'(uv)$ differs from $c(u)$ and $c(v)$ for all $uv \in E$. This ensures that $c'$ *avoids* $c$, which means a match-up between $u$ and $v$ is not scheduled on a day that one of them is absent.

$\chi_c^{'}(G)$ is defined as the *c-avoiding chromatic index* of $G$ which is the least number of rounds necessary to arrange all match-ups $e \in E$ such that $c'$ avoids $c$. To account for all $t$ possible absences of a single team, a multi-labelling of $v \in V$ is defined so each vertex may be assigned multiple labels. A multi-labelling is a *t-labelling* if $|c(v)| \leq t \ \forall v \in V$. Then the *t-avoiding chromatic index* of $G$ is $\chi^t(G) = \max\{\chi_c^{'} \ | c \text{ is a } t\text{-labelling}\}$.

For any graph $G$ such that $|E| > 0$, we have the general lower bound $\chi^t \geq \Delta(G) + 2t$. Since we are interested in round-robin schedules, we are interested in complete graphs. For pre-announced absences, we have

$$n + 2t - 1 \leq \chi^t(K_n) \leq n + 2t$$

Moreover, for $n \geq 2$, $\chi^1(K_n) = n + 1$.

For unannounced absences, we have $\chi_{OL}^t \leq n + 2t$, where $\chi_{OL}^t$ is the *on-line* $t$-avoiding chromatic index i.e. the information on absences are not announced and thus not known beforehand.

Schauz conjectures that $\chi^t(G) = \Delta(G) + 2t$ and $\chi_{OL}^t = \chi^{'}(G) + 2t$. The result is proven for all bipartite graphs, but not for all general graphs.

# Chapter 2

# APX Hardness of Unconstrained TTP

## 2.1 Overview

Ever since its proposal in 2001, TTP was suspected to be computationally hard to solve due to the similarity to the traveling salesman problem (TSP), which is well known to be strongly NP-hard and has become an important benchmark problem in computer science. The first NP-completeness proof for the unconstrained version of TTP was given by Bhattacharyya [2]. In this variant, the at-most constraint is ignored thus allowing teams to have unlimited consecutive home or away games, which in turn allows a reduction from TSP. Not long after, a reduction from 3-SAT was used to prove NP-hardness of TTP(3) by Thielen and Westphal [29]. More recently, TTP($k$) was shown to be NP-complete for all fixed $k > 3$ where the author used a similar approach by using a reduction from $k$-SAT [6].

The results stated above are examples of reductions that preserve hardness and they are used to prove that new problems are hard to solve by using reductions from known NP-hard problems. We can take this one step further and prove that problems are hard to approximate within a certain factor by using reductions that preserve approximation. These are reductions from a problem $\Pi$ to another problem $\Pi'$ such that if there is an approximation algorithm with approximation ratio $\alpha$ for problem $\Pi'$, then there is an approximation algorithm with approximation ratio $f(\alpha)$ for problem $\Pi$, where $f$ is some function. These results yield hardness theorems via the contrapositive: if there is no $f(\alpha)$-approximation algorithm for problem $\Pi$ unless P = NP, then there is no $\alpha$-approximation algorithm for problem $\Pi'$ unless P = NP. We can then use approximation preserving reductions from problems that are known to be hard to approximate to derive hardness results for a number of other problems. If we know that $\Pi$ is hard to approximate within some factor, the reduction implies that $\Pi'$ is also hard to approximate within some factor. Such a reduction is called an *L-reduction* and we define it formally now.

**Definition 1.** Given two optimization problems $\Pi$ and $\Pi'$, we say we have an L-reduction from $\Pi$ to $\Pi'$ if for some $a, b > 0$:

1. For each instance $I$ of $\Pi$ we can compute in polynomial time an instance $I'$ of $\Pi'$.
2. $\text{OPT}(I') \leq a{\cdot}\text{OPT}(I)$.
3. Given a solution of value $V'$ to $I'$, we can compute in polynomial time a solution of value $V$ to $I$ such that

$$|OPT(I) - V| \leq b|OPT(I') - V'|$$

Given an $L$-reduction between two minimization problems $\Pi$ and $\Pi'$, and we have an $\alpha$-approximation algorithm for problem $\Pi'$, then we can obtain a solution for instance $I$ of $\Pi$. We do so by generating the instance $I'$ in polynomial time and then using the $\alpha$-approximation algorithm to get a solution of value $V' \leq \alpha\text{OPT}(I')$. Finally we compute in polynomial time a solution to $I$ with value $V$ such that

$$V \leq OPT(I) - b(OPT(I') - V') \leq OPT(I) - abOPT(I) + ab\alpha OPT(I) = OPT(I)(1 - ab + ab\alpha)$$

Inspired by Bhattacharyya's proof of NP hardness of the unconstrained TTP, we build on this result to show the problem is also APX hard. In this chapter we present an $L$ reduction from $(1,2)$-TSP to TTP. To reach the desired result, we show that given an instance of TSP with a solution of cost $K$, we can construct an instance of TTP with a solution of cost at most $20m(m+1)cK$ where $m = c(n-1)+1$, $n$ is the number of teams, and $c > 5, c \in \mathbb{Z}$ is fixed. On the other hand, we show that given a feasible schedule to the constructed TTP instance, we recover a tour on the original TSP instance.

## 2.2 Reduction from (1,2)-TSP to UTTP

Given a complete graph $G$ on $n$ vertices, we want to construct a graph $G'$ such that the optimal TSP solution on $G$ has cost $K$ if and only if the optimal TSP solution on $G'$ has cost $cK$ for some choice of $c > 5$. To construct $G'$, begin by fixing a vertex $v$ of $G$. Consider $c$ copies of $G$, where all copies of $v$ are contracted into one vertex. This construction has $c(n-1)+1$ vertices. The cost of an edge between vertices in the same copy remains the same as it was in $G$, while the cost of an edge $e = (x, y)$ between vertices in different copies is determined by the cost of going from one endpoint to $v$ and then to the other endpoint i.e. $w'(e) = w(x, v) + w(v, y)$. We present an example of this construction in Figure 2.1 where we take $v = v_1$ as the central vertex and use an illegal choice of $c = 3$ just to keep the example small. We also do not include all the edges of the graph and instead record the full distance matrix in $D$.

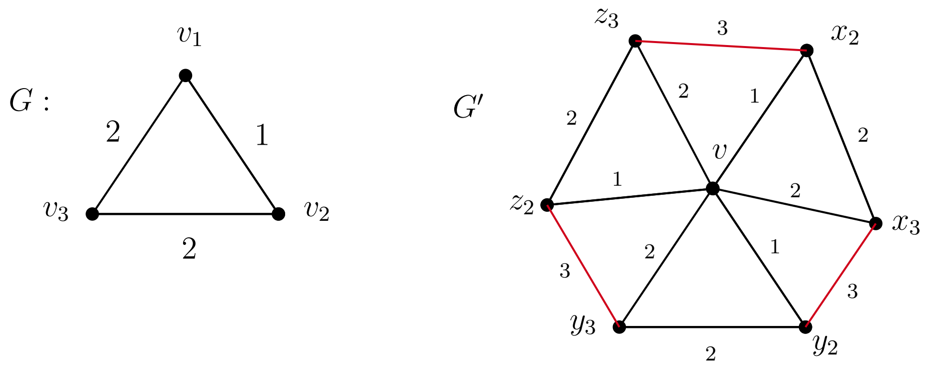


Figure 2.1: Small example: $G'$ from $G$ with $c = 3$.

$$D = \begin{array}{c} \\ v \\ x_2 \\ x_3 \\ y_2 \\ y_3 \\ z_2 \\ z_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} v & x_2 & x_3 & y_2 & y_3 & z_2 & z_3 \end{array} \\ \begin{bmatrix} - & 1 & 2 & 1 & 2 & 1 & 2 \\ 1 & - & 2 & 2 & 3 & 2 & 3 \\ 2 & 2 & - & 3 & 4 & 3 & 4 \\ 1 & 2 & 3 & - & 2 & 2 & 3 \\ 2 & 3 & 4 & 2 & - & 3 & 4 \\ 1 & 2 & 3 & 2 & 3 & - & 2 \\ 2 & 3 & 4 & 3 & 4 & 2 & - \end{bmatrix} \end{array}$$

Now let's show that the graph $G'$ is such that the optimal TSP solution on $G$ has cost $K$ if and only if the optimal TSP solution on $G'$ has cost $cK$. Let $\tau$ be a TSP tour on $G$. In $G'$, we start from $v$, and travel $\tau$ on a copy of $G$, but instead of completing the tour and returning to $v$, we move to the next copy of $G$ in $G'$. By construction, moving from one copy to the next directly, costs the same as returning to $v$ and then starting $\tau$ anew on this copy. After having travelled across each copy in this way, we have completed a salesman tour of $G'$ and incurred a cost of $cK$ since we had to complete $\tau$ $c$ times. The other direction of the claim is clear: due to the metric condition, going from one copy of $G$ to another is as costly as returning to $v$ and then moving to the next copy thus it is always best to visit one copy of $G$ at a time. Therefore if there is a TSP tour on $G'$ with cost $cK$, since each copy of $G$ (in $G'$) has equal weight we have a TSP tour on $G$ of cost $K$.

To show the reduction from $(1,2)$-TSP to TTP, we show that given an instance of $(1,2)$-TSP with cost $K$, we can construct an instance of TTP with cost at most $f(K)$, and vice-versa.

Given an instance of $(1,2)$-TSP on graph $G$, with cost $K$, we construct $G'$ as described earlier where $G'$ has $m = c(n-1)+1$ vertices. Construct a new graph $H$ by adding vertex $u$ to $G'$, and connecting $u$ to all vertices of $G'$ with an edge of weight $w_u = \frac{cK-1}{2}$. We construct the corresponding TTP instance on $H$ by placing 2 teams at $v$ (the central vertex of $G'$), one team at every other vertex of $G'$, and $(m+1)(10m-1)$ teams at $u$ to get $n' = 10m(m+1)$ teams in total.

**Lemma 1.** *If the input graph $G$ has a TSP solution of cost $K$, then $H$ has a feasible TTP solution of cost at most $20m(m+1)cK$.*

*Proof.* Let $\tau$ be a tour on $G$ of cost $K$, then $G'$ has a tour $\tau'$ of cost $cK$.

We begin by splitting up the teams into $10m$ groups of size $m+1$. All teams corresponding to vertices in $G'$ are in the same group, which we denote as group 1. We build the schedule such that every team plays $2m$ games against the other teams in their group, $m$ home games and $m$ away games. They then play $2(m+1)(10m-1)$ games against teams from different groups. Thus in the first $2m$ rounds of the schedule, games only occur within the same groups.

The total travel cost of these $2m$ rounds can be bounded. Note that games between all other groups besides group 1 carry a traveling cost of zero since they are all located at $u$. For a team in group 1, the max distance it will travel in these rounds is $8m$ since an edge in $G'$ has weight at most 4, and it must visit $m$ teams. Therefore the total distance travelled by all teams in the first $2m$ rounds is at most $8m(m+1)$.

For the remaining rounds of the schedule, identify each group as a single team and create a single-round robin "dummy tournament" where a game between dummy teams corresponds to $m+1$ games between members of each group (see Figure 2.2 for an illustration of these underlying games). The home-away assignment of the dummy teams apply to the underlying teams within the group i.e. if dummy team $i$ is at home and receives

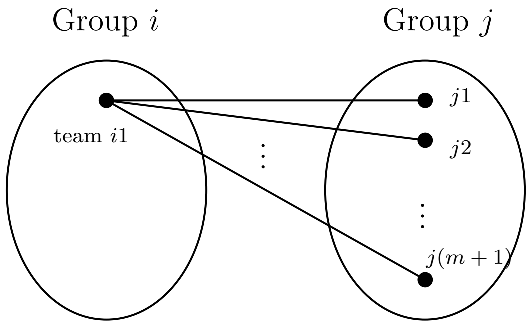


Figure 2.2: The $(m + 1)$ games team 1 of group $j$ will play in one round of the dummy tournament when dummy teams $i$ and $j$ play each other.

dummy team $j$, then all teams in group $i$ are at home for their games against teams in group $j$.

We assign all away games for dummy team 1 (which is the group of teams located in $G'$). This stage of the schedule has $10m - 1$ rounds which means each team in group 1 plays $(10m - 1)(m + 1)$ away games at $u$ in this stage. The cost of travelling from any vertex in $G'$ to $u$ is $w_u$. For all teams in group 1 to visit $u$, the total traveling cost incurred is $(m + 1)w_u$. Once they are at $u$, they play all teams in group $i$, then all teams in the following group, while remaining at $u$. When the teams of group 1 return home, they again collectively travel a distance of $(m + 1)w_u$. All other groups are at $u$ so no distance is travelled in this stage of the schedule. It follows that the total distance travelled in this stage of the schedule is $2(m + 1)w_u$.

To complete the schedule, we duplicate the single-round robin tournament of the previous stage, swapping the home-away assignments. Now all the teams at $u$ must travel to $G'$. Once they are at $G'$, they play teams in group 1 in the order of the tour $\tau'$. Traveling to $G'$ means covering a distance of $(m + 1)w_u$ per group, at which point they embark on the TSP tour of $G'$, $\tau'$, which has a cost of $cK$. Upon completing the tour, they return home to $u$. Note that each team does not travel the entire length of $\tau'$, since they return to $u$ once they have visited each team thus they skip one edge of $G'$. The edge that is skipped is different for each team in a group thus for a whole group, one entire length of $\tau'$ is economized. The total distance travelled by a single group in this stage is at most $2(m + 1)w_u + (m + 1)cK - cK = 2(m + 1)w_u + mcK$.

Having completed all these rounds, we have a feasible schedule which yields a feasible solution to TTP on $H$.

The cost of this solution is:

$$\begin{aligned}
&\leq (10m-1)(2w_u(m+1)+mcK)+2w_u(m+1)+8m(m+1)\\
&= 20w_u m(m+1)+(10m-1)mcK+8m(m+1)\\
&< 20w_u m(m+1)+10m^2cK+8m(m+1)\\
&< 20w_u m(m+1)+10m(m+1)cK+8m(m+1)\\
&< 20w_u m(m+1)+10m(m+1)cK+10m(m+1)\\
&= 20w_u m(m+1)+10m(m+1)(cK+1)\\
&= 20\left(\frac{cK-1}{2}\right)m(m+1)+10m(m+1)(cK+1)\\
&= 10m(m+1)cK-10m(m+1)+10m(m+1)cK+10m(m+1)\\
&= 20m(m+1)cK
\end{aligned}$$

This is the bound that was claimed hence the lemma is proved. □

**Lemma 2.** *If there is no TSP tour on $G$ of cost at most $K$ then the corresponding TTP instance on $H$ has no schedule with cost at most $20m(m+1)cK$.*

*Proof.* For any feasible TTP solutions, the distance traveled by any team is at least the cost of a TSP tour on the cities. Suppose the optimal tour on $G$ has cost $K+\alpha, \alpha > 1$. We know that $G'$ has a TSP tour of cost $c(K+\alpha)$. Now, since any edge of $G'$ has weight at most 4, and joining any vertex of $G'$ to $u$ has cost $w_u$, the TSP tour on $H$ has cost at least $2w_u + c(K+\alpha) - 4$. The cost of the tournament schedule on $H$ is therefore:

$$\begin{aligned}
&\geq 10m(m+1)(2w_u+c(K+\alpha)-4)\\
&= 20w_u m(m+1)+10m(m+1)c(K+\alpha)-40m(m+1)\\
&= 20w_u m(m+1)+m(m+1)(10cK+10c\alpha-40)\\
&> 20w_u m(m+1)+10m(m+1)(cK+1)\\
&= 20m(m+1)cK
\end{aligned}$$

We use the fact that $c > 5, \alpha \geq 1$ and replace $w_u$ by $\frac{cK-1}{2}$ to conclude the claim. □

## 2.3 APX Hardness of Boosted TSP

Let $G$ be a complete graph with metric costs on $m$ nodes, and let $m(m+1)$-TSP be the problem of computing a TSP tour $T$ such that the objective $(m(m+1))(\text{cost}(T))$ is minimized. This new problem will be referred to as *boosted TSP*.

**Claim 1.** *Boosted TSP is APX hard.*

*Proof.* Given some input graph $G$ on $m$ nodes, let $OPT_B$ and $OPT$ be the optimal values of the boosted TSP and regular TSP respectively, on this instance. Suppose there is a polynomial time approximation scheme for boosted TSP, then for any $\varepsilon$ we can find a solution $S$ in polynomial time such that $m(m+1) \cdot \text{cost}(S)$ is at most $(1+\varepsilon) \cdot OPT_B = (1+\varepsilon)(m(m+1)) \cdot OPT$. Therefore, $\text{cost}(S)$ is at most $(1+\varepsilon) \cdot OPT$. Moreover, since $S$ is a TSP tour on $G$, it is a feasible solution to the TSP. It follows that if boosted TSP has a PTAS, then for every $\varepsilon$, we can find a TSP tour $S$ such that $\text{cost}(S)$ is within $(1+\varepsilon)$ of OPT. Since TSP is known to be APX hard, we conclude that boosted TSP is as well. $\square$

## 2.4 APX Hardness of UTTP

In this section we make use of the results of Section 2.2 to show that unconstrained TTP is APX-hard using an $L$-reduction from $(1,2)$-TSP.

Let $I$ be a boosted TSP instance on graph $G$ with input integers $c$. Let $I'$ be the corresponding TTP instance constructed as in Section 2.2.

To get a valid $L$-reduction we need to verify three conditions:

1. For each instance $I$ of boosted TSP, we can compute an instance $I'$ of TTP in polynomial time.
2. $OPT(I') \leq a \cdot OPT(I)$
3. Given a solution of value $V'$ to $I'$, we can compute in polynomial time a solution to $I$ of cost $V$ such that

$$|OPT(I) - V| \leq b|OPT(I') - V'|$$

For condition 1, it suffices to use the reduction in Section 2.2 where the boosted TSP instance is the TSP instance on $G$ with its objective multiplied by $m(m+1)$. Since we are not given a TSP tour on $G$, we take $K = 2|G| = 2n$ which is an upper bound on the cost of any tour on $G$ due to the $(1,2)$ metric. Taking this value of $K$ allows us to use the construction described in Section 2.2, and preserves the upper bound on the constructed TTP solution.

For condition 2, we can compute $a$ directly. Suppose the instance of boosted TSP on the graph $G$ has optimal solution $m(m+1)K^*$, note that clearly $K^* \geq n$. Then due to Lemma 1 and our choice of $w_u = \frac{2cn-1}{2}$, the cost of the optimal solution to our constructed TTP instance is bounded.

$$\begin{aligned} OPT(I') &\leq 20w_u m(m+1) + 10m(m+1)(2cn+1) \\ &= 10(2cn-1)m(m+1) + 10m(m+1)(2cn+1) \\ &= 40m(m+1)cn \\ &\leq 40m(m+1)cK^* \\ &= 40c \cdot OPT(I) \end{aligned}$$

To show condition 3 holds we show that given a solution to our TTP instance, we recover a solution to the TSP instance on $G$. If the TTP solution is optimal, then the TSP solution is as well. On the other hand, if the TTP solution has some slack from the optimal, then the TSP solution constructed is at most 10 times as far from the optimal tour on $G$ which would satisfy condition 3 with $b = 10$.

All teams at $u$ visit all teams at $G'$. Let $u_i$ be the team for which its cost of visiting teams at $G'$ is minimal. Over the course of possibly multiple trips, $u_i$ visits each team at $G'$. The order in which teams are visited creates a sequence of vertices which include all vertices in $G'$. Let $P$ be the path defined by this sequence of vertices, since it covers all $v \in G'$, $P$ is a TSP path on $G'$. By our choice of the edge costs for $G'$, we know that going from one copy of $G$ to another is as costly as returning to the central vertex $v$, and then going to the next copy. This means that the TSP path $P$ can be mapped to a walk $P'$ by replacing edges $(x, y)$ of $P$ that cross copies by the edges $(x, v)$, $(v, y)$. Thus $P$ induces a tour on each copy of $G$, with the exception of the copies at which the path starts and ends. Let $S$ be the cheapest induced tour on a copy of $G$, and let $V$ be its cost.

**Claim 2.** *Given an optimal solution to TTP of cost $V'$, let $S$ be the cheapest induced TSP tour on $G$ with cost $V$. We claim that $S$ is optimal for the TSP instance on $G$.*

*Proof.* We proceed by contradiction. We have that $V'$ is the cost of our optimal TTP solution, and $V$ is the cost of the constructed TSP solution. Since the TSP solution is not optimal, let $S^*$ be an optimal TSP tour on $G$. We use $S^*$ to construct a TTP solution as in Section 2.2, and the cost its solution is denoted $V_{S^*}$. We bound this cost by Lemma 1.

$$V_{S^*} \leq 20w_u m(m+1) + (10m-1)m(cS^*) + 8m(m+1)$$

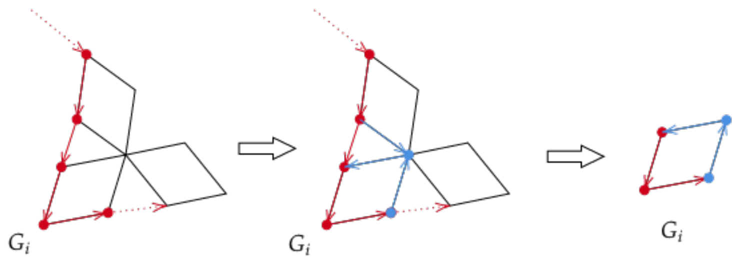


Figure 2.3: The leftmost graph shows $P$ (in red) going from one copy to $G_i$. The next graph shows the equivalent edges (blue) going back and forth from the central vertex. Finally we see the induced tour on a copy $G_i$.

For our current TTP solution $V'$, we have:

$$V' \geq 20w_u m(m+1) + (10m-1)(m+1)(cV-4)$$

We can then compare the cost of the two TTP solutions. After eliminating the identical first term in both expressions we have:

$$\begin{aligned} V' - V_{S^*} &\geq (10m-1)(m+1)(cV-4) - (10m-1)m(cS^*) - 8m(m+1) \\ &= (10m-1)m(cV - cS^* - 4) + (10m-1)(cV-4) - 8m(m+1) \\ &\geq (10m-1)m + 10m - 1 - 8m^2 - 8m \\ &= 2m^2 + m - 1 \end{aligned}$$

To reach the final inequality we use the fact that since $S$ is not optimal, $V$ differs from $S^*$ by at least 1, and that $c \geq 5$, therefore $c(V - S^*) \geq 5$. We remark that $2m^2 + m - 1 > 0$ for all $m \geq 1$, which is trivially true for any instance. Thus the cost of our solution $V'$ greater than the cost of the solution $V_{S^*}$ which we constructed using the optimal TSP tour $S^*$. This contradicts the optimality of $V'$ since we have just constructed a cheaper solution to our TTP instance. We conclude that $S$ must be optimal for the TSP instance on $G$. □

**Claim 3.** *Given a non-optimal solution to TTP of cost $V'$, let $S$ be the cheapest induced TSP tour on $G$ with cost $V$. We claim that condition 3 holds for $V$ and $V'$.*

*Proof.* We assume for a contradiction that the inequality in condition 3 does not hold for $b = 10$. We have that $V'$ is a non-optimal solution to the TTP instance $I'$, and $V$ is the cost of the constructed TSP solution. If $V$ is optimal, the inequality of condition 3 is trivially true.

$$b \cdot |OPT(I') - V'| \geq 0 = |OPT(I) - V|$$

We now assume $V$ is not optimal for the TSP instance $I$. Let $S^*$ be an optimal TSP tour on $G$. As in the proof of Claim 2, we use $S^*$ to construct a new TTP solution, denoted $V_{S^*}$. We also re-use the same bounds on $V'$ and $V_{S^*}$:

$$\begin{aligned} V_{S^*} &\leq 20w_u m(m+1) + (10m-1)m(cS^*) + 8m(m+1) \\ V' &\geq 20w_u m(m+1) + (10m-1)(m+1)(cV-4) \end{aligned}$$

We can then compare the cost of the two TTP solutions. After eliminating the identical first term in both expressions we have:

$$\begin{aligned} V' - V_{S^*} &\geq (10m-1)(m+1)(cV-4) - (10m-1)m(cS^*) - 8m(m+1) \\ &= (10m-1)m(cV - cS^* - 4) + (10m-1)(cV-4) - 8m(m+1) \\ &\geq (10m-1)m + 10m - 1 - 8m^2 - 8m \\ &= 2m^2 + m - 1 \\ &> 0 \quad \forall m \geq 1 \end{aligned}$$

Therefore $V' > V_{S^*}$. Moreover, since $OPT(I') \leq V_{S^*}$, the gap between $V'$ and $OPT$ is always at least the gap between $V'$ and $V_{S^*}$ i.e. $|OPT(I') - V'| \geq |V' - V_{S^*}|$. We then have:

$$\begin{aligned} |OPT(I') - V'| &\geq |V_{S^*} - V'| \\ &\geq |(10m-1)(m+1)(cV-4) - (10m-1)m(cS^*) - 8m(m+1)| \\ &\geq |(10m-1)m(c(V - S^*) - 4) + (10m-1)(cV-4) - 8m(m+1)| \\ &\geq (10m-1)m(c|V - S^*| - 4) + (10m-1)(cV-4) - 8m(m+1) \end{aligned}$$

Now we make use of our assumption that condition 3 does not hold i.e. $|S^* - V| = |OPT(I) - V| > b \cdot |OPT(I') - V'|$. Using $b = 10$ yields the following:

$$\begin{aligned} |OPT(I') - V'| &\geq (10m-1)m(c|V - S^*| - 4) + (10m-1)(cV-4) - 8m(m+1) \\ &> (10m-1)m(10c|OPT(I') - V'| - 4) + (10m-1)(cV-4) - 8m(m+1) \end{aligned}$$

Using the fact that $c \geq 5$, it holds that $cV - 4 \geq 1$. Moreover, using this bound on $c$, and the non-optimality of $V'$ we have:

$$\begin{aligned} c \cdot b|OPT(I') - V'| - 4 &\geq 4b \cdot |OPT(I') - V'| - 4 + b \cdot |OPT(I') - V'| \\ &\geq b \cdot |OPT(I') - V'| \end{aligned}$$

Armed with these two facts, we revisit the final inequality in the previous block.

$$\begin{aligned}
|OPT(I') - V'| &> (10m-1)m(10c|OPT(I') - V'| - 4) + (10m-1)(cV-4) - 8m(m+1) \\
&\geq 10(10m-1)m \cdot |OPT(I') - V'| + (10m-1) - 8m(m+1) \\
&= 10(10m-1)m \cdot |OPT(I') - V'| - 8m^2 + 2m - 1 \\
&\geq (10m-1)m \cdot |OPT(I') - V'| + 9(10m-1)m - 8m^2 + 2m - 1 \\
&= (10m-1)m \cdot |OPT(I') - V'| + 82m^2 - 7m - 1 \\
&> (10m-1)m \cdot |OPT(I') - V'| \\
&> |OPT(I') - V'|
\end{aligned}$$

We have reached a contradiction for $b = 10$, thus we conclude that condition 3 indeed holds. □

Having verified all conditions, we have an $L$-reduction from boosted TSP to TTP, and hence TTP is APX-hard.

# Chapter 3

# Mirrored TTP

## 3.1 Overview

In this chapter we consider the mirrored Traveling Tournament problem (mTTP). The problem is defined and we discuss common methods of constructing mirrored schedules. These methods rely on our ability to find good single round-robin tournaments hence the relationship between a single round-robin (SRR) tournament and a double round-robin (DRR) tournament is examined. We propose several new results which compare the optimal value of a SRR schedule to the optimal value of a DRR schedule, as well as the value of an optimal SRR schedule compared to that of an optimal mirrored DRR schedule. Finally, we introduce the first constructive algorithm for building a mirrored DRR schedule when the number of consecutive home or away games is restricted to two. We prove that the approximation guarantee of the algorithm is on the order of $3/2 + O(1)/n$.

## 3.2 The Problem

A widely studied variation of the Travelling Tournament problem is the mirrored version, first described by Ribeiro and Urrutia [27]. This problem is described identically to the TTP but with the additional constraint that the schedule of the first half of the season is the same as that of the second half of the season, but with the venues reversed. If team $A$ plays at away at team $B$'s venue on the first day of the first half of the season, then team $B$ plays away team $A$'s venue on the first day of the second half of the season. Table

| Team | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Day 1 | @2 | 1 | @4 | 3 |
| Day 2 | @3 | 4 | 1 | @2 |
| Day 3 | @4 | @3 | 2 | 1 |
| Day 4 | 2 | @1 | 4 | @3 |
| Day 5 | 3 | @4 | @1 | 2 |
| Day 6 | 4 | 3 | @2 | @1 |

Table 3.1: Mirrored DRR schedule on 4 teams.

3.1 shows an example of a feasible mirrored double round-robin schedule on 4 teams. The table is split into two halves to demonstrate the mirrored schedule.

As in the regular TTP, we have a parameter $k$ which is the maximum number of consecutive home or away games a team can play. We use the notation mTTP($k$) to specify mirrored TTP($k$). Due to the way the problem was introduced, mTTP(3) remains the most popular version and has seen the most progress.

Given a SRR schedule $S$, its mirror $m(S)$ is another SRR schedule with the same order of games but opposite home-away assignments. Concatenating the two SRR schedules yields a DRR schedule that is mirrored. This approach remains the most common and intuitive way of building a DRR schedule, mirrored or not. It becomes clear that any method for constructing a mirrored schedule is only as good as the SRR schedule used to create it. The relationship between an SRR schedule and its mirror, as well as the relationship between an SRR schedule and a DRR schedule (mirrored or not) has yet to be discussed thoroughly in the literature. We explore the relationship between single and double round-robin schedules in Section 3.5 and further explore the relationship between a SRR schedule and an optimal mirrored DRR schedule in Section 3.6.

When focusing on mTTP, the challenge is to construct a SRR schedule that can indeed be mirrored to create a mirrored DRR schedule. Note that when concatenating a SRR schedule and its mirror schedule, even if each of these schedules has at most $k$ consecutive home games (or $k$ consecutive away games), once they are concatenated, a team may have have more than $k$ consecutive home games (or $k$ consecutive away games). Feasibility therefore cannot be taken for granted for all values of $k$. The following result illustrates this.

There is no feasible schedule for mTTP(2) when $n = 4$. See Theorem 1 and its proof in Section 3.4.

Incidentally, several key results for the mirrored TTP were not intended to specifically solve the mirrored problem. The fact that the resulting schedule is mirrored is merely a byproduct of the method used to construct it.

## 3.3 Literature Review

In this section we discuss some papers that give key results for the mirrored TTP. Not all of these papers actually solved the mirrored problem but they each have some method that proved to be helpful for us.

The mirrored version of the problem was first introduced by Ribeiro and Urrutia [27]. They studied this version of the problem due to it being a common tournament structure in Latin America. A SRR tournament is constructed without assigning venues to games. This SRR tournament is then mirrored to get a full DRR tournament. A local search method is implemented to ensure the resulting DRR schedule satisfies the at-most constraint, and the schedule is then optimized to reduce the traveling cost.

Yamaguchi, Imahori, Miyashiro, and Matsui [35] propose an approximation algorithm to solve TTP($k$) for $k \geq 3$. They build a mirrored schedule by using a Kirkman schedule to generate an initial SRR schedule without giving teams a home-away assignment. They then define particular home-away assignments which allow them to concatenate a SRR schedule with its mirror to produce a feasible mirrored DRR schedule. When $k \leq 5$, the approximation ratio of the proposed algorithm is bounded by $(2k-1)/k + O(k/n)$, when $k > 5$, the ratio is bounded by $(5k-7)/(2k) + O(k/n)$. For $k = 3$, the approximation ratio of the proposed algorithm is $5/3 + O(1)/n$.

Imahori, Matsui, and Miyashiro [19] build a DRR schedule that happens to be mirrored. They solve the unconstrained TTP (i.e. $k = n-1$). They propose an approximation algorithm which has a ratio of 2.75. The method used to build the schedule relies on the fact that the problem being solved is unconstrained.

Thielen and Westphal [30] proposed an algorithm for constructing a DRR schedule for TTP(2) which gave an approximation ratio of $3/2 + O(1)/n$. Their method paved the way for building mirrored schedules even though the algorithm they proposed does not produce a mirrored solution. They begin by building a SRR schedule according to the canonical tournament by De Werra [10], also known as the circle method. They then create a DRR schedule where the second half of the season has the same order of games as the initial SRR schedule but with opposite venues. However, the concatenation of the two SRR schedules can create some issues. At the point where the two halves of the schedules meet, it is possible for a team to play more than two consecutive home or away games. To get around this, the second half of the schedule starts with the games of the last two days of the first half. This ensures that no team plays more than two home or away games in a row when connecting the two halves of the schedule. The same approach was then used by Westphal and Noparlik [33] to generate solutions for TTP($k$) when $k \geq 4$ and $n \geq 6$, albeit using a different lower bound to analyze the cost. Their analysis results in a 5.875-approximation algorithm.

Imahori [18] took this approach one step further. The method leans heavily on the work done by Thielen and Westphal: a perfect matching is computed and used as a lower bound, and the circle method is used to generate a SRR schedule. However, in this scheme, teams are paired up and these pairs are then matched together to schedule games between them. A match between two pairs of teams corresponds to four games for each of the four teams involved over two rounds. For example given pairs of teams $(A, B)$ and $(C, D)$, the games in the first round are between $A$ and $C$, and between $B$ and $D$. The games in the second round are between $A$ and $D$, and between $B$ and $C$.

## 3.4 Feasibility of mTTP(2)

**Theorem 1.** *There is no feasible schedule for mTTP(2) when* $n = 4$.

To demonstrate this result we present two lemmas covering both possible cases. Either the first and last day have the same HA assignment or not. The following lemmas show that each case results in an infeasible schedule violating the at-most constraint of $k = 2$.

**Lemma 3.** *If the home-away assignment of Day 1 is the same as Day* $n - 1$*, there is no feasible mirrored schedule for* $n = 4$.

*Proof.* Consider the table 3.2 depicting the first half of a schedule where days 1 and $n-1$ have the same home-away assignment. The particular choice of assignment is made without loss of generality. Columns correspond to teams. On day 1, team 1 plays team 3 or 4,

| | | | | |
|---|---|---|---|---|
| Day 1: | H | H | A | A |
| Day 2: | - | - | - | - |
| Day 3: | H | H | A | A |

Table 3.2: Home-away assignments of first and last day

while team 2 plays whichever team is is not being played by 1. On day $n-1$, teams 1 and

2 will switch and make the opposite choice. So if on day 1, team 1 played team 3, then that means team 2 played team 4. It follows that on day $n-1$, team 1 plays team 4, and team 2 plays team 3. However, since all teams must play each other once in one half of the schedule, teams 1 and 2 must play each other on day 2. This is impossible since they must both be away on day 2 to satisfy the at-most constraint. □

**Lemma 4.** *If the home-away assignment of Day 1 is the opposite as the assignment of Day $n-1$, there is no feasible mirrored schedule for $n = 4$.*

*Proof.* Consider the table 3.3 depicting the first half of a schedule. The particular choice of assignment is made without loss of generality. Columns correspond to teams. This time, days 1 and $n-1$ have different same home-away assignment. We have two cases. In the

| | | | | |
|---|---|---|---|---|
| Day 1: | H | H | A | A |
| Day 2: | - | - | - | - |
| Day 3: | H | A | A | H |
| Day 4: | A | A | H | H |
| Day 5: | - | - | - | - |
| Day 6: | A | H | H | A |

Table 3.3: Case 1: 2 teams have different home-away assignments

first case, the home-away assignments of day 1 and day $n-1$ differ in just two teams. Since day 2 and day 5 have opposite home-away assignments, if team 2 is home on day 2, then it is away on day 5 which creates a streak of three away games in a row for team 2. On the other hand, if team 4 is away on day 2, then it is home on day 5 which creates a streak of three home games in a row for team 4. Thus in this first case, it is impossible to create a feasible mirrored schedule for $n = 4$. In the second case,the home-away assignments of

| | | | | |
|---|---|---|---|---|
| Day 1: | H | H | A | A |
| Day 2: | - | - | - | - |
| Day 3: | A | A | H | H |
| Day 4: | A | A | H | H |
| Day 5: | - | - | - | - |
| Day 6: | H | H | A | A |

Table 3.4: Case 2: 4 teams have different home-away assignments

day 1 and day $n-1$ are different for all four teams. Since day 2 and day 5 have opposite home-away assignments, if team 2 is home on day 2, then it is away on day 5 which creates a streak of three away games in a row for team 2. On the other hand, if team 4 is away on day 2, then it is home on day 5 which creates a streak of three home games in a row for team 4. Thus in the second case as well, it is impossible to create a feasible mirrored schedule for $n = 4$. $\square$

## 3.5 SRR vs. DRR Tournaments

We are interested in the relationship between the cost of a single round robin schedule, and the cost of a double round robin schedule, each subject to the at-most constraint being $k$ i.e. a team can play at most $k$ consecutive home or $k$ consecutive away games. Let $SRR_k$ and $DRR_k$ denote the SRR and DRR schedules respectively. We construct an example to find a lower bound on the ratio of their costs:

$$\frac{c(DRR_k)}{c(SRR_k)}$$

Let $n$ be the number of teams, with $n-1$ teams located at the same point and thus at distance 0 from each other. Place the last team, $w$, at a distance $L$ from the other teams. The construction is illustrated below in Figure 3.1.

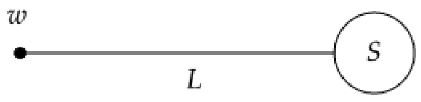


Figure 3.1: Illustration of the graph construction.

Let us now compute the cost of a SRR schedule on this instance. To minimize the total cost, $w$ visits $S$, visits $k$ teams and returns home. Our standing assumption is that $n-1$ is divisible by $k+1$. Using this assumption, we partition $S$ into groups of size $k+1$. On each trip performed by $w$, it visits $k$ teams in a group, then returns home and is visited by the remaining unvisited team in the group. The total cost per group is then $4L$: $2L$ for the roundtrip by $w$, and another $2L$ for the roundtrip done by the last team in the group.

The total cost of this schedule is:

$$c(SRR_k) = \left(\frac{n-1}{k+1}\right) \cdot 4L$$

With a slight modification to the construction of a SRR schedule shown by Westphal and Noparlik we can get a complete feasible schedule that attains this cost. We make it so that team $w$ plays $k$ away games, then just a single home game before continuing with another $k$ away games.

To evaluate the cost of a DRR schedule, we remark that since all teams at $S$ must visit $w$, the cost is at least $(n-1) \cdot 2L$. Therefore,

$$\frac{c(DRR_k)}{c(SRR_k)} \geq \frac{k+1}{2}$$

We can make this ratio arbitrarily large for large values of $k$.

## 3.6 SRR Schedule vs mirrored DRR Schedule

For any SRR schedule $S$, let $m(S)$ denote the mirrored SRR of $S$ meaning it is a SRR schedule with the same games each day but opposite HA assignments. Let $\mathrm{drr}(S)$ denote the DRR schedule we get by concatenating $S$ and $m(S)$. The cost of the concatenated schedule is computed by finding the cost of the resulting DRR schedule, not by adding the costs of its constituent parts.

The following results hold for any value of $k$.

**Claim 4.** *In the unit distance case, where all teams are at unit distance from each other, the cost of a SRR schedule does not change when the HA assignments are flipped.*

*Proof.* Let $p_i$ denote the number of *returning* teams on day $i$, and let $u_i$ denote the number of *embarking* teams on day $i$. A returning team is one that played away the previous day, and is now playing at home. An embarking team is one that played at home the previous day and is now playing away. On day $i$, to count the number of away teams, we can take the number of teams that were away the previous day, minus the number of teams that returned home for day $i$, plus the number of teams that are embarking on a trip on day $i$. Thus the number of away teams on day $i$ is $n/2 - p_i + u_i$. Since we know that the number of away teams is $n/2$ every day, we conclude that $p_i = u_i$ for all $i \in \{2, \ldots, n-1\}$.

For a SRR schedule $S$ with unit distances, the total cost of the schedule is the cost of $n/2$ teams playing away each day plus the cost of teams that return home from an away trip. Thus we have:

$$c(S) = n \cdot (n/2) + \sum_{i=2}^{n-1} p_i$$

Let $m(S)$ be the mirror of $S$, meaning it is a SRR schedule with the same games each day but opposite HA assignments. It is clear that $p_i$ in $S$ is equivalent to $u_i$ in $m(S)$ since whenever a team is embarking on a trip in $S$, it is returning from a trip in $m(S)$.

$$c(m(S)) = n \cdot (n/2) + \sum_{i=2}^{n-1} u_i$$

Since we have shown that $p_i = u_i$, we conclude that $c(S) = c(m(S))$ as desired. □

**Claim 5.** *In the $(1, 2)$-metric, where all teams are at distance $1$ or $2$ from each other, the cost of a mirrored SRR schedule is at most twice the cost of an optimal SRR schedule.*

*Proof.* Let $S$ be an SRR schedule on $n$ teams. Let $m(S)$ be its mirror. We have the following lower bound on the cost of an SRR schedule:

$$c(S) \geq n \cdot \frac{n}{2} + \sum_{i=2}^{n-1} p_i$$

We also have an upper bound on the cost of its mirror $m(S)$:

$$\begin{aligned} c(m(S)) &\leq 2 \cdot n \cdot \frac{n}{2} + 2 \cdot \sum_{i=2}^{n-1} u_i \\ &= 2 \cdot \frac{n^2}{2} + 2 \cdot \sum_{i=2}^{n-1} p_i \\ &\leq 2 \cdot c(S) \end{aligned}$$

□

Claim 5 can be extended to any metric and we get:

$$c(m(S)) \leq h \cdot c(S)$$

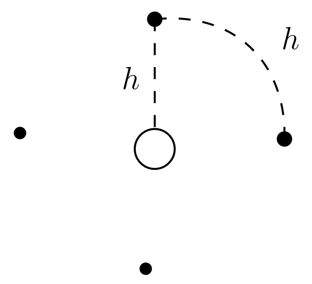


Figure 3.2: Configuration with $n/2$ inner teams in the center and $n/2$ outer teams in a ring surrounding them.

where $h$ is the cost of the heaviest edge i.e. the largest distance between any pair of teams. Moreover, we can use the result to get a bound on the cost of an optimal mirrored TTP solution. Concatenating $S$ and its mirror $m(S)$ yields a mirrored DRR schedule which we denote by $\mathrm{drr}(S)$. When the triangle inequality holds on the edge costs, the concatenated DRR has cost at most the sum of the costs of $S$ and $m(S)$.

$$\mathrm{drr}(S) \leq c(S) + c(m(S))$$

This is because if a team in $S$ plays away on the last day of the schedule, and also plays away on the first day of $m(S)$, instead of returning home and then embarking anew for their first game of $m(S)$, they travel directly from one venue to the next, thus short-cutting their trip. Therefore:

$$OPT(mTTP) \leq c(S^*) + c(m(S^*)) \leq (h+1) \cdot c(S^*)$$

**Theorem 2.** *Given an SRR schedule $S$ on $n$ teams where the furthest distance between two teams is $h$, we can bound the cost of the mirror SRR schedule m(S).*

$$c(m(S)) \leq c(S) + \Theta(n^2) \cdot h$$

*Proof.* We begin by showing that the additive term cannot be linear by constructing an instance where the cost of the mirrored SRR schedule is $c(S) + \Omega(n^2) \cdot h$.

Let $n$ be the number of teams where $n/2$ is even. We place $n/2$ teams together at distance zero from each other, and then place the remaining $n/2$ teams in a circle around the centered teams, such that the outer teams are at a distance of $h$ from all other teams. See Figure 3.2 for the configuration. We now construct a SRR schedule $S$ and bound its cost.

Let $u_i$, $i \in \{1, \ldots, n/2\}$ denote the teams on the outer circle, and let $w_j, j \in \{1, \ldots, n/2\}$ denote the teams in the inner circle. The outer teams visit the inner teams in sequence. On day 1, $u_1$ visits $w_1$ while $u_2$ visits $w_2$, and so on. On day 2, the $u_i$'s move over by one team so that $u_1$ is now visiting $w_2$, $u_2$ is now visiting $w_3$ and so on. After $n/2$ days, each inner team has been visited by each outer team. Table 3.5 shows the schedule constructed for the games between the inner and outer teams using $n = 8$.

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ |
|---|---|---|---|---|---|---|---|---|
| 1 | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $@w_1$ | $@w_2$ | $@w_3$ | $@w_4$ |
| 2 | $u_4$ | $u_1$ | $u_2$ | $u_3$ | $@w_2$ | $@w_3$ | $@w_4$ | $@w_1$ |
| 3 | $u_3$ | $u_4$ | $u_1$ | $u_2$ | $@w_3$ | $@w_4$ | $@w_1$ | $@w_2$ |
| 4 | $u_2$ | $u_3$ | $u_4$ | $u_1$ | $@w_4$ | $@w_1$ | $@w_2$ | $@w_3$ |

Table 3.5: First 4 days of the schedule with 8 teams.

The cost of the first part of the schedule has cost:

$$\frac{n}{2} \cdot h \cdot 2 = n \cdot h$$

The outer teams have not played each other yet. Over the next $(n/2-1)$ days the outer teams play against all other outer teams while all inner teams play amongst themselves. The cost of the inner teams playing themselves is zero, while the cost of the outer teams playing themselves can be found by computing the cost of an optimal SRR schedule with unit distances, and then multiplying the cost of the schedule by $h$. This is because all outer teams are at equal distance from each other thus an optimal solution to the unit distance case can just be scaled by $h$ to find the cost of an optimal SRR schedule on the outer teams. However, we must subtract $\frac{nh}{4}$ from this cost since half of the outer teams go directly from an inner team to visit an outer team thus saving one trip of cost $h$. We have a result due to [19] which gives the cost of an optimal SRR schedule with unit costs to be:

$$\left(\frac{n^2}{8} + \frac{n}{4} - 1\right) - \frac{n}{4}$$

Thus this portion of the schedule has the same cost multiplied by $h$. We therefore have the total cost of the schedule to be:

$$c(S) = nh + \left(\frac{n^2}{8} + \frac{n}{4} - 1\right) \cdot h - \frac{nh}{4}$$

|  | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ |
|---|---|---|---|---|---|---|---|---|
| 1 | $@u_1$ | $@u_2$ | $@u_3$ | $@u_4$ | $w_1$ | $w_2$ | $w_3$ | $w_4$ |
| 2 | $@u_4$ | $@u_1$ | $@u_2$ | $@u_3$ | $w_2$ | $w_3$ | $w_4$ | $w_1$ |
| 3 | $@u_3$ | $@u_4$ | $@u_1$ | $@u_2$ | $w_3$ | $w_4$ | $w_1$ | $w_2$ |
| 4 | $@u_2$ | $@u_3$ | $@u_4$ | $@u_1$ | $w_4$ | $w_1$ | $w_2$ | $w_3$ |

Table 3.6: First 4 days of the mirrored schedule with 8 teams.

In the mirrored SRR schedule, the outer teams are visited by the inner teams for the first $n/2$ days of the schedule. So on day 1, $w_1$ visits $u_1$ while $w_2$ visits $u_2$, and so on. After $n/2$ days, all inner teams have played all outer teams and the inner teams then return home. Table 3.6 shows the mirrored schedule constructed for the games between the inner and outer teams using $n = 8$.

The cost of this first part of the schedule is:

$$\left(\frac{n}{2}+1\right)\cdot h\cdot\left(\frac{n}{2}\right)$$

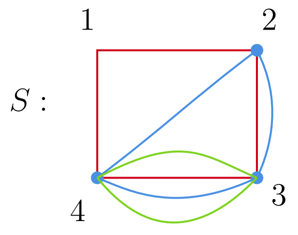
Over the next $n/2 - 1$ days the inner teams play amongst themselves while the outer teams do the same. The cost of the inner teams playing each other is zero while the cost of the outer teams playing each other is exactly the same as it was in $S$.

$$\begin{aligned}
c(m(S)) &= \left(\frac{n}{2}+1\right)\cdot h\cdot\left(\frac{n}{2}\right) + \left(\frac{n^2}{8}+\frac{n}{4}-1\right)\cdot h \\
&= \left(\frac{n^2}{8}+\frac{n}{4}-1\right)\cdot h + \frac{nh}{2} + \frac{n^2}{4}\cdot h \\
&= c(S) + \left(\frac{n^2}{4}-\frac{n}{4}\right)\cdot h \\
&= c(S) + \Omega(n^2)\cdot h
\end{aligned}$$

Now that we are resigned to a quadratic additive term, we attempt to find the tightest such bound.

First we define $\Delta = \sum_{(i,j)\in T^2} c(i,j)$ to be the sum of all distances between teams. Note that since we sum over all ordered pairs of teams, each distance is counted twice. It is easy to see that for any feasible SRR schedule $S$,

$$c(S) \leq \Delta$$



Trips taken in $S$:
$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$
$2 \rightarrow 3 \rightarrow 4 \rightarrow 2$
$3 \rightarrow 4 \rightarrow 3$

Edges used:
(1, 2), (2,3), (3,4), (4,1)
(2,3), (3,4), (4,2)
(3,4), (4,3)

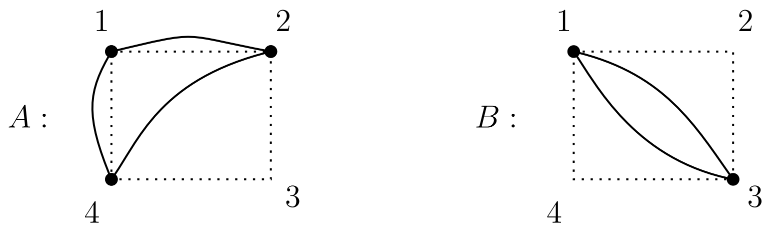


Figure 3.3: Illustration of an SRR schedule and the edges missing from $\Delta$.

We also have:

$$\Delta \leq c(S) + c(\text{missing edges})$$

Where the missing edges are those that are counted in $\Delta$ but not used in $S$. Let us further specify two kinds of missing edges. Let $A$ be the set of edges that are used in $S$ exactly once, and let $B$ be the set of edges counted in $\Delta$ that are not included in $S$. Figure 3.3 contains an example of an SRR schedule $S$ and demonstrates the partition of the edges used in $S$ into the sets $A$ and $B$.

We have that $c(A) \leq \frac{n}{2}(n-2) \cdot h$. This is due to the fact that each team is incident to at most $n-2$ edges in $A$, however, every edge will be counted twice. The cost of these edges is at most $\frac{n}{2}(n-2) \cdot h$ since $h$ is the heaviest edge.

The number of edges in $B$ is maximized when teams take longer trips thus avoiding using as many edges as possible. This is achieved when each team $i$ takes a trip that visits all teams $j > i$. In doing so, team $n$ has already been visited by all teams thus has no need to travel and has no missing edges. Team $n-1$ only travels to team $n$ and back, thus has $n-3$ edges missing. Team 1 starts its trip at team 2 and returns via team $n$ thus it also has $n-3$ missing edges. All other teams have $n-4$ missing edges since they have an edge coming in from the previous team, an edge for their embarking trip, and an edge for their return. We therefore have that $|B| \leq 2(n-3)+(n-3)(n-4)$. Thus $c(B) \leq (n^2-5n+6)\cdot h$.

Combining these bounds together we find a bound on the cost of $m(S)$.

$$\begin{aligned}
c(m(S)) \leq \Delta &\leq c(S)+c(A)+c(B) \\
&\leq c(S)+\frac{n}{2}(n-2)\cdot h+(n^2-5n+6)\cdot h \\
&= c(S)+\left(\frac{3n^2}{2}-6n+6\right)\cdot h \\
&= c(S)+O(n^2)\cdot h
\end{aligned}$$

We arrive at the claim since we have shown that the additive term is both $O(n^2)$ and $\Omega(n^2)$. □

**Corollary 1.** *Let $drr(S^*)$ denote the cost of a mirrored DRR schedule constructed by concatenating the optimal SRR schedule $S^*$ and its mirror schedule, $m(S^*)$. Then:*

$$drr(S^*) \leq OPT(mTTP)+O(n^2)\cdot h$$

*Proof.* We have already established that the concatenated DRR schedule has cost at most that of the two SRR schedules that make it up. We therefore only need to show that the inequalities hold for the cost of an optimal SRR schedule plus its mirror schedule.

The optimal mirrored TTP solution is made up of two concatenated SRR schedules each corresponding to one half of the schedule. Let $T$ be the SRR schedule corresponding to the first half of the schedule, and $m(T)$ be the SRR schedule of the second half. By optimality of $S^*$:

$$2c(S^*) \leq c(T)+c(m(T))$$

The full mirrored TTP solution is less costly due to the amount saved when concatenating the two halves. When a team plays away in its last game of $T$, and is away again in its first game of $m(T)$, we save on the cost of this team returning home at the end of $T$, and having to travel from home to start $m(T)$. Since there are $n/2$ teams that play away each day, and $h$ is the largest distance between teams, the cost saved when concatenating $T$ and $m(T)$ is at most $\frac{nh}{2}$.

$$c(T)+c(m(T)) = OPT(mTTP)+\text{“saved costs”} \leq OPT(mTTP)+\frac{nh}{2}$$

Using Theorem 2:

$$\begin{aligned}
c(S^*) + c(m(S^*)) &\leq 2c(S^*) + O(n^2) \cdot h \\
&\leq OPT(mTTP) + \text{“saved cost”} + O(n^2) \cdot h \\
&\leq OPT(mTTP) + \frac{nh}{2} + O(n^2) \cdot h \\
&= OPT(mTTP) + O(n^2) \cdot h
\end{aligned}$$

□

## 3.7 Constructing a mTTP(2) Schedule

We use a construction inspired by Imahori [18] to build a SRR schedule such that it can be concatenated with its mirror to yield a DRR schedule which is a feasible mirrored TTP(2) solution. We propose an algorithm for constructing a mirrored schedule when $n/2$ is even, and show it has cost at most $3/2 + \frac{24}{n-2}$ times the optimal mTTP(2) solution. We then propose a slightly modified algorithm to construct mTTP(2) solution when $n/2$ is odd, and prove its cost to be at most $3/2 + \frac{24}{n-4}$ times the optimal solution. Thus for all values of $n$ we have a valid method of constructing a mirrored schedule with an approximation ratio of $3/2 + O(1)/n$.

### 3.7.1 Lower Bound

Let $T$ be the set of $n$ teams. For each team $i \in T$, let $s(i) = \sum_{j \in T} c(i, j)$ be its *star weight*. We have

$$\sum_{i \in T} s(i) = \sum_{i \in T} \sum_{j \in T} c(i, j) = \Delta$$

We can represent the set of teams and the distances between them by a complete graph on $n$ vertices, $K_n$. The edge costs are the distances between team venues and all distances are assumed to be metric.

Let $c(M)$ be the cost of a minimum weight perfect matching on $K_n$. The travelling distance for any team $i$ is at least $s(i) + c(M)$. Thus

$$OPT \geq \sum_{i \in T} (s(i) + c(M)) = \Delta + n \cdot c(M)$$

### 3.7.2 Algorithm for n/2 even

Given the complete graph on the $n$ teams we compute a minimum weight perfect matching $M$ on it. Then, for all edges included in the matching, we compute the combined star weight of its endpoints. The endpoints with the lowest combined star weight are then labelled $(n-1, n)$ and we have that $s(n-1) + s(n) \leq \frac{4\Delta}{n}$. The remaining vertices are labelled such that the edges $(i, i+1)$ for all even $i$ and $(1, n-2)$, make up the edges of $M$. Additionally, we choose an ordering such that

$$\sum_{i \text{ odd}} s(i) \geq \sum_{j \text{ even}} s(j)$$

We begin by creating pairs of teams, each pair consists of teams $(i, i+1)$ for all odd $i$. We then schedule games between pairs of teams. As in the Circle Method used by Thielen and Westphal [30], teams circle around a graph to play each other, but now we fix the home-away patterns on each node. We construct a single round-robin schedule in this way, and due to our choice of home-away assignments, we are then able to concatenate the SRR schedule with its mirror to yield a feasible mirrored DRR schedule. See Figure 3.4 to Figure 3.6 for an illustration of the construction on 12 teams. See Figure 3.7 for an interpretation of how games between pairs of teams are decided.

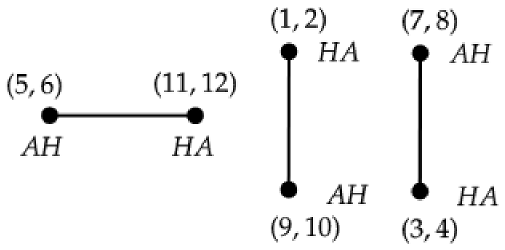


Figure 3.4: Initial configuration. First stage of the SRR schedule for 12 teams

To ensure that we have a feasible SRR schedule, teams within the same pair play each other on the first day of the schedule where team $i$ plays at team $i+1$. Teams then proceed with their scheduled games in the defined home-away pattern. After the first day where

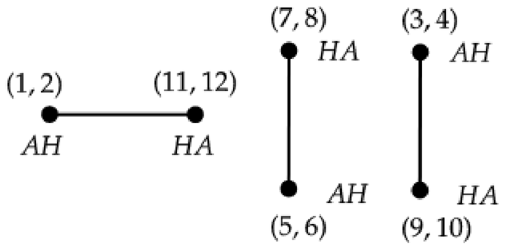


Figure 3.5: Second stage of the SRR for 12 teams

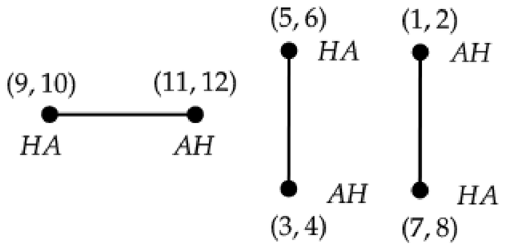
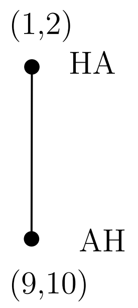


Figure 3.6: Final stage of the SRR schedule for 12 teams

teams play within their pair, there are $n/2-1$ stages of the schedule (corresponding to $n-2$ days) where teams are cycling through the graph. This creates a feasible SRR schedule since all pairs have been matched once, thus all teams have played all other teams once.

The home-away assignment of the horizontal arrow changes only at the last day, while the home-away assignment of all vertical arrows remain fixed and alternate from node to node. The alternating home-away assignments of nodes creates a pattern of $HAAHHA \ldots AHHA$. Thus as teams move along the vertical edges, they have consecutive home games followed by consecutive away games. Figure 3.8 shows an example of the configuration with 20 teams. The larger number of teams allows for a larger number of vertical arrows thus the alternating home-away assignments of the nodes ensures that a majority of the trips taken by any team (excluding $(n-1, n)$) will be of length 2.



| Team / Round | 1 | 2 | 9 | 10 |
|---|---|---|---|---|
| 1 | 9 | 10 | @1 | @2 |
| 2 | @10 | @9 | 2 | 1 |

Figure 3.7: Games played between pairs

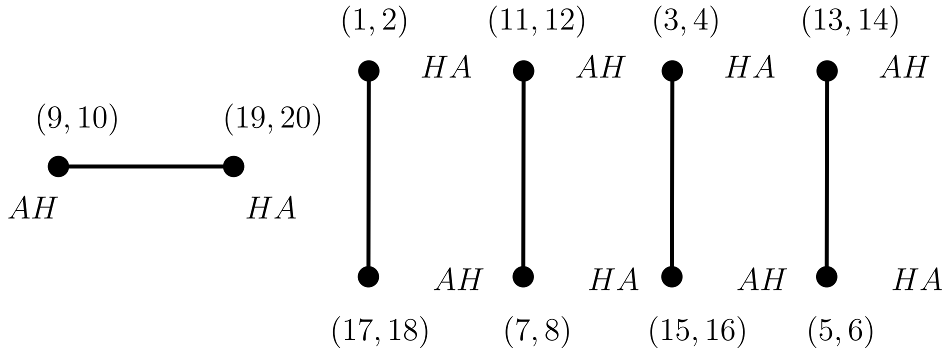


Figure 3.8: Example with 20 teams.

Once the SRR schedule is constructed, we take its mirror and concatenate it with the original SRR schedule thus creating a DRR schedule. We now verify that the DRR schedule is indeed a feasible mTTP(2) solution.

Firstly we note that there is no team that plays more than two consecutive home or away games in either half of the DRR schedule due to the alternating home-away assignments on the nodes. Moreover, there is no streak of more than two home or away games that occur at the meeting point of the two SRR schedules due to our construction. Note that if a team ends the first half on a HA node, then it started the season on a AH node and vice-versa. Let $(i, i+1)$ be a pair of teams that ended on a HA node. Then on day 1, $i$ played at team $i+1$ so it has a home-away pattern of $AAH$ to start the season, and a pattern of $HA$ to finish. On the other hand, on day 1 team $i+1$ plays at home thus has a pattern of $HAH$ to start the season. Table 3.7 shows the home-away patterns at the point where the two SRR schedules meet for teams $i$ and $i+1$. Because of our choice of patterns, there is no streak of more than two consecutive home or two consecutive away games for $i$ or $i+1$. The result is symmetric for any pair of teams that end the first half on a $AH$ node. We conclude that the DRR schedule created by concatenating the SRR schedule and its mirror is indeed a feasible solution to mTTP(2).

| Team | $i$ | $i+1$ |
|---|---|---|
| Day $n-2$ : | H | H |
| Day $n-1$ : | A | A |
| Day $n$ : | H | A |
| Day $n+1$ : | H | H |
| Day $n+2$ : | A | A |

Table 3.7: Home-away pattern at the meeting point of the two SRR schedules

The full DRR schedule obtained from our construction is shown in Table 3.8. Columns correspond to teams while rows are days.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | @2 | 1 | @4 | 3 | @6 | 5 | @8 | 7 | @10 | 9 | @12 | 11 |
| 2 | 9 | 10 | 7 | 8 | @11 | @12 | @3 | @4 | @1 | @2 | 5 | 6 |
| 3 | @10 | @9 | @8 | @7 | 12 | 11 | 4 | 3 | 2 | 1 | @6 | @5 |
| 4 | @ 11 | @12 | @9 | @10 | @7 | @8 | 5 | 6 | 3 | 4 | 1 | 2 |
| 5 | 12 | 11 | 10 | 9 | 8 | 7 | @6 | @5 | @4 | @3 | @2 | @1 |
| 6 | @ 3 | @ 4 | 1 | 2 | 9 | 10 | @11 | @12 | @5 | @6 | 7 | 8 |
| 7 | 4 | 3 | @2 | @1 | @10 | @9 | 12 | 11 | 6 | 5 | @8 | @7 |
| 8 | 5 | 6 | @11 | @12 | @1 | @2 | @9 | @10 | 7 | 8 | 3 | 4 |
| 9 | @6 | @5 | 12 | 11 | 2 | 1 | 10 | 9 | @8 | @7 | @4 | @3 |
| 10 | @7 | @8 | @5 | @6 | 3 | 4 | 1 | 2 | 11 | 12 | @9 | @10 |
| 11 | 8 | 7 | 6 | 5 | @4 | @3 | @2 | @1 | @12 | @11 | 10 | 9 |
| 12 | 2 | @1 | 4 | @3 | 6 | @5 | 8 | @7 | 10 | @9 | 12 | @11 |
| 13 | @9 | @10 | @7 | @8 | 11 | 12 | 3 | 4 | 1 | 2 | @5 | @6 |
| 14 | 10 | 9 | 8 | 7 | @12 | @11 | @4 | @3 | @2 | @1 | 6 | 5 |
| 15 | 11 | 12 | 9 | 10 | 7 | 8 | @5 | @6 | @3 | @4 | @1 | @2 |
| 16 | @12 | @11 | @10 | @9 | @8 | @7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 17 | 3 | 4 | @1 | @2 | @9 | @10 | 11 | 12 | 5 | 6 | @7 | @8 |
| 18 | @4 | @3 | 2 | 1 | 10 | 9 | @12 | @11 | @6 | @5 | 8 | 7 |
| 19 | @5 | @6 | 11 | 12 | 1 | 2 | 9 | 10 | @7 | @8 | @3 | @4 |
| 20 | 6 | 5 | @12 | @11 | @2 | @1 | @10 | @9 | 8 | 7 | 4 | 3 |
| 21 | 7 | 8 | 5 | 6 | @3 | @4 | @1 | @2 | @11 | @12 | 9 | 10 |
| 22 | @8 | @7 | @6 | @5 | 4 | 3 | 2 | 1 | 12 | 11 | @10 | @9 |

Table 3.8: Full mirrored DRR schedule for our example on 12 teams

### 3.7.3 Counting the trips taken with 12 teams

To analyze the cost of the full mirrored schedule produced by our algorithm, we split it into several parts and find a bound on the cost of each part. To facilitate the process we consider the schedule on 12 teams and demonstrate how we count the cost on this example. We start by splitting the trips taken into several parts and analyze their costs individually.

First we consider the trips involving the fixed teams $n, n-1$. We assume that before any other team plays at a fixed pair's venue, they first return home and then travel from their home venue to the fixed team's venue. This detour may be more costly but it is beneficial for our analysis. Similarly, any time one of the fixed teams play an away game, we assume they travel from their home venue to their opponent's venue and then return home directly afterwards. The set of edges in this component is denoted $C_n$.

Next we consider the games taking place on the very first day of the SRR schedule. These are the games played between teams within the same pair. For odd $i$, we assume team $i$ plays at $i+1$ then returns home. In the mirrored half of the season, $i+1$ travels to $i$ and then returns home. The edge $(i, i+1)$ is therefore used twice in the first half of the season, and twice again in the second half. The set of edges counted in this component is denoted $C_p$.

The set of edges used in all the remaining games is denoted $C_o$.

The trips taken by even and odd teams are counted differently. We use team 7 to illustrate how the cost of an arbitrary odd team is counted. By inspecting Table 3.8 we notice that team 7 plays the other teams in cyclic order. This is by design. Furthermore, notice that team 7 plays consecutive home/away games when going from one pair to the next. For example, team 7 plays the pair $(3, 4)$ followed by the pair $(5, 6)$ and plays away at team 4 on day 14 and then plays away at team 5 on day 15. Thus team 7 travels directly from team 4 to team 5, incurring the cost of the edge $(4, 5)$. Recall that we chose the perfect matching $M$ to include the edges $(j, j+1)$ for all even $j$, therefore the edge $(4, 5)$ is in the matching. In fact, all of the trips between two cities will be of this sort for team 7. To count the edges used in these trips we need to count the edges in the matching, $M$, since team 7 uses the edges between pairs, and we also need to count the edges used by team 7 when departing and returning home. In total, team 7 uses the edges $s(7) + M$. Figure 3.9 shows all the trips of length 2 we've counted for team 7 thus far.

We are not done yet since the games before and after playing the pair $(11, 12)$ cannot be counted in this manner. The game before playing the fixed pair is counted as a single trip since team 7 returns home before playing $(11, 12)$, and similarly for the game played after visiting the fixed pair. For team 7, the games against team 6 and team 9 are counted

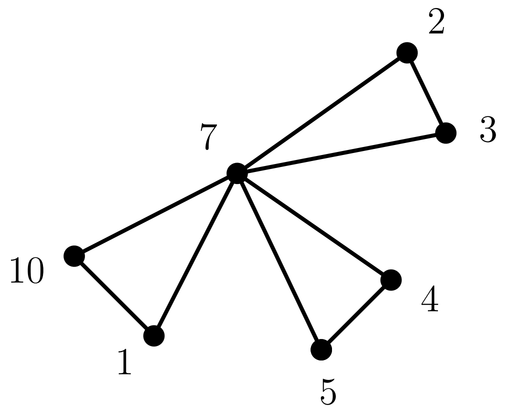


Figure 3.9: Trips of length 2 taken by Team 7

as single trips. Figure 3.10 shows all trips we've counted for team 7 excluding the games against its partner team 8 and against the fixed pair $(11, 12)$. The edges counted in this component, made up of the trips of length two, as well as the trips taken before and after visiting $(11, 12)$, are all included in $C_o$.

The only trips left to count are those that occur on the second day of each season half, as well as the games on the last day of each half and we count these as single trips. For team 7, these are the trips to team 2 and team 3. The set of edges counted by this component is denoted $C_s$.

We now claim that we have counted the cost of all trips taken by team 7. Figure 3.11 shows the actual trips taken by team 7 in the DRR schedule constructed by our algorithm. The trips to 11 and 12 were counted in $C_n$ and are shown in black. The trips of length 2, as well as the trips before and after visiting $(11, 12)$ were counted by $C_o$ and are shown in green. The trips taken on the second and last days of the schedule were counted in $C_s$ and are shown in blue. The trip to its partner team was counted in $C_p$ and is shown in red.

Note the trip visiting teams 3 and 8 includes the edge $(3, 8)$ which was not actually counted in any of our components. However, the edge $(7, 8)$ was counted twice in $C_p$, and the edge $(3, 7)$ was counted twice in $C_s$, thus by the triangle inequality, we incurred at least the cost of the edge $(3, 8)$. Similarly, we remark that the edge $(6, 11)$ is used by team 7 in the schedule but was not counted. However, the edge $(7, 11)$ was counted twice in $C_n$, and the edge $(6, 7)$ was counted twice in $C_o$: once by the star weight $s(7)$, and once in the matching $M$. Applying the triangle inequality once more we conclude that we incurred a cost of at least the cost of all edges actually used in the schedule for team 7.

Now we turn to the even teams. We resort to a simplistic method of counting the trips

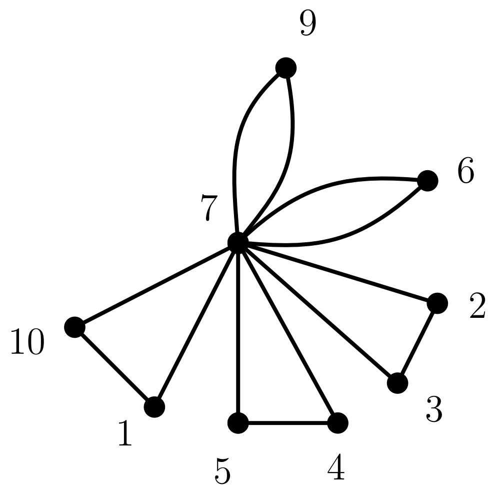


Figure 3.10: Trips taken by Team 7 including single trips

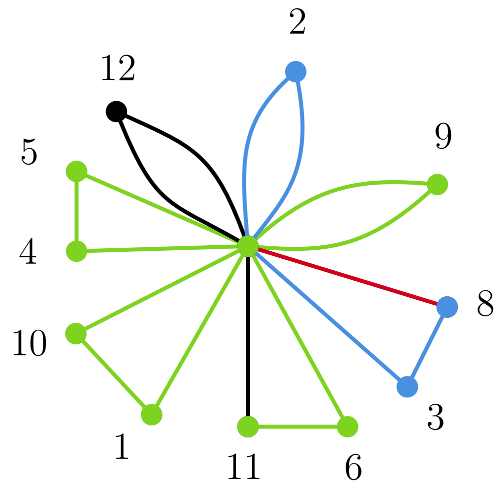


Figure 3.11: Actual trips taken by Team 7

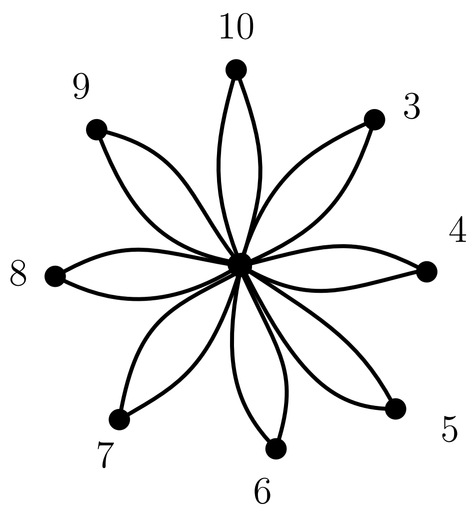


Figure 3.12: Trips counted for Team 2

taken by assuming that all even teams take single trip when visiting all other teams. That is, even teams starts from their home venue, play an away game at their opponent venue, and then return home. Figure 3.12 shows the trips counted in $C_o$ for team 2. Note that in the actual schedule constructed, team 2 will have some trips of length 2 thus shortcutting the single trips we have counted. However, since the distances are metric, the triangle inequality implies that our counting of trips is never longer than the actual trips taken by team 2. Figure 3.13 shows the actual edges used by team 2 in the DRR schedule constructed by our algorithm. The trips to 11 and 12 were counted in $C_n$ and are shown in black. All other trips were counted by $C_o$ and are shown in green. The trips taken on the second and last days of the schedule were counted in $C_s$ and are shown in blue. The trip to its partner team was counted in $C_p$ and is shown in red.

Note that all of the actual edges used by team 2 in the schedule we constructed are included in at least one of the edge sets we defined, thus all the trips have been counted. Therefore we have successfully counted all trips taken by team 7 and team 2, which we used as examples of arbitrary odd and even teams respectively. To count the trips of any other odd (respectively, even) team, we follow the same process described to count the trips of team 7 (respectively, team 2). All that remains to do is bound the cost of each component to arrive at a bound on the total cost of the schedule constructed.

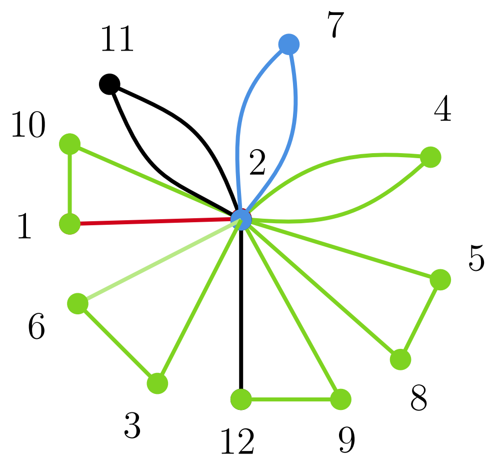


Figure 3.13: Actual trips taken by Team 2

### 3.7.4 Cost Analysis

Having seen how we count the trips taken by an even and an odd team in the previous section, we now focus on the task of computing the cost of each component we defined.

Recall our key assumptions: the $n$ teams are numbered such that the edges joining two pairs, $(i, i+1)$ for all even $i$, and $(1, n-2)$, make up the edges of a minimum weight perfect matching $M$. We choose an ordering such that $s(n-1) + s(n) \leq \frac{4\Delta}{n}$. Additionally, we choose an ordering such that

$$\sum_{i \text{ odd}} s(i) \geq \sum_{j \text{ even}} s(j)$$

The costs are split as follows:

$C_h$ : The cost associated with home games at the pair $(n-1, n)$. We assume teams travel to $(n-1, n)$ from their home venue and then return there after the game. Thus $C_h \leq 2(s(n-1) + s(n)) \leq \frac{8\Delta}{n}$.

$C_a$ : The cost associated with away games of the pair $(n-1, n)$. By the same reasoning as above, $C_a \leq 2(s(n-1) + s(n)) \leq \frac{8\Delta}{n}$.

We combine these two parts together, counting the home and away games of $n$ and $n-1$, and denote it $C_n$. We have $C_n \leq \frac{16\Delta}{n}$.

$C_s$ : The cost associated with the games on the second and last day of each season half. The games on the first day of the SRR schedule are games between teams within a pair and are counted separately. We consider the cost as if all games on the second and last days are single trips.

For any edge $(i, j)$ it can be counted in four ways in the full DRR schedule: $i$ and $j$ can play each other on the first or last day, and with alternating venues. If an edge is included, it is used twice since $i$ (or $j$) travel back and forth. Overall there are $n/2 - 1$ choices of initial configurations. By configuration we mean an assignment of pairs to nodes that maintain the relative order, thus each stage of the schedule is a configuration. Each one defines a different set of edges to be counted in this manner and an edge is counted twice in at most 4 of these choices. Over all possible configurations we have a total cost of

$$\sum_{i \in T} \sum_{j \in T} 8 \cdot c(i, j) \leq 4\Delta$$

Consequently, there must be some choice of configuration such that

$$C_s \leq \frac{4\Delta}{n/2-1} = \frac{8\Delta}{n-2}$$

$C_p$ : The cost incurred by games played between teams in the same pair. We bound the cost of these games by using twice the cost of the edge between the teams in the pair. They must play each other twice, each traveling back and forth to do so, thus $C_p \leq 4\sum_{i \text{ odd}} c(i, i+1)$.

$C_o$ : The other costs. Here we further split the costs into the cost incurred by odd teams and even teams. We begin with the cost incurred by all odd teams.

Note that odd teams visit teams within pairs in order, and visit pairs in order as well. By adding an additional edge from the last opponent to the first, we have a cyclic order on the opponents of $i$. Since we have already covered the costs associated with playing teams $n-1$ and $n$, the costs of the remaining trips in the cyclic order are exactly the same as the costs for playing against the opponents in order $i+1, i+2, \ldots, n-2, 1, 2, \ldots, i-1$. This can be seen for team 1 in Table 3.8.

For any team, the game played after having played at $(n-1, n)$ is a single trip due to the alternating home-away assignment of our construction. Note that a team $i$ will play team $j$ after having played $(n-1, n)$ if $j$ is two nodes away from $i$ (going in a clockwise manner). In Figure 3.4 this means team 1 will play 3, team 3 will play 5, and so on. The games after having played $(n-1, n)$ result in a single trip for the teams at the bottom leftmost node, and since this node has a fixed home-away assignment of $AH$, it will only be the first team in the pair that performs this single trip in the first SRR schedule. In the mirrored SRR schedule, this cost will be counted by the even team in the pair thus its cost is handled later. The cost of this single trip for team $i$ is $2c(i, i+2)$. This pattern does not hold for team $n-3$ who plays at $(n-1, n)$ in the final stage of the SRR schedule. Instead, for this team, they have a single trip in the game *before* playing $(n-1, n)$. This difference comes from the fact that we swap the home-away assignment of the vertical arrow in the last stage. For the team $n-3$, they take a single trip to team $n-4$. We remark that this edge is in $M$ thus we do not need to bound its cost. For all other odd teams we use the triangle inequality:

$$c(i, i+2) \leq c(i, i+1) + c(i+1, i+2) \quad \forall i \neq n-3$$

Thus the total cost of these single trips for all odd teams is

$$\begin{aligned}
&2 \sum_{\substack{i \text{ odd} \\ i \neq n-3}} c(i, i+2) + 2c(n-3, n-4) \\
&\leq \sum_{\substack{i \text{ odd} \\ i \neq n-3}} c(i, i+2) + c(i, i+1) + c(i+1, i+2) + 2c(n-3, n-4) \\
&\leq \sum_{\substack{i \text{ odd} \\ i \neq n-3}} c(i, i+1) + c(i, i+2) + 3c(M)
\end{aligned}$$

Now that we have covered the cost of the single trips taken by $i$ throughout the entire DRR schedule. The only trips left are those of length two, and since any home-stand in the first half of the schedule is a road trip in the mirrored half, we can compute the total cost for the full schedule. These trips of length two occur when $i$ travels from one pair to another, and include the edge $(j, j+1)$ for $j$ even, which is an edge in $M$. As shown in Figure 3.17, $i$ travels to the second team $j$ in a pair, then uses the edge $(j, j+1)$ to get to the next pair, then returns home from team $j+1$.

We have already covered the cost of teams playing within their own pair, and games against $n-1$ and $n$, and we counted the single trips between $i$ and $i+2$. The cost for team $i$ when $i \neq n-3$ is therefore at most

$$s(i) - c(i, i+1) - c(i, i+2) - c(i, n-1) - c(i, n) + c(M)$$

For team $n-3$ we replace the term $c(i, i+2)$ by the term $c(n-3, n-4)$ but we can ignore it since the cost of this edge is included in $c(M)$.

The total cost of all trips of length 2 for all odd teams is therefore at most

$$\sum_{i \text{ odd}} (s(i) - c(i, i+1) - c(i, n-1) - c(i, n)) + (n/2 - 1) \cdot c(M) - \sum_{\substack{i \text{ odd} \\ i \neq n-3}} c(i, i+2)$$

Combining this cost with that of the single trips we have the total cost for odd teams to

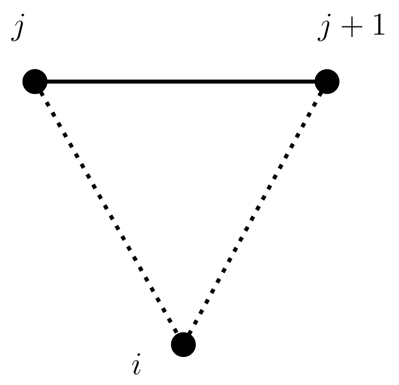


Figure 3.14: Trip of length two for odd $i$ travelling between pairs

be at most

$$\begin{aligned}
&\sum_{i \text{ odd}} (s(i) - c(i,i+1) - c(i,n-1) - c(i,n)) + (n/2-1)\cdot c(M) - \sum_{\substack{i \text{ odd}\\ i\neq n-3}} c(i,i+2) \\
&+ \sum_{\substack{i \text{ odd}\\ i\neq n-3}} (c(i,i+1) + c(i,i+2)) + 3c(M) \\
&\le \sum_{i \text{ odd}} (s(i) - c(i,n-1) - c(i,n)) + (n/2+2)\cdot c(M)
\end{aligned}$$

We now handle the total cost of all even teams. The cost incurred by an even team $j$ can be trivially bounded by $2s(j)$ however this bound can be sharpened by excluding the costs that have been counted elsewhere. Games within pairs were already counted by $C_p$, so we can subtract the cost of the edge between $j$ and its partner team $j-1$. We can also subtract the cost of going to teams $n-1$ and $n$ since these trips are counted by $C_n$. Therefore the cost for an even team $j$ is at most $2(s(j) - c(j,j-1) - c(j,n-1) - c(j,n))$. The total cost of all even teams is at most

$$\begin{aligned}
&2\sum_{j \text{ even}} (s(j) - c(j,j-1) - c(j,n-1) - c(j,n)) \\
&= 2\sum_{j \text{ even}} s(j) - 2\sum_{i \text{ odd}} (c(i,i+1) + c(i+1,n-1) + c(i+1,n))
\end{aligned}$$

Putting the cost of the even and odd teams together we arrive at

$$\begin{aligned}
C_o &\leq \sum_{i \text{ odd}} s(i) + (n/2+2) \cdot c(M) + 2 \sum_{j \text{ even}} s(j) - 2 \sum_{i \text{ odd}} c(i, i+1) - s(n) - s(n-1) \\
&= \Delta + (n/2+2) \cdot c(M) + \sum_{j \text{ even}} s(j) - 2 \sum_{i \text{ odd}} c(i, i+1) - s(n) - s(n-1)
\end{aligned}$$

The final cost of the entire DRR schedule can be bounded by putting all parts together:

$$\begin{aligned}
C_n + C_s + C_p + C_o \leq \frac{16\Delta}{n} + \frac{8\Delta}{n-2} + 4 \sum_{i \text{ odd}} c(i, i+1) + \Delta + (n/2+2) \cdot c(M) \\
+ \sum_{j \text{ even}} s(j) - 2 \sum_{i \text{ odd}} c(i, i+1) - s(n) - s(n-1) \\
\leq (3/2) \cdot \Delta + (n/2+2) \cdot c(M) + \frac{24\Delta}{n-2}
\end{aligned}$$

To arrive at the final inequality we use our assumption that $s(i) \geq s(i+1)$ for all odd $i$. Moreover,

$$\sum_{i \text{ odd}} s(i) + s(i+1) = \Delta$$

thus we have that

$$\sum_{j \text{ even}} s(j) \leq \frac{1}{2}\Delta$$

We also use the triangle inequality to find $c(i, i+1) \leq c(i, n) + c(i+1, n)$. Similarly, $c(i, i+1) \leq c(i, n-1) + c(i+1, n-1)$ Therefore

$$2 \sum_{i \text{ odd}} c(i, i+1) \leq s(n) + s(n-1)$$

We have a lower bound of $\Delta + n \cdot c(M)$ on the cost of any feasible DRR schedule, thus we have found an approximation ratio of $\frac{3}{2} + \frac{24}{n-2}$.

### 3.7.5 Algorithm for n/2 odd

The case where $n/2$ is odd is similar in construction to the even case. Given a complete graph on the $n$ teams, we choose an ordering of the teams such that $s(n) + s(n-1) +$

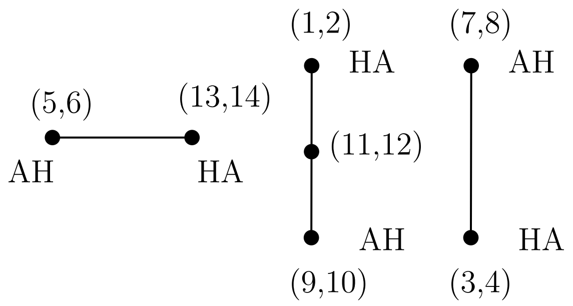


Figure 3.15: Initial configuration for 14 teams

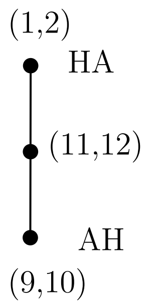
$s(n-2) + s(n-3) \leq \frac{4\Delta}{n}$. We compute a minimum weight perfect matching on the graph and order the nodes such that the edges $(i, i+1)$ and $(1, n)$, where $i$ is even, make up the edges of the matching. Moreover, we ensure that our ordering of the teams satisfies

$$\sum_{i \text{ odd}} s(i) \geq \sum_{j \text{ even}} s(j)$$

Once the teams are ordered, we create pairs of teams, each pair consisting of teams $i$ and $i+1$ for all odd $i$.

The configuration now has two pairs of teams that are fixed. The pair $(n, n-1)$ is still fixed on the horizontal arrow while $(n-2, n-3)$ is fixed at the center node of the leftmost vertical arrow and is the only arrow with three nodes on it. The other pairs of teams rotate counterclockwise through the graph as before to visit all other pairs. Figure 3.16 shows an example of the games generated by the edge with three nodes. The top and bottom pairs play according to their HA assignment while the HA pattern of $(11, 12)$ is determined by the other pairs. Thus team 11 will play HH while team 12 plays AA. To avoid creating a streak of more than two consecutive home or away games for either of these teams, they alternate their role each stage, meaning in the following stage, 11 will play the games assigned to 12 and vice-versa. This has the desired effect of having teams 11 and 12 play HHAAHH...

After $n/2 - 2$ stages, each pair has done a full traversal of the graph, and each team has played $n-3$ games (including the game against their partner on the very first day). Each team is still missing two games. This is because when pairs visit the left-most vertical arrow, they only play one team from each of the other two pairs thus visiting this edge twice means each team has a deficit of two games. To illustrate where the missing games come from, consider the pair $(1, 2)$. When this pair is the top of the triple edge, team 1 plays teams 9 and 12 while team 2 plays against teams 10 and 11. In their second visit to



| Team / Round | 1 | 2 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|
| 1 | 12 | 10 | @11 | @2 | 9 | @1 |
| 2 | @9 | @11 | 1 | 12 | 2 | @10 |

Figure 3.16: Games between pairs on the edge with three nodes.

the triple edge, $(1, 2)$ is the bottom node and this time team 1 plays teams 11 and 3 while team 2 plays teams 12 and 4. Thus we see that the pair $(1, 2)$ has played all their games against pair $(11, 12)$ but are missing games against the other two pairs that were matched to the triple edge with them. Team 1 must still play teams 10 and 4 while team 2 is missing games against teams 9 and 3. Since the pair $(1, 2)$ behaves like all other non-fixed pairs, we have established that all teams will be missing two games after running our algorithm for a full traversal of the graph. The teams in the fixed pairs will in fact also be missing two games since $(n - 1, n)$ was never matched with $(n - 3, n - 2)$, thus every team needs to play two more games to complete the SRR schedule. We complete the SRR schedule by adding two days at the end and schedule the missing games. When scheduling the missing games, we enforce a HA pattern of $HA$ for odd teams and $AH$ for even teams, with the exception of assigning $AA$ for team $n - 2$ and $HH$ for team $n - 3$. Table 3.9 shows the full SRR schedule constructed in this manner with 14 teams.

**Claim 6.** *The choice of HA assignments on the last two days of the SRR schedule ensures it can be mirrored to produce a feasible mTTP(2) solution for any number of teams.*

*Proof.* The SRR schedule created has no streak of home or away games longer than 2 games. However, when combining the SRR schedule with its mirror, a streak may be created at the point where the two halves meet. To avoid such a scenario, the HA assignment of the first and last two days is chosen carefully. There are two problematic cases that may arise: a team plays at home on the first two days, then plays away on the last day. This creates a streak of 3 consecutive away games when mirrored. The second problematic case is the opposite; a team plays away on the first two days and home on the last day. Note that on day 1, team $i$ plays at team $i + 1$ for all odd $i$. Thus the only teams that can have two consecutive away games to start the season are odd teams. For this reason, we choose odd teams to play away on the last day of the SRR schedule. Similarly, only even teams can

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | @2 | 1 | @4 | 3 | @6 | 5 | @8 | 7 | @10 | 9 | @12 | 11 | @14 | 13 |
| 2 | 12 | 10 | 7 | 8 | @13 | @14 | @3 | @4 | @11 | @2 | 9 | @1 | 5 | 6 |
| 3 | @9 | @11 | @8 | @7 | 14 | 13 | 4 | 3 | 1 | 12 | 2 | @10 | @6 | @5 |
| 4 | @13 | @14 | @9 | @10 | @12 | @8 | 11 | 6 | 3 | 4 | @7 | 5 | 1 | 2 |
| 5 | 14 | 13 | 10 | 9 | 7 | 11 | @5 | @12 | @4 | @3 | @6 | 8 | @2 | @1 |
| 6 | @11 | @4 | 12 | 2 | 9 | 10 | @13 | @14 | @5 | @6 | 1 | @3 | 7 | 8 |
| 7 | 3 | 12 | @1 | @11 | @10 | @9 | 14 | 13 | 6 | 5 | 4 | @2 | @8 | @7 |
| 8 | 5 | 6 | @13 | @14 | @1 | @2 | @12 | @10 | 11 | 8 | @9 | 7 | 3 | 4 |
| 9 | @6 | @5 | 14 | 13 | 2 | 1 | 9 | 11 | @7 | @12 | @8 | 10 | @4 | @3 |
| 10 | @7 | @8 | @11 | @6 | 12 | 4 | 1 | 2 | @13 | @14 | 3 | @5 | 9 | 10 |
| 11 | 8 | 7 | 5 | 12 | @3 | @11 | 2 | 1 | 14 | 13 | 6 | @4 | @10 | @9 |
| 12 | 10 | @3 | 2 | @5 | 4 | @7 | 6 | @9 | 8 | @1 | @13 | 14 | 11 | @12 |
| 13 | @4 | 9 | @6 | 1 | @8 | 3 | @10 | 5 | @2 | 7 | @14 | 13 | @12 | 11 |

Table 3.9: SRR schedule for 14 teams

play two consecutive home games to start the SRR schedule thus they play at home on the last day of the SRR schedule. Enforcing these conditions ensures that no team plays more than two consecutive home or away games in the full DRR schedule produced by the concatenation of the SRR schedule and its mirror. □

### 3.7.6 Cost Analysis

Our analysis of the cost is virtually identical in content and organization to the analysis of the algorithm for $n/2$ being even, presented in section 3.7.4. Nevertheless we repeat it in full detail here.

We split the cost of the schedule into several parts and bound each cost individually. Recall our key assumptions: the $n$ teams are numbered such that the edges joining two pairs, $(i, i+1)$ for all even $i$ and $(1, n)$, make up the edges of a minimum weight perfect matching $M$. We choose an ordering such that $s(n) + s(n-1) + s(n-2) + s(n-3) \leq \frac{4\Delta}{n}$. Additionally, we choose an ordering such that

$$\sum_{i \text{ odd}} s(i) \geq \sum_{j \text{ even}} s(j)$$

For ease of presentation, let $n_f$ denote the collection of fixed teams $n, n-1, n-2$ and

$n-3$. Using this notation we have

$$s(n_f) = s(n) + s(n-1) + s(n-2) + s(n-3)$$
$$c(i, n_f) = c(i, n) + c(i, n-1) + c(i, n-2) + c(i, n-3)$$

We now present how we split the cost of the schedule:

$C_h$ : The cost associated with home games at the pairs $(n-1, n)$ and $(n-2, n-3)$. We assume teams travel to these two pairs from their home venue and then return there after the game. Thus $C_h \leq 2s(n_f) \leq \frac{8\Delta}{n}$

$C_a$ : The cost associated with away games of the pairs $(n-1, n)$ and $(n-2, n-3)$. By the same reasoning as above, $C_a \leq 2s(n_f) \leq \frac{8\Delta}{n}$.

We combine these two parts together thus counting the home and away games of the four fixed teams and denote it $C_n$. We have $C_n \leq \frac{16\Delta}{n}$.

$C_s$ : The cost associated with the games on the second and last day of each season half. The games on the first day of the SRR schedule are games between teams within a pair and are counted separately. We consider the cost as if all games on the second and last days will be single trips.

For any edge $(i, j)$ it can be counted in four ways in the full DRR: $i$ and $j$ can play each other on the first or last day with alternating venues. Finally, if such an edge is included, it is used twice since $i$ (or $j$) will use it to go back and forth. Overall there are $n/2 - 2$ choices of initial configurations. A configuration is an assignment of pairs to nodes that maintain the relative order so each stage of the schedule is a configuration. Each one defines a different set of edges to be counted in this manner and an edge is counted twice in at most 4 of these choices. Over all possible configurations we have a total cost of

$$\sum_i \sum_j 8 \cdot c(i, j) \leq 4\Delta$$

Consequently, there must be some choice of configuration such that

$$C_s \leq \frac{4\Delta}{n/2 - 2} = \frac{8\Delta}{n-4}$$

$C_p:$ The cost incurred by games played between teams in the same pair. We can bound the cost of these games by using twice the cost of the edge between the teams in the pair. They must play each other twice, each traveling back and forth to do so, thus $C_p \leq 4\sum_{i \text{ odd}} c(i, i+1)$.

$C_o:$ The other costs not counted by the above.

We begin by considering the games played on the triple edge and bounding their cost as if they were single trips. For any team $i$, there are six games played due to this triple edge but two of these games are against the fixed pair $(n-2, n-3)$ and their cost is included in $C_n$. Another one of these games is played on the very last day of the SRR schedule and thus is counted by $C_s$ which leaves three games to be counted. For a team $i$, let $q_1, q_2, p_1$ be the teams it plays in these three remaining games. The cost associated with these three games is at most

$$2(c(i, q_1) + c(i, q_2) + c(i, p_1))$$

By the triangle inequality, $c(i, j) \leq c(i, n) + c(j, n)$. The idea is to use this inequality to bound the total cost of games played on the triple edge:

$$\sum_{i \in T}(c(i, q_1) + c(i, q_2) + c(i, p_1)) \leq s(n) + s(n-1) + s(n-2)$$

**Lemma 5.** *An edge $(i, j)$ used in the games associated with the triple edge is counted by the star weight of at most one fixed team.*

For now, we ignore the existence of the triple edge and we examine the resulting cost incurred by odd teams and even teams. We begin with counting the cost of trips taken by odd teams.

Note that odd teams visit teams within pairs in order, and visit pairs in order as well. By adding an additional edge from the last opponent to the first, we have a cyclic order on the opponents of $i$. The costs of the trips in the cyclic order are exactly the same as the costs for playing against the opponents in order $i+1, i+2, \ldots, n, 1, 2, \ldots, i-1$. This can be seen for team 1 in Figure 3.9.

We covered the cost of the single trips taken by $i$ throughout the entire DRR schedule, thus the only trips left are those of length two. Moreover, since any home-stand in the first half of the schedule is a road trip in the mirrored half, we can compute the total cost for the full schedule. These trips occur when $i$ travels from one pair to another and include

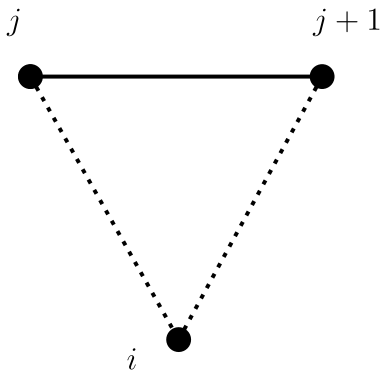


Figure 3.17: Trip of length two for odd $i$ travelling between pairs

the edge $(j, j+1)$ for $j$ even, which is an edge in $M$. As shown in Figure 3.14, $i$ travels to the second team $j$ in a pair, then uses the edge $(j, j+1)$ to get to the next pair, then returns home from team $j+1$.

We have already covered the cost of teams playing within their own pair, and games against the fixed teams, and we counted the trips played on the triple edge. The cost for team $i$ is therefore at most

$$s(i) - c(i, i+1) - c(i, n_f) + c(M)$$

The total cost of all trips of length two for all odd teams is therefore at most

$$\sum_{i \text{ odd}} (s(i) - c(i, i+1) - c(i, n_f)) + (n/2 - 2) \cdot c(M)$$

Combining this cost with that of the single trips we find the total cost for odd teams to be at most

$$\begin{aligned}
&\sum_{i \text{ odd}} (s(i) - c(i, i+1) - c(i, n_f)) + (n/2 - 2) \cdot c(M) + \text{“triple edge costs”} \\
&\leq \sum_{i \text{ odd}} (s(i) - c(i, i+1) - c(i, n_f)) + (n/2 - 2) \cdot c(M) + s(n) + s(n-1) + s(n-2) \\
&= \sum_{i \text{ odd}} (s(i) - c(i, i+1) - c(i, n_f)) + (n/2 - 2) \cdot c(M) + s(n_f) - s(n-3)
\end{aligned}$$

We now turn to the cost for even teams. The cost incurred by an even team $j$ can be trivially bounded by $2s(j)$ however this bound can be sharpened by excluding the costs that have been counted elsewhere. Games within pairs were already counted by $C_p$, so we can subtract the cost of the edge between $j$ and its partner team $j-1$. We can also subtract the cost of playing against the four fixed teams since these trips are counted by $C_n$. We now have that cost for even team $j$ is at most

$$2(s(j) - c(j, j-1) - c(j, n_f))$$

The total cost of all even teams is therefore at most

$$2\sum_{j \text{ even}} (s(j) - c(j, j-1) - c(j, n_f)) = 2\sum_{j \text{ even}} s(j) - 2\sum_{i \text{ odd}} (c(i, i+1) + c(i+1, n_f))$$

Putting the cost of the even and odd teams together we arrive at

$$\Delta + (n/2 - 2) \cdot c(M) + \sum_{j \text{ even}} s(j) - 3\sum_{i \text{ odd}} c(i, i+1) - s(n-3)$$

The final cost of the entire DRR schedule can be bounded by putting all parts together:

$$\begin{aligned}
&C_n + C_s + C_p + C_o \\
&\leq \frac{16\Delta}{n} + \frac{8\Delta}{n-4} + 4\sum_{i \text{ odd}} c(i, i+1) + \Delta + (n/2 - 2) \cdot c(M) \\
&+ \sum_{j \text{ even}} s(j) - 3\sum_{i \text{ odd}} c(i, i+1) - s(n-3) \\
&\leq (3/2) \cdot \Delta + (n/2 - 2) \cdot c(M) + \frac{24\Delta}{n-4}
\end{aligned}$$

To arrive at the final inequality we use our assumption that $s(i) \geq s(i+1)$ for all odd $i$. Moreover,

$$\sum_{i \text{ odd}} s(i) + s(i+1) = \Delta$$

thus we have that

$$\sum_{j \text{ even}} s(j) \leq \frac{1}{2}\Delta$$

We also use the triangle inequality to find $c(i, i+1) \leq c(i, n-3) + c(i+1, n-3)$. Therefore

$$\sum_{i \text{ odd}} c(i, i+1) \leq s(n-3)$$

We have a lower bound of $\Delta + n \cdot c(M)$ on the cost of any feasible DRR schedule, thus we have found an approximation ratio of $\frac{3}{2} + \frac{24}{n-4}$.

# Chapter 4

# Local Search Methods

## 4.1 Introduction

Given an optimization problem that is computationally hard to solve, local search methods may be used to find an optimal solution. For the local search methods studied in this chapter, we assume that each instance of the problem has a discrete set of feasible solutions. A local search algorithm starts from a candidate solution and then iteratively moves to a neighboring solution until a stopping criterion is met. Each local search algorithm is therefore defined by how the neighborhood of solutions is chosen, as well as its stopping criterion.

A discrete optimization problem is described by $\Pi = (\mathcal{I}, \mathcal{S})$ where $\mathcal{I}$ is the set of instances and $\mathcal{S}(x)$ is the discrete set of feasible solutions for each instance $x \in \mathcal{I}$. To have an optimization problem we need a measure of the quality of a solution thus we must have an objective function $f : \mathcal{S}(x) \to \mathbb{R}$ that evaluates each feasible solution. Given an instance of the problem $x$, the goal is then to find a feasible solution $y \in \mathcal{S}(x)$ with minimum objective function value $f(y)$.

## 4.2 Applications to TTP

Below is a collection of papers which have made use of local search methods to solve TTP along with a brief description of their contents. Three papers are presented in further detail in the following sections. The most recent results for all benchmark instances are available at [31].

**A simulated annealing approach to the traveling tournament problem:** In one of the earliest examples of local search methods being applied to TTP, Anagnostopoulos, Michel, Van Hentenryck, and Vergados [1] present a simulated annealing approach to produce high-quality solutions. Experiments were run on the benchmark instances introduced by Easton, Nemhauser and Trick [13]. The algorithm's worst solution quality over 50 runs is always smaller or equal to the best-known solutions at the time.

**Heuristics for the mirrored traveling tournament problem:** This paper by Ribeiro and Urrutia [27] features the first appearance of the mirrored version of the TTP in the literature. They propose a new heuristic based on the combination of the GRASP and iterated local search meta-heuristic to solve mTTP.

**Clustering Search Approach for the TTP:** Biajoli and Lorena [3] propose a hybrid heuristic to solve the mirrored TTP using Clustering Search. The method consists in detecting supposed promising search areas based on clustering. The performance of the method is shown on benchmark problems available in literature and real benchmark problems, such as the Brazilian Soccer Championship.

**A Variable Neighborhood Search Method for Solving the TTP:** Khelifa and Boughaci [22] make use of a variable neighborhood search (VNS) method to solve TTP(3). They use solution neighborhoods that are similar to those used in [1, 27]. Their method reaches optimal solutions for small instances and beats the previously best known results on several benchmark instances.

**Evolutionary harmony search algorithm for sport scheduling problem:** Khelifa, Boughaci, and Aimeur [23]. The approach succeeds in finding optimal solutions for several instances. For the other instances, the general deviation from optimality is equal to 4.45%. This paper generated the best solutions for the unconstrained TTP for multiple benchmark instances. These results held until 2019.

**A Variable Neighborhood Search for Major League Baseball Problem:** Liang et al. [25] use a VNS scheme inspired by the method of Khelifa and Boughaci [22]. They perform experiments with their algorithm in an attempt to improve the total traveling distance of the 2016 and 2019 MLB schedules. The output of their algorithm shows a 2.5% improvement compared to the actual 2016 schedule and a 6% improvement compared to the actual 2019 schedule.

## 4.3 Heuristics for MTTP

The mirrored version of the traveling tournament problem was introduced by Ribeiro and Urrutia [27] and they proposed heuristic methods for solving it. They begin by describing a three phase method to construct a mTTP schedule which will act as the initial feasible solution to be optimized by conducting local changes on it.

The construction of the mirrored schedule begins by generating a SRR schedule using the circle method. This is done by considering the complete graph $K_n$ representing $n$ abstract teams and their pairwise distance from each other, and partitioning it into $n-1$ perfect matchings. Each matching corresponds to one day of the SRR schedule and each edge in the matching defines a game between the teams on its endpoints. The schedule is then duplicated to create a mirrored DRR. At this stage, we have generated a schedule that tells what match-ups occur on each day, but we have not determined the venue at which these matches occur. This preliminary schedule is used to generate a $n \times n$ matrix where the entry $(i, j)$ is equal to the number of times the teams $i$ and $j$ are consecutive opponents of other teams. The idea is that if a pair of teams $i$ and $j$ are consecutive opponents of many other teams, these two teams should have a small distance between them since many teams will incur the cost of travelling between $i$ and $j$.

This brings us to the second stage of the construction. We now assign real teams to the abstract teams with the intention of minimizing the traveling cost of the schedule we created. We do so by prioritizing the assignment of teams that are close to each other to the abstract teams that are frequently consecutive opponents of other teams. This amounts to solving a particular quadratic assignment problem (QAP) and a quick heuristic is applied to solve it.

The third and final stage of the construction is to assign venues to each of the scheduled games. The choice of venue determines which team will be playing at home or away for each match-up. To minimize the distance travelled, it is preferable to schedule road trips that are as long as possible thus avoiding unnecessary trips to and from a teams home venue. This stage occurs in two parts. First, venues are assigned randomly to yield a feasible schedule, then, a local search is implemented to improve the traveling cost of the schedule. The local search procedure uses the randomly generated assignment of venues as initial solution and the SwapHome neighborhood which is described below. A solution is returned once a local optimum has been reached. The neighborhoods structures used by the heuristic are described as follows:

|  | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 5 | 4 | @6 | @2 | @1 | 3 |
| 2 | 4 | @6 | @5 | @1 | 3 | 2 |
| 3 | 3 | @5 | @1 | @6 | 2 | 4 |
| 4 | 6 | @3 | 2 | @5 | 4 | @1 |
| 5 | 2 | @1 | 4 | @3 | 6 | @5 |
| 6 | @5 | @4 | 6 | 2 | 1 | @3 |
| 7 | @4 | 6 | 5 | 1 | @3 | @2 |
| 8 | @3 | 5 | 1 | 6 | @2 | @4 |
| 9 | @6 | 3 | @2 | 5 | @4 | 1 |
| 10 | @2 | 1 | @4 | 3 | @6 | 5 |

Table 4.1: Schedule with 6 teams

**SwapHome(A,B,$i$) :** For some game played by $A$ and $B$ on day $i$, this moves swaps the venues of the match-up thus swapping the home/away assignment of these teams. This move only affects this match-up and so it causes the least disturbance to the overall schedule. Using the initial schedule on 6 teams shown in Table 4.1, the move SwapHome$(1, 4, 2)$ is then applied to it. The resulting schedule is shown in Table 4.2 where gray colored cells indicate the games affected by the move.

**SwapTeam(A,B) :** For two teams $A$ and $B$, this move swaps all opponents of team $A$ with the opponents of team $B$ over all rounds. So if on day $i$ the match-ups are $(A, C)$ and $(B, D)$, then after this move, the match-ups of day $i$ are $(A, D)$ and $(B, C)$. This process is done for every day of the season. Using the initial schedule on 6 teams shown in Table 4.1, the move SwapTeam$(2, 3)$ is then applied to it. The resulting schedule is shown in Table 4.3 where gray colored cells indicate the games affected by the move.

**PartialSwapRound(A,B,C,D,$i, j$) :** For four teams $A, B, C$ and $D$, and two rounds $i$ and $j$ such that games $(A, B)$ and $(C, D)$ take place in round $i$, and games $(A, C)$ and $(B, D)$ take place in round $j$, this move swaps games taking place on day $i$ with games taking place on day $j$, and vice versa. The result is games $(A, C)$ and $(B, D)$ now take place in round $i$ while games $(A, B)$ and $(C, D)$ now take place in round $j$.

The three neighborhoods SwapHome, SwapTeam, and PartialSwapRound are explored by local search.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 5 | 4 | @6 | @2 | @1 | 3 |
| 2 | @4 | @6 | @5 | 1 | 3 | 2 |
| 3 | 3 | @5 | @1 | @6 | 2 | 4 |
| 4 | 6 | @3 | 2 | @5 | 4 | @1 |
| 5 | 2 | @1 | 4 | @3 | 6 | @5 |
| 6 | @5 | @4 | 6 | 2 | 1 | @3 |
| 7 | 4 | 6 | 5 | @1 | @3 | @2 |
| 8 | @3 | 5 | 1 | 6 | @2 | @4 |
| 9 | @6 | 3 | @2 | 5 | @4 | 1 |
| 10 | @2 | 1 | @4 | 3 | @6 | 5 |

Table 4.2: Schedule after applying SwapHome$(1, 4, 2)$

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 5 | @6 | 4 | @3 | @1 | 2 |
| 2 | 4 | @5 | @6 | @1 | 2 | 3 |
| 3 | 2 | @1 | @5 | @6 | 3 | 4 |
| 4 | 6 | 3 | @2 | @5 | 4 | @1 |
| 5 | 3 | 4 | @1 | @2 | 6 | @5 |
| 6 | @5 | 6 | @4 | 3 | 1 | @2 |
| 7 | @4 | 5 | 6 | 1 | @2 | @3 |
| 8 | @2 | 1 | 5 | 6 | @3 | @4 |
| 9 | @6 | @3 | 2 | 5 | @4 | 1 |
| 10 | @3 | @4 | 1 | 2 | @6 | 5 |

Table 4.3: Schedule after applying SwapTeam$(2, 3)$

The last move used is Game Rotation, which consists of enforcing an arbitrary game to be played in an arbitrary round, followed by the appropriate modifications to avoid teams playing more than one game in the same round. The move is finalized by the application of a fast Tabu search algorithm that throws away possible infeasibilities in the sequence of home and away games generated by the move. The GR neighborhood is explored only as a diversification move performed less frequently by the heuristic due to the high computation costs associated with its investigation.

ST is the first neighborhood explored. Once a local optimum with respect to this neighborhood is found, a quick local search using the SH neighborhood is performed in search of a better venue assignment. Next, the PRS neighborhood is investigated. As before, once a local optimum with respect to this neighborhood is found, the algorithm performs a quick local search using the SH neighborhood in search of a better stadium assignment. This scheme is repeated until a local optimum with respect to all three neighborhoods is found.

## 4.4 Simulated Annealing Approach to TTP

Before discussing the paper by Anagnostopoulos, Michel, Hentenryck, and Vergados [1], we present a basic overview of simulated annealing.

Simulated annealing is a probabilistic local search heuristic for approximately solving an optimization problem, typically used when the search space is large and discrete. Named after a technique used in metallurgy, annealing involves the heating and controlled cooling of a material to alter its physical properties. The central concept of the algorithm is to simulate the evolution of an unstable physical system toward a thermodynamic stable equilibrium point at a fixed temperature. In general, simulated annealing algorithms work as follows.

The temperature is some initial positive value, and progressively decreases to zero. At each iteration, the current candidate solution is replaced by a randomly selected neighboring solution. The algorithm first evaluates the improvement made by the choice of new solution, if the choice of solution leads to a better objective value, it is selected. However, even if no improvement is made, a worse solution can still be accepted according to some probability which depends on the current temperature. Due to the slow cooling implemented in the simulated annealing algorithm, moving to a worse solution is progressively less likely as the solution space is explored. Accepting worse solutions allows for a more extensive search for the global optimal solution.

Simulated annealing can be used for very hard computational optimization problems where exact algorithms fail; even though it usually achieves an approximate solution to the global minimum, it could be enough for many practical problems.

Anagnostopoulos et al. describe a simulated annealing algorithm (TTSA) to solve the TTP(3). The algorithm starts from an initial configuration. Its basic step moves from the current configuration $c$ to a configuration in the neighborhood of $c$. A configuration corresponds to a double-round robin schedule. The algorithm uses four important design features that allow it to produce high-quality solutions:

1. TTSA separates the tournament constraints and the pattern constraints into hard and soft constraints. Hard constraints are those that are always satisfied by the configurations, this includes the round-robin constraints. Soft constraints are those that may or may not be satisfied. These are the no repeaters and the at-most constraints. By dividing the constraints in this manner, all configurations found in the search represents a double round-robin tournament, while they may or may not violate the no repeat and at-most constraints. To steer the search toward feasible

solutions, TTSA modifies the original objective function to include a penalty term for violating soft constraints.

2. TTSA uses a large neighborhood of size $O(n^3)$, where $n$ is the number of teams. The moves defining the search neighborhoods can be rather complex and significantly affect the schedule.

3. TTSA includes a strategic oscillation strategy to balance the time spent in the feasible and infeasible regions.

4. TTSA incorporates the concept of "reheats" to escape from local minima when the algorithm has very low temperatures. The reheats increase the temperature again and divide the search into several phases.

TTSA was applied to the National League instances of the TTP. TTSA matched the best found solutions on the smaller instances (up to eight teams) and improved all the best-known solutions (at the time of the experiments) on instances with at least 10 teams. Their results also show that the worst solution of TTSA is always smaller than or equal to the best-known solution, indicating the robustness of TTSA.

It is computationally intensive to find very high-quality solutions using TTSA. However, in their experiments, TTSA required just 1000 seconds to beat the previously best known results for 12 and 14 teams. This demonstrates the ability of simulated annealing to successfully attack these hard optimization problems with a much simpler machinery compared to other earlier approaches.

The TTSA uses five types of moves, each defining a neighborhood of solutions. The first three moves are simple swap moves, some were defined in the previous section. The last two moves are generalizations of the first three and are partial swap moves.

**SwapHome(A,B,$i$) :** For some game played by $A$ and $B$ on day $i$, this moves swaps the venues of the match-up thus swapping the home/away assignment of these teams. This move only affects this match-up and so it causes the least disturbance to the overall schedule. Using the initial schedule on 6 teams shown in Table 4.1, the move SwapHome$(1, 4, 2)$ is then applied to it. The resulting schedule is shown in Table 4.2 where gray colored cells indicate the games affected by the move.

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 5 | 4 | @6 | @2 | @1 | 3 |
| 2 | 4 | @6 | @5 | @1 | 3 | 2 |
| 3 | @4 | 6 | 5 | 1 | @3 | @2 |
| 4 | 6 | @3 | 2 | @5 | 4 | @1 |
| 5 | 2 | @1 | 4 | @3 | 6 | @5 |
| 6 | @5 | @4 | 6 | 2 | 1 | @3 |
| 7 | 3 | @5 | @1 | @6 | 2 | 4 |
| 8 | @3 | 5 | 1 | 6 | @2 | @4 |
| 9 | @6 | 3 | @2 | 5 | @4 | 1 |
| 10 | @2 | 1 | @4 | 3 | @6 | 5 |

Table 4.4: Schedule after applying SwapRound$(3, 7)$

**SwapRound**$(i, j)$ **:** For two rounds $i$ and $j$, this move swaps all games taking place on day $i$ with games taking place on day $j$, and vice versa. This move affects all teams, but the change is restricted to these two days. Using the initial schedule on 6 teams shown in Table 4.1, the move SwapRound$(3, 7)$ is then applied to it. The resulting schedule is shown in Table 4.4 where gray colored cells indicate the games affected by the move.

**SwapTeam(A,B) :** For two teams $A$ and $B$, this move swaps all opponents of team $A$ with the opponents of team $B$ over all rounds. So if on day $i$ the match-ups are $(A, C)$ and $(B, D)$, then after this move, the match-ups of day $i$ are $(A, D)$ and $(B, C)$. This process is done for every day of the season. Using the initial schedule on 6 teams shown in Table 4.1, the move SwapTeam$(2, 3)$ is then applied to it. The resulting schedule is shown in Table 4.3 where gray colored cells indicate the games affected by the move.

The next two moves are similar in structure but are partial swaps. They significantly enlarge the neighborhood and yield a larger search space.

**PartialSwapRound(A,B,C,D,**$i, j$**) :** For four teams $A, B, C$ and $D$, and two rounds $i$ and $j$ such that games $(A, B)$ and $(C, D)$ take place in round $i$, and games $(A, C)$ and $(B, D)$ take place in round $j$, this move swaps games taking place on day $i$ with games taking place on day $j$, and vice versa. The result is games $(A, C)$ and $(B, D)$ now take place in round $i$ while games $(A, B)$ and $(C, D)$ now take place in round $j$.

**PartialSwapTeam(A,B,$i$) :** For two teams $A$ and $B$, this move swaps all opponents of team $A$ with the opponents of team $B$ but only for round $i$. So if on day $i$ the matchups are $(A, C)$ and $(B, D)$, then after this move, the match-ups of day $i$ are $(A, D)$ and $(B, C)$.

TTSA starts from a random initial schedule and proceeds with the traditional simulated annealing procedure. Given a temperature $T$, the algorithm randomly selects one of the moves in the neighborhood and computes $\Delta$, the change in the objective function produced by the move. If $\Delta < 0$, TTSA applies the move. Otherwise, it applies the move with probability $\exp(\Delta/T)$. Since the temperature decreases over time, the probability of accepting a non-improving move decreases as well. This behavior is obtained by decreasing the temperature as follows. TTSA uses a variable counter which is incremented for each non-improving move and reset to zero when a new best solution is found. When the counter reaches a specified upper limit, the temperature is updated to $T \cdot \beta$ (where $\beta$ is a fixed constant smaller than 1) and the counter is reset to zero. Figure 4.1 includes the pseudo-code of the algorithm.

The schedules considered by the TTSA algorithm may not be feasible since it considers schedules that do not satisfy the no repeaters and at-most constraints. Moreover, even if a solution schedule was initially feasible the moves considered may not maintain the feasibility. To steer the algorithm towards feasible solutions, the objective function cannot simply account for the cost of the schedule, rather it must combine the distance travelled with the number of violated constraints.

The number of violated soft constraints in a schedule $S$ is denoted $nbv(S)$. The new objective function $C$ is given as:

$$C = \begin{cases} cost(S) & \text{if } S \text{ feasible} \\ \sqrt{cost(S)^2 + [w \cdot f(nbv(S))]^2} & \text{otherwise} \end{cases}$$

Where $w$ is a weight, and $f : \mathbb{N} \to \mathbb{N}$ is a sub-linear function such that $f(1) = 1$.

The TTSA makes use of a reheating technique to allow it to escape local minima. Once a very low temperatures is reached in a simulated annealing algorithm, it becomes very difficult for the algorithm to move to a new solution to escape from a local minimum since the probability of accepting non-decreasing moves is very small. Reheating solves this issue by increasing the temperature once again to escape the current local minimum. TTSA uses a relatively simple reheating method: upon completion of the outermost loop, reheat by doubling the value of the temperature when the best solution was found. TTSA now terminates when the number of consecutive reheats without improving the best solution reaches a given limit.

```
find random schedule $S$;
$bestSoFar \leftarrow cost(S)$;
$counter \leftarrow 0$;
**while** $phase \leq maxP$ **do**
    $phase \leftarrow 0$;
    $counter \leftarrow 0$;
    **while** $counter \leq maxC$ **do**
        select a random move $m$ from $neighborhood(S)$;
        let $S'$ be the schedule obtained from $S$ with $m$;
        **if** $cost(S') < cost(S)$ **then**
            $accept \leftarrow$**true**;
        **else**
            $accept \leftarrow$ **true** with probability $\exp(-\Delta/T)$,
                          **false** otherwise;
        **end if**
        **if** $accept$ **then**
            $S \leftarrow S'$;
            **if** $cost(S') < bestSoFar$ **then**
                $counter \leftarrow 0$; $phase \leftarrow 0$;
                $bestSoFar \leftarrow cost(S')$;
            **else**
                $counter$++;
            **end if**
        **end if**
    **end while**
    $phase$++;
    $T \leftarrow T \cdot \beta$;
**end while**
```

Figure 4.1: Simulated Annealing algorithm. Source: [1, Fig. 1].

| *n* | Best (Nov. 2002) | min(D) | max(D) | mean(D) | std(D) |
|---|---|---|---|---|---|
| 8 | 39721 | 39721 | 39721 | 39721 | 0 |
| 10 | 61608 | **59583** | **59806** | **59605.96** | 53.36 |
| 12 | 118955 | **112800** | **114946** | **113853.00** | 467.91 |
| 14 | 205894 | **190368** | **195456** | **192931.86** | 1188.08 |
| 16 | 281660 | **267194** | **280925** | **275015.88** | 2488.02 |

Figure 4.2: Results of TTSA on TTP. Source: [1, Table 3]

The TTSA was applied to the National League instances as defined by Easton, Nemhauser, and Trick [13]. The latest results on these benchmarks, and more, are maintained by Trick on his webpage [31]. For $n = 6$, TTSA always succeeds in finding the optimal solution so this instance is not considered. The most successful version of the algorithm uses a very slowly cooling system ($\beta \simeq .9999$), a large number of phases (so that the system can reach low temperatures), and long phases. To avoid big oscillations in the value of the penalty weight $w$, the parameters $\delta$ and $\theta$ were chosen to be close to 1 ($\simeq 1.03$). For each instance set (i.e., for every value of $n$), in all 50 runs, the parameters had the same initial values. Table 4.2 shows the results of TTSA for different values of $n$ over 50 runs. The second column shows the previously best found results as of November 2002. The next columns record the smallest, largest, and average solution found by TTSA over 50 runs, as well as the standard deviation.

As mentioned earlier, finding high-quality solutions using simulated annealing is computationally expensive. Thus, studying the evolution of solution quality over time can be informative for TTSA. Figure 4.3 shows the values of solutions to 12 team instances with respect to running time (in seconds). The figure features a superposition of curves representing the solution values over many runs.

It is interesting to observe the sharp initial decline in the solution values which is followed by a long tail where improvements are very slow. In particular, TTSA takes about 1,000 seconds to beat the previous best results for 12 teams, after which improvements proceed at a much slower rate. The same phenomenon arises for 14 teams.

Further experiments were run with different parameter values. The impact of different components was also studied to evaluate how important they are to the performance of the algorithm, however we restricted the discussion to the most successful version of TTSA.

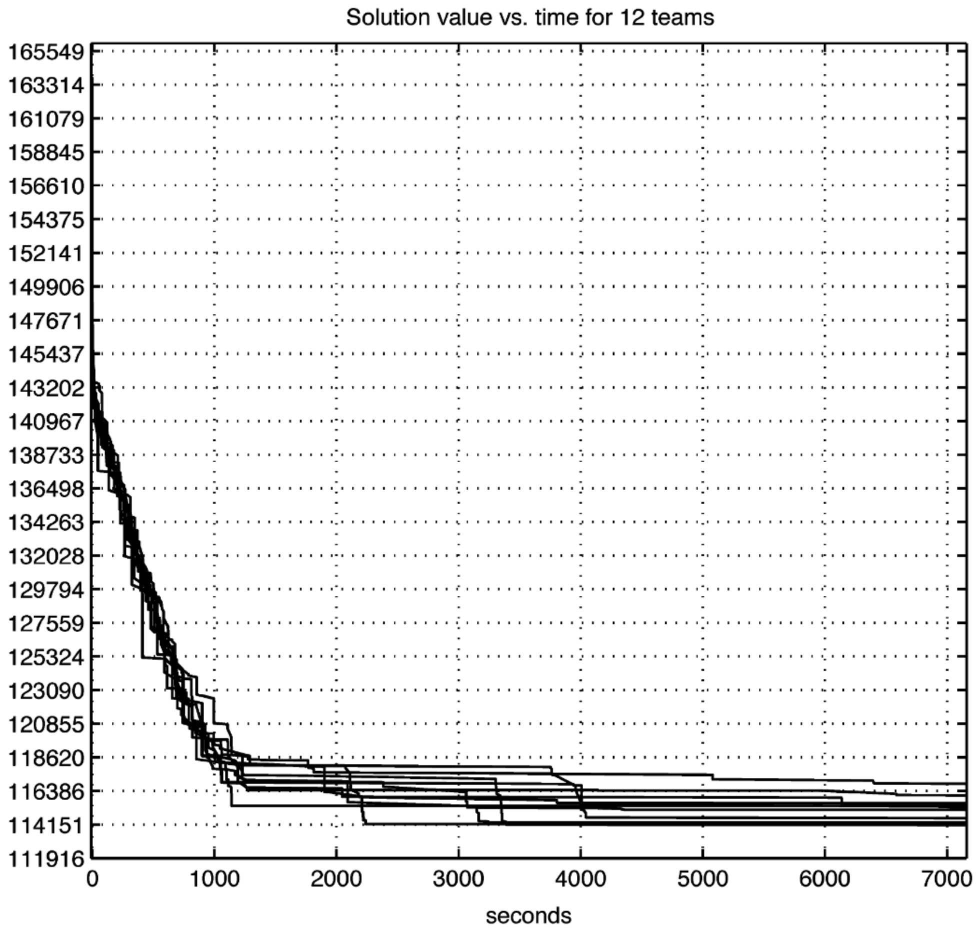


Figure 4.3: Solution quality over time for 12 teams. Source: [1, Fig. 4]

## 4.5 Variable Neighborhood Search Method for solving TTP

Khelifa and Boughaci [22] propose a variable neighborhood search (VNS) method to achieve high-quality solutions to the TTP(3). They construct an initial feasible schedule and then make use of three swap moves to explore the solution space and improve the traveling cost. The moves used in this paper have been defined previously but we present them here for the sake completeness.

**SwapHome(A,B,$i$) :** For some game played by $A$ and $B$ on day $i$, this moves swaps the venues of the match-up thus swapping the home/away assignment of these teams. This move only affects this matchup and so it causes the least disturbance to the overall schedule. The move SwapHome$(1, 4, 2)$ is shown on an example with 6 teams in Table 4.2. The gray colored cells are the games affected by the move.

**SwapRound($i, j$) :** For two rounds $i$ and $j$, this move swaps all games taking place on day $i$ with games taking place on day $j$, and vice versa. This move affects all teams, but the change is restricted to these two days. The move is shown on an example with 6 teams in Table 4.4.

**SwapTeam(A,B,) :** For two teams $A$ and $B$, this move swaps all opponents of team $A$ with the opponents of team $B$ over all rounds. So if on day $i$ the matchups are $(A, C)$ and $(B, D)$, then after this move, the matchups of day $i$ are $(A, D)$ and $(B, C)$. This process is done for every day of the season. The move SwapTeam$(2, 3)$ is shown on an example with 6 teams in Table 4.3. The gray colored cells are the games affected by the move.

Given a feasible schedule, performing any of these three moves will maintain its feasibility.

To generate the initial feasible schedule, a single round-robin tournament is constructed, and then duplicated to yield a double round-robin tournament. The SRR schedule is constructed using the Circle method as follows. Teams are enumerated from 1 to $n$ and match-ups are defined by a pairing based on teams' positions. On day 1, team 1 plays team $n$, team 2 plays team $n - 1$, and so on until the game between team $n/2$ and $n/2 - 1$ is defined. For the second day, we recreate the same pattern of pairings, but rotated clockwise. An example of this procedure with 6 teams is shown in Figure 4.4. The pairing of teams yields a single round-robin schedule. To produce a double round-robin schedule, we mirror the SRR schedule constructed and combine the two halves to yield a DRR tournament.

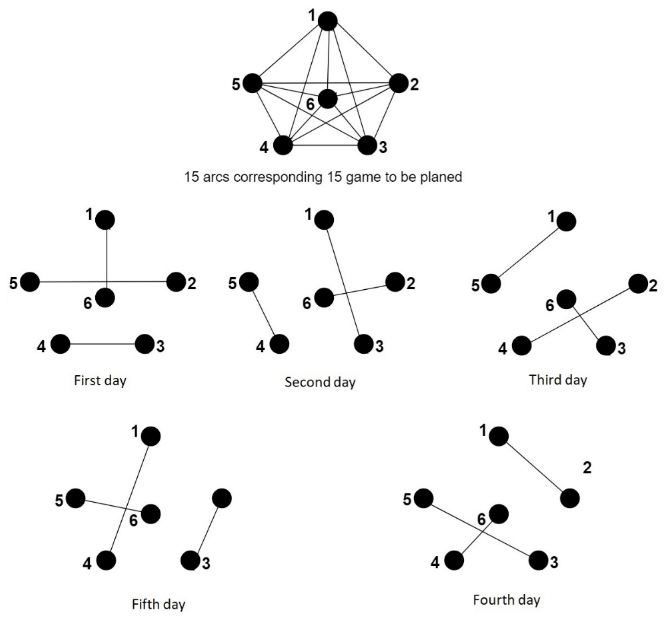


Figure 4.4: Generating the initial configuration. Source: [22]

While the combination of the initial SRR schedule with its mirror does yield a mirrored DRR schedule, it may not be feasible due to violating the at-most constraint. To remedy this, a simulated annealing algorithm is run on the DRR schedule but the only move available in the procedure is SwapRound. By penalizing each violation of the at-most constraints, the output of the simulated annealing algorithm is a feasible DRR schedule that satisfies the at-most and no repeaters constraints. The output schedule is then fed to a Variable Neighborhood Search algorithm where each move consists of a neighborhood. Each neighborhood is explored until a local optimum was found, at which point the next neighborhood is searched. Figure 4.5 shows a sketch of the VNS algorithm.

The algorithm was run on the popular benchmark instances including the National League ($NL_x$), circular ($CIRC_x$) and constant distance ($CON_x$) instances. Figure 4.6 contains the results of the VNS method.

The last two columns of the table contain the previously best-known results, as well as the techniques used to find these results. The clustering method results are due to

**Algorithm 2** : The VNS method for TTP.

```
Require: a TTP instance, three Neighborhood structures are used $V_k$:, $k = 1$ for Swap Round structure,
    $k = 2$: for Swap Home and $k = 3$: for Swap Team, $Maxiter$ is the maximum number of iterations.
Ensure: Best schedule $S$ for TTP
 1: Create an initial configuration (CS) verifying the DRRT constraint
 2: Apply SA on CS to obtain a CSC configuration verifying the three constraints
 3: $k \leftarrow 1$
 4: Create a list (list) of movements $(R_i, R_j)$ where the cost(S) is equals to zero by using the aspiration
    technique
 5: start from a random initial solution (CSC)
 6: $S_0 \leftarrow$ SwapRound (S,$R_i$, $R_j$)
 7: $S^* \leftarrow S_0$
 8: (Apply SLS with the neighborhood structure ($N_1$) on ($S_0$) to obtain $S$).
 9: for $I = 1$ to $maxiter$ do
10:     $S^{'} \leftarrow$ choose Random solution $\in N_k(S)$
11:     $S^{"} \leftarrow$ local search
12:     Apply SLS with the neighborhood structure ($N_k$) on ($S^{'}$) to obtain $S^{"}$.
13:     if $f(S^{"}) < f(S)$ then
14:         $S \leftarrow S^{"}$
15:         $S^* \leftarrow S$
16:     else
17:         if $k < |K|$ then
18:             $k \leftarrow k + 1$
19:         else
20:             $k \leftarrow 1$
21:         end if
22:     end if
23: end for
24: return the best schedule $S$ found.
```

Figure 4.5: The VNS algorithm for TTP. Source: [22]

[3] while the AIS results are due to [4]. The VNS algorithm reaches optimality for the constant distance instances up to 14 teams. The algorithm does not reach optimality on the National League instance for more that 6 teams, however we see substantial improvement over the previous best results. In fact, with the exception of the $NL4$ and $NL6$ instances, the average result over multiple runs of the VNS algorithm outperforms the previous best methods. This indicates the robustness of the VNS algorithm and its viability for consistently producing high-quality solutions.

| INS | lower bound | **VNS** | | | **Others** | |
|---|---|---|---|---|---|---|
| | | Time | Best | AVG | Best | Techniques |
| CON4 | **17** | 0.8 | **17** | 17 | **17** | Clustering |
| CON6 | **43** | 22.32 | **43** | 43 | 48 | Clustering |
| CON8 | **80** | 26.48 | **80** | 80 | **80** | Clustering |
| CON10 | **124** | 94.33 | **124** | 125 | 130 | Clustering |
| CON12 | **182** | 132.56 | **182** | 184 | 192 | Clustering |
| CON14 | **252** | 323.97 | **252** | 254 | 256 | Clustering |
| CON16 | **327** | 512.65 | 338 | 342 | 342 | Clustering |
| NL4 | **8276** | 42.15 | **8276** | 8276 | **8276** | Clustering |
| NL6 | **23916** | 133.01 | **23916** | 26588 | 24073 | Clustering |
| NL8 | **39947** | 484.34 | 41505 | 43112 | 42517 | Clustering |
| NL10 | **59583** | 831,24 | 60293 | 63832 | 68691 | AIS |
| NL12 | **111248** | 1229.63 | 120696 | 120906 | 143655 | AIS |
| NL14 | **188728** | 1957.58 | 207343 | 208086 | 301113 | AIS |
| NL16 | **261687** | 4255.61 | 285614 | 293645 | 346530 | Clustering |
| CIRC6 | **64** | 244.48 | **64** | 64 | **64** | Clustering |
| CIRC8 | **130** | 415.22 | 140 | 142 | 148 | Clustering |
| CIRC10 | **242** | 731.41 | 266 | 276 | 288 | Clustering |

Figure 4.6: Results of VNS on benchmark instances. Source [22, Table 2]. The clustering results are due to [3], AIS results are due to [4]

# Chapter 5

# Conclusions

In this thesis we studied the Traveling Tournament problem (TTP). The TTP was introduced with a parameter $k$ which is the maximum number of consecutive home/away games a team can play. We used the notation TTP($k$) to specify which version of the problem was being discussed. Throughout our three technical chapters a wide range of topics connected to the TTP were explored. We began by considering the computational complexity of the problem. Despite existing results on the NP-hardness of TTP, the question of whether or not TTP is also APX-hard was an unexplored area in the literature. Our result proving the affirmative is therefore of significant interest. The next chapter delves into a popular variation of the problem, the mirrored TTP (mTTP). Building upon previous techniques, we proposed the first approximation algorithm for solving mTTP(2), and proved an approximation ratio of $3/2 + O(1)/n$. Lastly, we presented a survey of local search methods for solving TTP and discussed the performance of these techniques on benchmark instances.

Many approximation algorithms have been proposed for TTP, thus determining whether the problem is APX-hard is of interest to the research community. To the best of our knowledge, no such result appears in the literature. Inspired by Bhattacharyya's [2] proof of NP hardness of the unconstrained TTP, we built on this result to show the problem is also APX hard. In Chapter 2 we presented an $L$-reduction from $(1, 2)$-TSP to TTP. To reach the desired result, we showed that given an instance of TSP with a solution of cost $K$, we can construct an instance of TTP with a solution of cost at most $20m(m + 1)cK$ where $m = c(n - 1) + 1$, $n$ is the number of teams, and $c > 5, c \in \mathbb{Z}$ is fixed.. On the other hand, we showed that given a feasible schedule to the constructed TTP instance, we can recover a tour on the original TSP instance.

In Chapter 3 we considered the mirrored Traveling Tournament problem (mTTP), an important variant of the original TTP which has seen considerable research progress. Unlike the TTP, which has been studied for different values of $k$, the mTTP has primarily been explored for $k = 3$. This led us to consider extending the existing approximation algorithm for TTP(2), such that we can produce mirrored schedules, thus solving mTTP(2).

The most common methods of constructing mirrored schedules rely on our ability to find good single round-robin tournaments hence the relationship between a single round-robin (SRR) tournament and a double round-robin (DRR) tournament was studied. Several new results were presented including relationships between the optimal value of a SRR schedule to an optimal DRR schedule, as well as the value of an optimal SRR schedule compared to that of an optimal mirrored DRR tournament. The key result of the chapter was the presentation of the first constructive algorithm for building a mirrored DRR schedule when the number of consecutive home or away games is restricted to two. The approximation guarantee of the algorithm is shown to be on the order of $3/2 + O(1)/n$.

The final chapter contains a survey of several papers which make use of local search methods to solve TTP. We discussed several heuristic methods which achieve good results on benchmark instances of TTP. Three papers were discussed in more detail. Due to our interest in the mirrored TTP, we studied the work of Ribeiro and Urrutia [27] who first introduced the problem and proposed a greedy iterative search method to find high-quality solutions to the mirrored problem. We then explored one of the first examples of heuristic methods being applied to TTP which was a simulated annealing technique by Anagnostopoulos et al. [1] to solve TTP(3). Finally we looked at a more recent development, a Variable Neighborhood search technique proposed by Khelifa and Boughaci [22] to solve TTP(3).

## Further Research Directions

Over the course of researching the TTP, several interesting research directions came up. Below are several open questions, as well as results that can potentially be improved upon.

1. Prove NP-hardness of TTP(2).
2. Prove APX-hardness of TTP($k$) for fixed values of $k$.
3. Improve the approximation ratio of mTTP($k$) for $k > 2$. Yamaguchi et al. [35] presented approximation ratios for mTTP($k$) for all fixed $k > 2$. When $k \leq 5$, the approximation ratio of the proposed algorithm is bounded by $(2k - 1)/k + O(k/n)$, when $k > 5$, the ratio is bounded by $(5k - 7)/(2k) + O(k/n)$.

# References

[1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. *J. Sched.*, 9(2):177–193, 04 2006.

[2] Rishiraj Bhattacharyya. Complexity of the unconstrained traveling tournament problem. *Operations Research Letters*, 44(5):649–654, 2016.

[3] Fabrício Lacerda Biajoli and Luiz Antonio Nogueira Lorena. Clustering search approach for the traveling tournament problem. In Alexander Gelbukh and Ángel Fernando Kuri Morales, editors, *MICAI 2007: Advances in Artificial Intelligence*, pages 83–93, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[4] Leslie Pérez Cáceres and Maeía Cristina Riff. Aisttp: An artificial immune algorithm to solve traveling tournament problems. *International Journal of Computational Intelligence and Applications*, 11(01):1250008, 2012.

[5] Robert Thomas Campbell and DS Chen. A minimum distance basketball scheduling problem. *Management science in sports*, 4:15–26, 1976.

[6] Diptendu Chatterjee. Complexity of traveling tournament problem with trip length more than three, 2021. arXiv:2110.02300.

[7] Diptendu Chatterjee and Bimal Kumar Roy. An improved scheduling algorithm for traveling tournament problem with maximum trip length two. 2021. arXiv:2109.09065.

[8] Kevin K.H. Cheung. Solving mirrored traveling tournament problem benchmark instances with eight teams. *Discrete Optimization*, 5(1):138–143, 2008.

[9] Kevin K.H. Cheung. A benders approach for computing lower bounds for the mirrored traveling tournament problem. *Discrete Optimization*, 6(2):189–196, 2009.

[10] D. de Werra. Scheduling in sports. In P. Hansen, editor, *Annals of Discrete Mathematics (11)*, volume 59 of *North-Holland Mathematics Studies*, pages 381–395. North-Holland, 1981.

[11] D. de Werra. Some models of graphs for scheduling sports competitions. *Discrete Applied Mathematics*, 21(1):47–65, 1988.

[12] Alexandre Duarte, Celso Ribeiro, Sebastián Urrutia, and Edward Haeusler. Referee assignment in sports leagues. *Lecture Notes in Computer Science*, 3867:158–173, 01 2006.

[13] Kelly Easton, George Nemhauser, and Michael Trick. The traveling tournament problem description and benchmarks. In Toby Walsh, editor, *Principles and Practice of Constraint Programming — CP 2001*, pages 580–584, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[14] Kelly Easton, George Nemhauser, and Michael Trick. Solving the Travelling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach. In Edmund Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, pages 100–109, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[15] Nobutomo Fujiwara, Shinji Imahori, Tomomi Matsui, and Ryuhei Miyashiro. Constructive algorithms for the constant distance traveling tournament problem. PATAT'06, page 135–147, Berlin, Heidelberg, 2006. Springer-Verlag.

[16] Richard Hoshino and Ken-ichi Kawarabayashi. The inter-league extension of the traveling tournament problem and its application to sports scheduling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 25(1):977–984, 08 2011.

[17] Richard Hoshino and Ken-ichi Kawarabayashi. An approximation algorithm for the bipartite traveling tournament problem. *Mathematics of Operations Research*, 38(4):720–728, 2013.

[18] Shinji Imahori. A $1 + O(1/n)$ approximation algorithm for TTP(2), 2021. arXiv:2108.08444.

[19] Shinji Imahori, Tomomi Matsui, and Ryuhei Miyashiro. A 2.75-approximation algorithm for the unconstrained traveling tournament problem. *Annals of Operations Research*, 218, 10 2011.

[20] Shinji Imahori, Ryuhei Miyashiro, and Tomomi Matsui. An approximation algorithm for the traveling tournament problem. *Annals of Operations Research*, 194:317–324, 2012.

[21] Graham Kendall, Sigrid Knust, Celso C. Ribeiro, and Sebastián Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1):1–19, 2010.

[22] Meriem Khelifa and Dalila Boughaci. A variable neighborhood search method for solving the traveling tournaments problem. *Electronic Notes in Discrete Mathematics*, 47:157–164, 2015. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[23] Meriem Khelifa, Dalila Boughaci, and Esma Aïmeur. *Evolutionary Harmony Search Algorithm for Sport Scheduling Problem*, pages 93–117. Springer International Publishing, Cham, 2018.

[24] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[25] Yun-Chia Liang, Yen-Yu Lin, Angela Hsiang-Ling Chen, and Wei-Sheng Chen. Variable neighborhood search for major league baseball scheduling problem. *Sustainability*, 13(7), 2021.

[26] Celso C. Ribeiro and Sebastián Urrutia. Scheduling the Brazilian Soccer Tournament with Fairness and Broadcast Objectives. In Edmund K. Burke and Hana Rudová, editors, *Practice and Theory of Automated Timetabling VI*, pages 147–157, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[27] Celso C. Ribeiro and Sebastián Urrutia. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 179(3):775–787, 2007.

[28] Uwe Schauz. The tournament scheduling problem with absences. *European Journal of Operational Research*, 254(3):746–754, 2016.

[29] Clemens Thielen and Stephan Westphal. Complexity of the traveling tournament problem. *Theor. Comput. Sci.*, 412:345–351, 02 2011.

[30] Clemens Thielen and Stephan Westphal. Approximation algorithms for TTP(2). *Mathematical Methods of Operations Research*, 76:1–20, 2012.

[31] Michael Trick. Challenge traveling tournament instances. https://mat.tepper.cmu.edu/TOURN/. Accessed: 2022-05-12.

[32] Sebastián Urrutia and Celso C. Ribeiro. Maximizing breaks and bounding solutions to the mirrored traveling tournament problem. *Discrete Applied Mathematics*, 154(13):1932–1938, 2006. Traces of the Latin American Conference on Combinatorics, Graphs and Applications.

[33] Stephan Westphal and Karl Noparlik. A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, 218:1–14, 01 2010.

[34] Mingyu Xiao and Shaowei Kou. An Improved Approximation Algorithm for the Traveling Tournament Problem with Maximum Trip Length Two. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 89:1–89:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[35] Daisuke Yamaguchi, Shinji Imahori, Ryuhei Miyashiro, and Tomomi Matsui. An improved approximation algorithm for the traveling tournament problem. *Algorithmica*, 61:1077–1091, 01 2011.

[36] Jingyang Zhao and Mingyu Xiao. The traveling tournament problem with maximum tour length two: A practical algorithm with an improved approximation bound. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4206–4212. International Joint Conferences on Artificial Intelligence Organization, 08 2021.