A Matching-Based Algorithm for the Traveling Tournament Problem

Jingyang Zhao, Mingyu Xiao*

University of Electronic Science and Technology of China
jingyangzhao1020@gmail.com, myxiao@gmail.com

Abstract

The Traveling Tournament Problem (TTP-k) is a well-known benchmark problem in tournament timetabling. It involves designing a feasible double round-robin tournament for a sports league of n teams under several feasibility requirements, while minimizing the total traveling costs of the teams. The parameter k requires that in the tournament at most k consecutive home games or away games for each team are allowed. TTP-k with a small k, especially for $k = 2, 3$ and 4, have been extensively studied in the literature. In this paper, we focus on TTP-4 and design an efficient algorithm for it based on minimum weight matching. In theory, we prove that our algorithm has an approximation ratio of $1.625+\varepsilon$ for any constant $\varepsilon > 0$, improving the best-known approximation ratio of $1.7 + \varepsilon$. In practice, our experimental results indicate an average improvement of 6.65% over the best-known solutions on 9 benchmark instances.

Introduction

The Traveling Tournament Problem (TTP-k) is a combinatorial optimization problem that combines elements of the Traveling Salesman Problem (TSP) and sports scheduling. In TTP-k, we are going to design a double round-robin tournament under some additional feasibility constraints, aiming to minimize the total distance traveled by all teams. TTP-k was initially introduced in (Easton, Nemhauser, and Trick 2001) for scheduling Major League Baseball games in real life. Later, it finds practical applications in diverse fields such as professional sports leagues, international tournaments, college sports, youth leagues, and more. These illustrate how TTP-k is not just a theoretical concept but also a practical tool used at various levels of sports scheduling to enhance the overall experience for players and organizers.

Next, we give the definition of TTP-k. In a double round-robin tournament, a set of n teams must play against each other twice, one home game at its own venue and one away game at its opponent's venue, in $2(n-1)$ consecutive days, with each team playing exactly one game per day. TTP-k is to design a double round-robin tournament satisfying the following two constraints, minimizing the total distance traveled by all teams.

- *No-repeat*: No pair of teams play against each other in two consecutive games.
- *At-most-k*: Each team plays at most k consecutive home games or away games.

When calculating the traveling distance, we assume that each team begins at his home venue before the first game and returns home after the last game; for two consecutive days, each team travels directly from the first game venue to the second game venue; the distances are metric.

Although TTP-k is an interesting problem with many applications, it is challenging in terms of computation. Due to the feasibility constraints, it is not even easy to find a feasible solution (Verduin, Thomson, and van den Berg 2024). In the online benchmark website (Trick 2024), the optimal solutions to most instances with 12 teams are not found yet even computed by high-performance computers for a long time. On the other hand, efficient heuristic and approximation algorithms play a pivotal role in addressing the complexities of TTP-k by devising optimal schedules that meet the constraints and objectives of the problem. In the literature, these algorithms leverage techniques from combinatorial optimization, graph theory, constraint programming, and other computational methods to generate schedules that are not only practical but also exhibit qualities of fairness, balance, and cost-effectiveness.

In this paper, we focus on TTP-4, where at most 4 consecutive home games or away games for each team are allowed. We will design a practical algorithm with a good theoretical guarantee on approximation quality.

Related Work

The NP-hardness of TTP-k was established in several papers (Thielen and Westphal 2011; Chatterjee 2021; Bhattacharyya 2016). To solve TTP-k, many heuristic algorithms have been proposed, such as algorithms based on combinatorial methods (Easton, Nemhauser, and Trick 2002; Lim, Rodrigues, and Zhang 2006), simulated annealing (Anagnostopoulos et al. 2006; Hentenryck and Vergados 2007), tabu search (Di Gaspero and Schaerf 2007), integer programming (Goerigk and Westphal 2016), and beam search (Frohner et al. 2023).

In order to guarantee the solution quality, we also work on approximation algorithms (Thielen and Westphal 2012;

*Corresponding author

Xiao and Kou 2016; Chatterjee and Roy 2021; Miyashiro, Matsui, and Imahori 2012; Yamaguchi et al. 2011; Westphal and Noparlik 2014; Hoshino and Kawarabayashi 2012, 2013). Approximation solutions can then be further refined into high-quality solutions by combining them with heuristic methods (Thielen and Westphal 2012; Westphal and Noparlik 2014; Goerigk et al. 2014). Currently, the best-known approximation ratios TTP-k are as follows: $1+\varepsilon$ for $k=2$ (Zhao and Xiao 2021; Imahori 2021), where $\varepsilon>0$ is any positive constant; $1.598+\varepsilon$ for $k=3$ (Zhao, Xiao, and Xu 2022); $1.7+\varepsilon$ for $k=4$ (Zhao, Xiao, and Xu 2022); and 2.75 for $k \geq n-1$ (Imahori, Matsui, and Miyashiro 2014).

More surveys on TTP-k and its variants can be found in (Bulck et al. 2020; Durán 2021).

Our Results

In this paper, we propose a matching-based algorithm for TTP-4. First, we prove that our algorithm can guarantee an approximation ratio of $1.625+\varepsilon$, improving the best-known ratio of $1.7+\varepsilon$ (Zhao, Xiao, and Xu 2022). Second, the algorithm is also efficient in practice. Experimental results show that we can improve the best-known results (Westphal and Noparlik 2014) for all 9 benchmark instances with $n \equiv 0 \pmod 8$, achieving an average improvement ratio of 6.65%.

To reach this result, we utilize a construction based on minimum weight matching, ensuring that every team frequently plays 4 consecutive away games along two matching edges. Additionally, we introduce two new lower bounds related to the matching and a randomized labeling algorithm. These components allow us to analyze the approximation ratio of our algorithm easily.

Due to limited space, the proofs of lemmas and theorems marked with "*" were omitted and they can be found in the full version of this paper.

Preliminaries

Let $G=(V=\{v_1,...,v_n\},E)$ be the input complete graph, where vertices in V represent the n participating teams and n is always even. There is a non-negative weight function $w: E \to \mathbb{R}_{\geq 0}$ on the edges in E. For any edge $ab \in E$, we use $w(a,b)$ to denote its weight, which denotes the distance between the home venues of teams a and b. The function $w(\cdot)$ is a *metric* function satisfying the properties of symmetry and the triangle inequality. Specifically, for three vertices $a,b,c \in V$, it holds that $w(a,b)+w(b,c) \geq w(a,c) = w(c,a)$. We also extend the weight function w to a set of edges, i.e., for any $E' \subseteq E$, let $w(E')=\sum_{e\in E'} w(e)$. For any subgraph G' of G, let $V(G')$ and $E(G')$ denote the vertex set and edge set of G'. For any vertex set $V' \subseteq V$, the complete graph induced by V' is denoted by $G[V']$.

For any $v \in V$, let $E(v)=\{uv \mid uv \in E\}$ denote the set of all edges incident on v in G. The *weighted degree* of v is defined as $\delta(v)=w(E(v))$. We define $\delta(V)$ as the sum of the weighted degrees of all vertices, i.e.,

$$\delta(V)=\sum_{v\in V}\delta(v)=2w(E). \tag{1}$$

A *walk* W is a sequence of vertices where each consecutive pair of vertices is connected by an edge. We use $w(W)$ to denote the total weight of the edges traversed in the sequence. A walk is *closed* if the first and last vertices are the same. A *path* is a walk with no repeated vertices. A *cycle* is a walk where only the first and the last vertices are the same. An i-path $v_1v_2\dots v_i$ is a path on i different vertices. It consists of $i-1$ edges $\{v_1v_2,\dots,v_{i-1}v_i\}$, and its *length* is said to be i. An i-cycle $v_1v_2\dots v_iv_1$ is a cycle on i different vertices. It consists of i edges $\{v_1v_2,\dots,v_iv_1\}$, and its *length* is also said to be i. Given a walk W, we can skip several vertices along the walk to obtain a new walk W', and this operation is called *shortcutting*. By the triangle inequality, it holds that $w(W') \leq w(W)$. If only a specific vertex v is consistently skipped and no other vertices are skipped, we refer to this operation as *shortcutting* v.

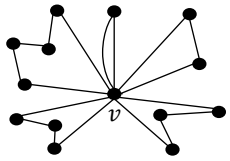


Figure 1: An itinerary of team v, which contains five road trips, and each road trip is an i-cycle satisfying $2 \leq i \leq 5$

A *matching* in G is a set of $n/2$ vertex-disjoint edges. We let M^* denote the minimum weight matching in G, which can be found in $O(n^3)$ time (Gabow 1974; Lawler 1976). An edge in M^* is referred to as a *matching edge*, while an edge in $E \setminus M^*$ is referred to as a *non-matching edge*. A *Hamiltonian cycle* in G is a simple cycle on n different vertices.

In any feasible solution to TTP-4, every team $v \in V$ follows an *itinerary*, represented as a closed walk starting and ending at v, and it contains every other vertex exactly once. This itinerary can be split into several minimal closed walks, each starting and ending at v, referred to as *road trips*. Since each team plays at most 4 consecutive away games in TTP-4, each road trip of v is an i-cycle containing v with $2 \leq i \leq 5$. Moreover, any pair of its road trips share only one common vertex v. See Figure 1 for an illustration. For any road trip of v, we refer to the edges incident to v as *home edges*, and the other edges as *away edges*. Clearly, each road trip contains exactly two home edges.

Fix an optimal solution of TTP-4. For any team $v \in V$, we let I_v^* denote the itinerary of v. Hence, the optimal solution has a weight of

$$\text{OPT}=\sum_{v\in V} w(I_v^*). \tag{2}$$

For any pair of teams $u,v \in V$, we use $u \to v$ or $v \leftarrow u$ to denote a game between u and v at home venue of v, and use $u \leftrightarrow v$ to denote games $u \to v$ and $u \leftarrow v$. Hence, in a double round-robin tournament, we need to schedule all games in $\{u \leftrightarrow v \mid u,v \in V\}$.

The Matching-Based Construction

Our construction is based on minimum weight matching M^* in G. Assume w.l.o.g. $M^*=\{e_1,...,e_{n/2}\}$, where $e_i = v_{2i-1}v_{2i}$ for each $i \in \{1,...,n/2\}$. To get a good schedule, we aim to ensure that

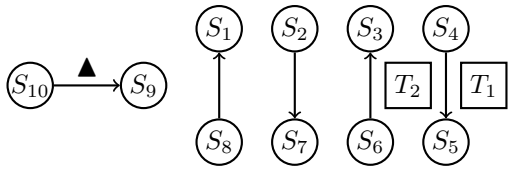


Figure 2: The super-tournament in the first time slot, where $m = 10$ and $r = 2$

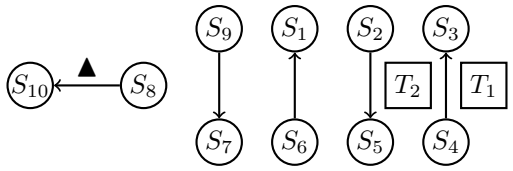


Figure 3: The super-tournament in the second time slot, where $m = 10$ and $r = 2$

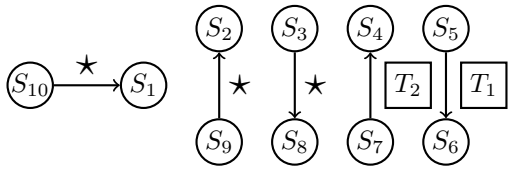


Figure 4: The super-game schedule in the last time slot, where $m = 10$ and $r = 2$

- For each team, almost all of its road trips are 5-cycles, i.e., it mainly plays 4 consecutive away games;
- Almost all 5-cycles contain two matching edges.

To achieve this, we use the *packing-and-extending* technique for TTP-k (Hoshino and Kawarabayashi 2012; Goerigk et al. 2014; Zhao and Xiao 2023). The idea is to pack some teams as a *super-team* according to some graph structures, then schedule a *super-tournament* between super-teams, and finally extend the super-tournament into a feasible schedule for TTP-k. For TTP-3, some algorithms pack three teams on a 3-path (Goerigk et al. 2014; Zhao and Xiao 2023) as a super-team. For TTP-4, the best-known 1.7-approximation algorithm (Zhao, Xiao, and Xu 2022) packs four teams on a 4-cycle as a super-team. In our construction, we pack four teams on two edges in M^* as a super-team. We will show that our method is better than the previous method based on packing 4-cycles and furthermore our analysis is simple.

Let $m = 2\lfloor n/8 \rfloor$. We form m super-teams by packing four teams on two edges in M^* as a super-team, specifically from e_1 to e_{2m}. These m super-teams are denoted by $\{S_1, \ldots, S_m\}$, where $S_i = \{e_{2i-1}, e_{2i}\}$ and S_i consists of the four teams $\{v_{4i-3}, v_{4i-2}, v_{4i-1}, v_{4i}\}$ associated with the edges e_{2i-1} and e_{2i}. Let $r = n/2 - 2m$. The r pairs of teams on the r edges $\{e_{2m+1}, \ldots, e_{2m+r}\}$ are not included in any super-team. We denote each pair of teams by a *team-pair* T_i, where $T_i = \{e_{2m+i}\}$ consists of two teams on e_{2m+i}.

Remark 1. *Since n is even, we have $r = n/2 - 4\lfloor n/8 \rfloor \in \{0, 1, 2, 3\}$. If $n \equiv 0 \pmod 8$, there are no team-pairs, and thus this case is the simplest case. We will mainly focus on this simplest case and then extend to other cases.*

Next, we present our super-tournament.

The Super-Tournament

Our super-tournament comprises $m-1$ *time slots*, with each time slot featuring $m/2$ *super-games* between super-teams. Each super-team (denoted by a cycle) and each team-pair (denoted by a square) are represented visually. A directed edge between two super-teams signifies a super-game played at the home venue of the receiving super-team. There are four types of super-games:

- $S_i \to S_j$: Normal super-game.
- $S_i \xrightarrow{\blacktriangle} S_j$: Left super-game.
- $S_i \xrightarrow{T_{i'}} S_j$: Right super-game.
- $S_i \xrightarrow{\star} S_j$: Last super-game.

In the 1st time slot, the super-games are scheduled as depicted in Figure 2. Specifically, there is 1 left super-game, $m/2 - 1 - r$ normal super-games, and r right super-games.

In the 2nd time slot, super-games are scheduled as shown in Figure 3. We maintain the positions of the last super-team and the r team-pairs, while shifting the positions of the other super-teams in the cycle $S_1 S_2 \ldots S_{m-1} S_1$ by one position clockwise. Additionally, we reverse the direction of each edge, but the types of super-games are unchanged. The super-tournament for the first $m - 2$ time slots is derived analogously.

In the last time slot, super-games are scheduled as shown in Figure 4. The only difference is that the $m/2 - r$ super-games all become last super-games.

In summary, each of the first $m - 2$ time slot includes 1 left super-game, $m/2 - r - 1$ normal super-games, and r right super-games. In the last time slot, there are $m/2 - r$ last super-games and r right super-games. Each left, normal, or last super-game involves only two super-teams, while each right super-game involves two super-teams and one team-pair. Note that we need to ensure $m/2 - r - 1 \geq 0$, and thus, we assume $n \geq 32$ for the sake of presentation.

Next, we show how to extend super-games into games played between individual teams. This extension transforms the super-tournament into an incomplete solution for TTP-4. There may still be some unscheduled games, which will be addressed and scheduled after the final time slot.

Extending Super-Games

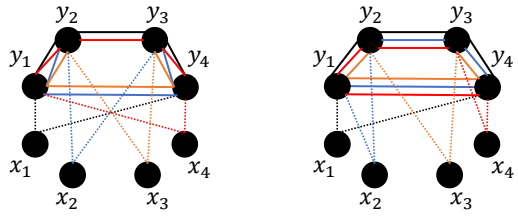
We present the extension of the four types of super-games.

Normal super-games: $S_i \to S_j$. For the sake of presentation, we relabel $S_i = \{x_1x_2, x_3x_4\}$ and $S_j = \{y_1y_2, y_3y_4\}$. We extend the normal super-game into games on 8 days, as shown in Table 1, where games in $\{x \leftrightarrow y \mid x \in S_i, y \in S_j\}$ are all scheduled in extending the normal super-game.

Since almost all super-games in the super-tournament are normal super-games, the design of normal super-games is the most important part. We show several good properties of our normal super-games.

	1	2	3	4	5	6	7	8
x_1	y_1	y_2	y_3	y_4	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$
x_2	y_2	y_1	y_4	y_3	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$
x_3	y_3	y_4	y_1	y_2	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$
x_4	y_4	y_3	y_2	y_1	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$
y_1	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	x_1	x_2	x_3	x_4
y_2	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_2	x_1	x_4	x_3
y_3	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	x_3	x_4	x_1	x_2
y_4	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_4	x_3	x_2	x_1

Table 1: Extending the normal super-game from super-teams $S_i = \{x_1x_2, x_3x_4\}$ to $S_j = \{y_1y_2, y_3y_4\}$ into games on 8 days, where home games are marked in bold



(a) Based on matchings (b) Based on 4-cycles

Figure 5: The road trips of $x_1, ..., x_4$ in two kinds of normal super-games from $\{x_1x_2, x_3x_4\}$ to $\{y_1y_2, y_3y_4\}$, where we use dotted edges and solid edges to denote the home edges and away edges: Figure (a) represents our super-games based on matchings, while Figure (b) represents the previous super-games based on 4-cycles

From Table 1, we know that every team in S_i (resp., S_j) will play 4 consecutive away games (resp., home games) and 4 consecutive home games (resp., away games). Clearly, every team incurs a single road trip, which is a 5-cycle. More importantly, since $x_1x_2, x_3x_4, y_1y_2, y_3y_4$ are all matching edges, each team's road trip includes two matching edges as away edges. For example, we consider the road trips of teams in S_i (the road trips of teams in S_j are similar). These four road trips include 12 away edges, with y_1y_2 and y_3y_4 appearing four times each, and y_2y_3 and y_1y_4 appearing two times each (see (a) in Figure 5). These away edges will be included again in other normal super-games involving S_j. Since y_1y_2 and y_3y_4 are matching edges, the cost of our normal super-games should be positive.

We remark that the 1.7-approximation algorithm (Zhao, Xiao, and Xu 2022) is based on 4-cycles, where it first computes a set of vertex-disjoint 4-cycles in G and then packs the four teams on each 4-cycle as a super-team. In their normal super-games, the road trips of teams in S_i also have 12 away edges, but they include each edge on the 4-cycle $y_1y_2y_3y_4y_1$ three times without variation (see (b) in Figure 5). Assume $A = w(y_1, y_2) + w(y_3, y_4) \leq B = w(y_2, y_3) + w(y_1, y_4)$. Their cost on away edges is $3A + 3B$, which is greater than or equal to ours cost, i.e., $4A + 2B$.

Left super-games: $S_i \xrightarrow{\blacktriangle} S_j$**.** Similarly, we relabel $S_i = \{x_1x_2, x_3x_4\}$ and $S_j = \{y_1y_2, y_3y_4\}$. We extend the left super-game into games on 8 days, as shown in Table 2,

	1	2	3	4	5	6	7	8
x_1	y_1	y_2	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	y_3	y_4
x_2	y_2	y_1	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	y_4	y_3
x_3	y_3	y_4	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	y_1	y_2
x_4	y_4	y_3	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$	y_2	y_1
y_1	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	x_3	x_4	x_1	x_2	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$
y_2	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_4	x_3	x_2	x_1	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$
y_3	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	x_1	x_2	x_3	x_4	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$
y_4	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_2	x_1	x_4	x_3	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$

Table 2: Extending the left super-game from super-teams $S_i = \{x_1x_2, x_3x_4\}$ to $S_j = \{y_1y_2, y_3y_4\}$ into games on 8 days, where home normal games are marked in bold

	1	2	3	4	5	6	7	8
x_1	y_1	y_5	$\boldsymbol{y_3}$	$\boldsymbol{y_2}$	y_3	y_2	$\boldsymbol{y_1}$	$\boldsymbol{y_5}$
x_2	y_5	y_4	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	y_2	y_1	$\boldsymbol{y_5}$	$\boldsymbol{y_4}$
x_3	y_4	y_3	$\boldsymbol{y_1}$	$\boldsymbol{y_5}$	y_1	y_5	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$
x_4	y_3	y_2	$\boldsymbol{y_5}$	$\boldsymbol{y_4}$	y_5	y_4	$\boldsymbol{y_3}$	$\boldsymbol{y_2}$
x_5	y_2	y_1	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$	y_4	y_3	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$
y_1	$\boldsymbol{x_1}$	$\boldsymbol{x_5}$	x_3	x_2	$\boldsymbol{x_3}$	$\boldsymbol{x_2}$	x_1	x_5
y_2	$\boldsymbol{x_5}$	$\boldsymbol{x_4}$	x_2	x_1	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_5	x_4
y_3	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_1	x_5	$\boldsymbol{x_1}$	$\boldsymbol{x_5}$	x_4	x_3
y_4	$\boldsymbol{x_3}$	$\boldsymbol{x_2}$	x_5	x_4	$\boldsymbol{x_5}$	$\boldsymbol{x_4}$	x_3	x_2
y_5	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_4	x_3	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_2	x_1

Table 3: Extending the right super-game from super-teams $S_i = \{x_1x_2, x_3x_4\}$ to $S_j = \{y_1y_2, y_3y_4\}$ with team-pair $\{x_5y_5\}$ into games on 8 days, where home normal games are marked in bold

where games in $\{x \leftrightarrow y \mid x \in S_i, y \in S_j\}$ are all scheduled in extending the left super-game.

Right super-games: $S_i \xrightarrow{T_{i'}} S_j$**.** Note that $T_{i'}$ is a team-pair. We relabel $S_i = \{x_1x_2, x_3x_4\}$ and $S_j = \{y_1y_2, y_3y_4\}$. If the position of S_i is above S_j, we relabel $T_{i'} = \{x_5, y_5\}$ such that $v_{4m+2i'-1} = x_5$ and $v_{4m+2i'} = y_5$; otherwise we set $v_{2i'-1} = y_5$ and $v_{2i'} = x_5$. Then, we extend the right super-game into games on 8 days, as shown in Table 2.

Note that we relabel the teams in $T_{i'}$ this way to avoid scheduling repeated games, as each super-team plays two right super-games involving $T_{i'}$. Moreover, since there are two additional teams, the games in $\{x_i \leftrightarrow y_{5-i} \mid 1 \leq i \leq 4\} \cup \{x_5 \leftrightarrow y_5\}$ are not scheduled during the extension.

Last super-games: $S_i \xrightarrow{\star} S_j$**.** Similarly, we relabel $S_i = \{x_1x_2, x_3x_4\}$ and $S_j = \{y_1y_2, y_3y_4\}$. We extend the last super-game into games on 8 days, as shown in Table 4, where games in $\{x \leftrightarrow y \mid x \in S_i, y \in S_j\}$ are all scheduled in extending the last super-game.

Scheduling Unscheduled Games

After extending all super-games, some games may remain unscheduled. In this section, we address and schedule these remaining games.

Firstly, we consider super-team S_m and the r team-pairs.

Since their positions are fixed, by the extension of super-games, games between any pair of teams within $V' = \cup_{i \leq r} T_i \cup S_m$ are unscheduled, i.e., $\{u \leftrightarrow v \mid u, v \in V'\}$, forming a double round-robin tournament among the teams

	1	2	3	4	5	6	7	8
x_1	y_1	y_2	y_3	y_4	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$
x_2	y_2	y_1	y_4	y_3	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$
x_3	y_3	y_4	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$	$\boldsymbol{y_3}$	$\boldsymbol{y_4}$	y_1	y_2
x_4	y_4	y_3	$\boldsymbol{y_2}$	$\boldsymbol{y_1}$	$\boldsymbol{y_4}$	$\boldsymbol{y_3}$	y_2	y_1
y_1	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	x_1	x_2	x_3	x_4
y_2	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_2	x_1	x_4	x_3
y_3	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	x_1	x_2	x_3	x_4	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$
y_4	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	x_2	x_1	x_4	x_3	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$

Table 4: Extending the last super-game from super-teams $S_i = \{x_1x_2, x_3x_4\}$ to $S_j = \{y_1y_2, y_3y_4\}$ into games on 8 days, where home games are marked in bold

	1	2	3	4	5	6
x_1	x_4	x_3	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	$\boldsymbol{x_4}$	x_2
x_2	x_3	x_4	x_1	$\boldsymbol{x_4}$	$\boldsymbol{x_3}$	$\boldsymbol{x_1}$
x_3	$\boldsymbol{x_2}$	$\boldsymbol{x_1}$	x_4	x_1	x_2	$\boldsymbol{x_4}$
x_4	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$	x_2	x_1	x_3

Table 5: Extending an away self-game for super-team $S_i = \{x_1x_2, x_3x_4\}$ into games on 6 days, where home games are marked in bold

in V'. To schedule these games, we use any TTP-2 algorithm, such as the one from Thielen and Westphal (2012). Since $S_m = \{x_1x_2, x_3x_4\}$ plays an away last super-game in the last time slot, both x_1 and x_2 have just played 4 consecutive home games, as shown in Table 4. So, when invoking the TTP-2 algorithm, we additionally ensure that both x_1 and x_2 start with away games. This can be done trivially since every day must contain $|V'|/2 \geq |S_m|/2 \geq 2$ games.

Secondly, we consider a super-team $S_i \in \{S_1, ..., S_{m-1}\}$.

On one hand, similar to the previous case, games between any pair of teams in S_i are unscheduled, i.e., $\{u \leftrightarrow v \mid u, v \in S_i\}$, forming a double round-robin tournament among the teams in S_i. To schedule these games, we assign an away (resp., home) *self-game* for S_i if S_i plays an away (resp., home) super-game in the last time slot. The away self-game will be extended into games on 6 days, as shown in Table 5. Note that the extension for the home self-game follows the same format as the away self-game but with reversed home venues.

On the other hand, we need to schedule the missing games resulting from extending the right super-games. Recall that in extending a right super-game between super-teams $\{x_1x_2, x_3x_4\}$ and $\{y_1y_2, y_3y_4\}$, the games in $\{x_i \leftrightarrow y_{5-i} \mid 1 \leq i \leq 4\}$ are not scheduled. Since every super-team in $\{S_1, ..., S_{m-1}\}$ plays two right super-games with each team-pair $T_{i'}$, each team in $V'' = \cup_{i<m} S_i$ has 4 unscheduled games due to right super-games with $T_{i'}$. Therefore, there are $4r$ unscheduled games in total. Let $U_{i'}$ denote the set of unscheduled games due to right super-games with $T_{i'}$ for all teams in V''. Consequently, $U_{i'}$ contains $4 \cdot |V''|$ games. Let A (resp., H) denote the state of a team playing an away (resp., a home) game. We have the following result.

Lemma 2 (*). *For any $i' \leq r$, all games in $U_{i'}$ can be scheduled within 4 days such that each team in V'' plays games in the patterns AHHA or HAAH.*

Since no team plays 4 consecutive home or away games in the last 4 days of the extension of self-games, the games in $U_{i'}$ can be safely scheduled one by one for each $i' \leq r$.

Theorem 3 (*). *For TTP-4 with any $n \geq 32$, our construction generates a feasible solution.*

Lower Bounds of the Optimal Solution

In this section, we propose two lower bounds for the weight of the optimal solution of TTP-4. Both of these bounds are related to the minimum weight matching, which is crucial for proving the approximation ratio of our algorithm.

The first lower bound is simple but will be useful.

Lemma 4 (*). *For TTP-4, it holds that $n \cdot w(M^*) \leq \frac{1}{2} \cdot OPT$.*

The second lower bound is also related to $\delta(V)$. Note that Westphal and Noparlik (2014) proved that $\delta(V) \leq 2 \cdot \text{OPT}$ for TTP-4. We propose a stronger result.

Lemma 5 (*). *For TTP-4, it holds that $\delta(V) + n \cdot w(M^*) \leq 2 \cdot OPT$.*

An Improved Approximation Algorithm

In this section, we demonstrate that by using our matching-based construction and the new lower bounds related to the minimum weight matching, we can achieve a $(1.625 + \varepsilon)$-approximation algorithm for TTP-4 for any constant $\varepsilon > 0$.

By labeling the $n/2$ edges in the minimum weight matching $M^* = \{e_1, ..., e_{n/2}\}$ such that $e_i = v_{2i-1}v_{2i}$ for each $i \in \{1, ..., n/2\}$, the previous construction provides a solution. However, there are $(n/2)! \cdot 2^{n/2}$ possible ways to label all teams, and different ways may result in different solutions. To find a good solution, we choose one labeling of teams uniformly at random from the $(n/2)! \cdot 2^{n/2}$ possibilities. The algorithm is outlined in Algorithm 1, which has a time complexity of $O(n)$, achieved by using the Fisher–Yates shuffle (Knuth 1997).

Algorithm 1: Randomized Labeling of Teams

Require: The minimum weight matching M^*
Ensure: One labeling of teams
1: Obtain a permutation uniformly at random from all possible permutations of the edges in M^*, and label the edges as $e_1, e_2, \ldots, e_{n/2}$.
2: **for** each edge e_i **do**
3: Select one of its vertices uniformly at random, label it as v_{2i-1}, and label the other vertex as v_{2i}.
4: **end for**

In our solution, the road trips of each team are related to their labels. To analyze their weight, due to the randomness, we need to calculate the *expected* weight of each edge v_iv_j, where $i, j \in \{1, ..., n\}$. Next, we analyze the expected weights of matching edges and non-matching edges.

Lemma 6 (*). *Each matching edge has an expected weight of $\frac{2}{n} w(M^*)$, and each non-matching has an expected weight of at most $\frac{1}{n(n-2)} \delta(V)$ and at least $\frac{2}{n} w(M^*)$.*

Lemma 6 demonstrates that, under our randomized labeling algorithm, each (non-)matching edge has the same expected weight. This property will simplify our forthcoming analysis of the expected weight of our algorithm, as we will only need to count the number of (non-)matching edges used in the road trips of all teams. For the previous approximation algorithm in (Zhao, Xiao, and Xu 2022), the analysis is derived by making a trade-off between a construction based on an approximate Hamiltonian cycle (Yamaguchi et al. 2011) and a construction based on an approximate 4-cycle packing (i.e., a set of vertex-disjoint 4-cycles), both of which involve NP-hard problems.

Theorem 7. *For TTP-4 with $n \geq 32$, there is a randomized algorithm to generate a solution with an expected weight of at most $n \cdot w(M^*) + (\frac{3}{4} + \frac{206}{n-2})\delta(V)$ in $O(n^3)$ time.*

Proof. Our solution of TTP-4 can be denoted by a $n \times (2n-2)$ table. There are $2n(n-1)$ cells in the table, where each cell represents a specific game.

In our super-tournament, each of the first $m-2$ time slots includes $m/2-r-1$ normal super-games. Since $r \leq 3$, the number of normal super-games is at least

$$\left(\frac{m}{2} - r - 1\right)(m-2) \geq \left(\frac{m}{2} - 3 - 1\right)(m-2) = \frac{m^2}{2} - 5m + 8.$$

Let $z = \frac{m^2}{2} - 5m + 8$. We fix z normal super-games arbitrarily in our super-tournament. As shown in Figure 5, there are 8 road trips in extending one normal super-game, which include 16 matching edges and 24 non-matching edges. Hence, the road trips corresponding to games in extending the z normal super-games include m_1 matching edges and n_1 non-matching edges, where

$$(m_1, n_1) = (16z, 24z). \tag{3}$$

By Table 1, each normal super-game occupies $8 \times 8 = 64$ cells. The z normal super-games therefore occupy $64z$ cells in total. The remaining $\overline{z} = 2n(n-1) - 64z$ cells are referred to as *bad* cells. Consider any bad cell and assume that the corresponding game is $u \to v$. We assume that u and v are at home before the game starts and return home after the game ends. This assumption only increases the weight of our analysis by the triangle inequality. So, u takes a road trip including the edge uv two times, while v takes no road trips. Hence, each bad cell corresponds to at most one road trip, which is a 2-cycle including two edges. Moreover, we assume that this road trip contains two non-matching edges, as the expected weight of a matching edge is at most the expected weight of a non-matching edge by Lemma 6. Hence, the games in bad cells incur at most $\overline{z}$ road trips,including m_2 matching edges and n_2 non-matching edges, where

$$(m_2, n_2) = (0, 2\overline{z}). \tag{4}$$

By (3), (4), and Lemma 6, the expected weight of our solution is at most

$$\frac{16z}{n} w(M^*) + \frac{24z + 2\overline{z}}{n(n-2)} \delta(V). \tag{5}$$

On one hand, since $m = 2\lfloor \frac{n}{8} \rfloor$ and $n \geq 32$, we have $8 \leq m \leq \frac{n}{4}$, and thus $z = \frac{m^2}{2} - 5m + 8 \leq \frac{m^2}{2} \leq \frac{n^2}{16}$. Thus,

$$\frac{16z}{n} w(M^*) \leq n \cdot w(M^*). \tag{6}$$

On the other hand, we have $6 \leq \frac{n}{4} - 2 = 2(\frac{n}{8} - 1) \leq m \leq \frac{n}{4}$, so $z = \frac{m^2}{2} - 5m + 8 \geq \frac{1}{2}(\frac{n}{4} - 2)^2 - 5(\frac{n}{4}) + 8 \geq \frac{n^2}{32} - 2n$. Recall that $\overline{z} = 2n(n-1) - 64z$. We have $24z + 2\overline{z} = 4n(n-1) - 104z \leq \frac{3}{4}n^2 + 204n$. Hence, we have

$$\begin{aligned} \frac{24z + 2\overline{z}}{n(n-2)} \delta(V) &\leq \frac{\frac{3}{4}n^2 + 204n}{n(n-2)} \delta(V) \\ &\leq \left(\frac{3}{4} + \frac{206}{n-2}\right) \delta(V). \end{aligned} \tag{7}$$

By (5), (6), and (7), the expected weight of our solution is at most $n \cdot w(M^*) + (\frac{3}{4} + \frac{206}{n-2})\delta(V)$.

Our algorithm first computes the minimum weight matching in $O(n^3)$ time[1], then find a labeling of all teams in $O(n)$ time, and finally obtain a schedule using the matching-based construction in $O(n^2)$ time. Therefore, the overall running time of our algorithm is $O(n^3)$. □

Then, by using our new lower bounds related to the minimum weight matching, we can analyze the expected approximation ratio of our algorithm.

Theorem 8 (*)**.** *For TTP-4 with any constant $\varepsilon > 0$, there is a randomized $(1.625 + \varepsilon)$-approximation algorithm.*

Note that by a more refined analysis, the item $\frac{206}{n-2}\delta(V)$ in Theorem 7 could be further improved. However, this would not lead to an improvement in the key constant 1.625 in our approximation ratio.

Moreover, our algorithm can be derandomized in polynomial time using the method of conditional expectations, while preserving the approximation ratio. Examples of this approach can be found in (Williamson and Shmoys 2011; Miyashiro, Matsui, and Imahori 2012).

Remark 9. *Currently, our "matching-based" method can only improve the approximation ratio for $k = 4$. For $k > 4$, although our method can still derive a feasible solution, the approximation ratio becomes worse since the item related to $\delta(V)$ in the analysis becomes too large. Below are some detailed arguments.*

For TTP-k, we may still be able to ensure that almost all road trips consist of $k-1$ away edges and 2 home edges (i.e., each team always plays k consecutive away games), where there are at most $\lfloor \frac{k}{2} \rfloor$ matching edges in the away edges. In our analysis, the rest $k + 1 - \lfloor \frac{k}{2} \rfloor$ edges, should be regarded as non-matching edges. So, this method may yield a solution with a weight of at most $\frac{2\lfloor \frac{k}{2} \rfloor}{k} \cdot n \cdot w(M^*) + \frac{k+1-\lfloor \frac{k}{2} \rfloor}{k} \delta(V) + \varepsilon \cdot OPT$. Unfortunately, for $k > 4$, the item related to $\delta(V)$ will become large and we cannot beat the known result. And, for $k = 2$ or $k = 3$, other methods achieve better approximation ratios, and it is challenging to improve upon them.

[1]Recently, advanced techniques have led to some improvements in the running time of the minimum weight matching algorithm (see e.g., Chen et al. (2022)).

Experimental Results

In the construction and analysis, we can see that the simplest case is the case where $n \equiv 0 \pmod 8$. Other cases need additional steps to handle some remaining teams. Although these additional parts only handle a few special cases, they are relatively complex in the construction and not easy to implement. In the experimental part, we will also mainly focus on the case where $n \equiv 0 \pmod 8$, due to the limited space. For this case, there are no team-pairs and, therefore, no right super-games.

For TTP-4, the current best-known experimental results are from Westphal and Noparlik (2014), where they tested 30 benchmark instances from the online website (Trick 2024). Among these, nine instances satisfy $n \equiv 0 \pmod 8$. Their results were achieved by combining an approximation algorithm based on the Hamiltonian cycle with the LKH heuristic algorithm (Helsgaun 2000) and additional swapping heuristics (Di Gaspero and Schaerf 2007; Ribeiro and Urrutia 2007). This approach produces high-quality solutions; for instance, it improved 10 hard benchmark instances for the widely studied TTP-3. So, improving upon their results for TTP-4 poses a significant challenge.

Next, we introduce some details on the implementations of our algorithm. In fact, we will modify the algorithm a little bit which will not lead to worse theoretical results, but they are easier to implement and have more practical effects.

First, we use two minor modifications of the construction.

- We replace the left super-game $S_m \xrightarrow{\blacktriangle} S_{m-1}$ in the first time slot with a normal super-game $S_m \leftarrow S_{m-1}$.
- After S_m plays an away last super-game in the last time slot, instead of calling a TTP-2 algorithm, we assign an away self-game for it, as we do for other super-teams.

So, any last super-game, along with the two subsequent self-games, can be regarded together as a new last super-game.

Second, we optimize our super-games as follows: during the extension of each super-game between any pair of super-teams S_i and S_j, we relabel the teams in S_i and S_j to minimize the total traveling distance for all teams in $S_i \cup S_j$ on the extended days. There are $4! \times 4! = 576$ possible ways to relabel the teams, and we will choose the best way.

Last, instead of using the randomized labeling algorithm, we form m super-teams as follows: (1) We obtain a new graph G' by contracting each edge in M^* into a vertex, where the weight between vertices corresponding to edges $xy \in M^*$ and $zt \in M^*$ is defined as $\min\{w(x,z) + w(y,t),\ w(x,t) + w(y,z)\}$. (2) We then find a minimum weight matching M' in G', which pairs the $2m$ edges in M^* into m pairs of edges. (3) Finally, we take the four teams from each pair of edges in M' to form a super-team. This approach is chosen because the away edges in the road trips of a normal super-game form 4-cycles containing edges in M^*, as illustrated in Figure 5(a).

After forming m super-teams, we attempt to swap the indices of two super-teams to find better solutions. Specifically, there are $\binom{m}{2}$ possible pairs of super-teams. We consider these pairs in a random order. For each pair, from the first to the last, we test whether swapping the indices of the super-teams results in a better schedule. If not, we do not swap them and move on to the next pair. If it does result in a better schedule, we swap them and proceed to the next pair. After evaluating all $\binom{m}{2}$ pairs, if an improvement is found, we repeat the procedure. Otherwise, the process concludes.

The above swapping heuristic is also known as pair-wise swapping local search (Goerigk et al. 2014), which is simple yet effective, especially for these TTP-4 instances where the size is relatively small, i.e., $n \leq 40$.

Our algorithm is implemented in C++ and executed on a standard Windows desktop computer with an Intel Core i5-11400 CPU (2.6 GHz) and 16 GB of RAM. Our code is available at https://github.com/JingyangZhao/TTP-4.

Data Set	Best-Known Result	Our Result	Improvement Ratio
Galaxy40	216863	**212680**	1.93%
Galaxy32	104433	**103054**	1.32%
Galaxy24	40832	**39453**	3.38%
Galaxy16	14365	**13188**	8.19%
NFL32	809725	**779620**	3.72%
NFL24	436531	**403660**	7.53%
NFL16	235936	**213649**	9.45%
NL16	263745	**229301**	13.06%
Brazil24	464921	**412399**	11.30%

Table 6: Experimental Results

Our results are summarized in Table 6. The column '*Data Set*' lists the nine instances, '*Best-Known Result*' indicates the best-known results from Westphal and Noparlik (2014), '*Our Result*' shows the results obtained by our algorithm after applying the pair-wise swapping local search, and '*Improvement Ratio*' is defined as $\frac{\text{Best-Known Result} - \text{Our Result}}{\text{Best-Known Result}}$.

We can see that our algorithm improves upon the previous results for all the nine instances satisfying $n \equiv 0 \pmod 8$, achieving an average improvement ratio of 6.65%. Our algorithm also runs very fast. All the nine instances can be solved together within 9.55 seconds.

We remark that to extend our algorithm to cases where $n \not\equiv 0 \pmod 8$, we only need to design effective right super-games and schedule the games after the last time slot.

Conclusion

In this paper, we present an improved approximation algorithm for TTP-4. Compared to the previous approximation algorithm in (Zhao, Xiao, and Xu 2022), our approach to design super-games is different. Our design is based on two edges in a minimum weight matching, whereas their design is based on 4-cycles. Additionally, our analysis of the solution quality is simple. We focus on counting the number of (non-)matching edges in road trips, while their analysis involves calculating the costs of super-games. Finally, we can improve the approximation ratio from $1.7+\varepsilon$ to $1.625+\varepsilon$ for any constant $\varepsilon > 0$. Our algorithm also demonstrates excellent practical performance, particularly for the case $n \equiv 0 \pmod 8$. Experimental results indicate that our algorithm improves all 9 instances satisfying $n \equiv 0 \pmod 8$, achieving an average improvement ratio of 6.65%.

Acknowledgments

The work is supported by the National Natural Science Foundation of China, under the grants 62372095, 62172077, and 62350710215.

References

Anagnostopoulos, A.; Michel, L.; Van Hentenryck, P.; and Vergados, Y. 2006. A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, 9(2): 177–193.

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

Bulck, D. V.; Goossens, D. R.; Schönberger, J.; and Guajardo, M. 2020. RobinX: A three-field classification and unified data format for round-robin sports timetabling. *European Journal of Operational Research*, 280(2): 568–580.

Chatterjee, D. 2021. Complexity of Traveling Tournament Problem with Trip Length More Than Three. arXiv:2110.02300.

Chatterjee, D.; and Roy, B. K. 2021. An Improved Scheduling Algorithm for Traveling Tournament Problem with Maximum Trip Length Two. In *ATMOS 2021*, volume 96 of *OASIcs*, 16:1–16:15.

Chen, L.; Kyng, R.; Liu, Y. P.; Peng, R.; Gutenberg, M. P.; and Sachdeva, S. 2022. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, 612–623. IEEE.

Di Gaspero, L.; and Schaerf, A. 2007. A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, 13(2): 189–207.

Durán, G. 2021. Sports scheduling and other topics in sports analytics: a survey with special reference to Latin America. *Top*, 29(1): 125–155.

Easton, K.; Nemhauser, G. L.; and Trick, M. A. 2001. The Traveling Tournament Problem Description and Benchmarks. In *CP 2001, Proceedings*, volume 2239 of *Lecture Notes in Computer Science*, 580–584. Springer.

Easton, K.; Nemhauser, G. L.; and Trick, M. A. 2002. Solving the Travelling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach. In *PATAT 2002*, volume 2740 of *Lecture Notes in Computer Science*, 100–112. Springer.

Frohner, N.; Neumann, B.; Pace, G.; and Raidl, G. R. 2023. Approaching the traveling tournament problem with randomized beam search. *Evolutionary Computation*, 31(3): 233–257.

Gabow, H. N. 1974. *Implementation of algorithms for maximum matching on nonbipartite graphs*. Ph.D. thesis, Stanford University.

Goerigk, M.; Hoshino, R.; Kawarabayashi, K.; and Westphal, S. 2014. Solving the Traveling Tournament Problem by Packing Three-Vertex Paths. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2271–2277. AAAI Press.

Goerigk, M.; and Westphal, S. 2016. A combined local search and integer programming approach to the traveling tournament problem. *Annals of Operations Research*, 239(1): 343–354.

Helsgaun, K. 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1): 106–130.

Hentenryck, P. V.; and Vergados, Y. 2007. Population-Based Simulated Annealing for Traveling Tournaments. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 267–272. AAAI Press.

Hoshino, R.; and Kawarabayashi, K. 2012. The Linear Distance Traveling Tournament Problem. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 1770–1778. AAAI Press.

Hoshino, R.; and Kawarabayashi, K. 2013. An Approximation Algorithm for the Bipartite Traveling Tournament Problem. *Mathematics of Operations Research*, 38(4): 720–728.

Imahori, S. 2021. A 1+O(1/N) approximation algorithm for TTP(2). arXiv:2108.08444.

Imahori, S.; Matsui, T.; and Miyashiro, R. 2014. A 2.75-approximation algorithm for the unconstrained traveling tournament problem. *Annals of Operations Research*, 218(1): 237–247.

Knuth, D. E. 1997. *The art of computer programming*, volume 3. Pearson Education.

Lawler, E. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston.

Lim, A.; Rodrigues, B.; and Zhang, X. 2006. A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research*, 174(3): 1459–1478.

Miyashiro, R.; Matsui, T.; and Imahori, S. 2012. An approximation algorithm for the traveling tournament problem. *Annals of Operations Research*, 194(1): 317–324.

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

Thielen, C.; and Westphal, S. 2011. Complexity of the traveling tournament problem. *Theoretical Computer Science*, 412(4): 345–351.

Thielen, C.; and Westphal, S. 2012. Approximation algorithms for TTP(2). *Mathematical Methods of Operations Research*, 76(1): 1–20.

Trick, M. 2024. Challenge traveling tournament instances. https://mat.tepper.cmu.edu/TOURN/. Accessed: 2024-08-15.

Verduin, K.; Thomson, S. L.; and van den Berg, D. 2024. Too Constrained for Genetic Algorithms. Too Hard for Evolutionary Computing. The Traveling Tournament Problem. In *15th International Conference on Evolutionary Computation Theory and Applications, ECTA 2023*, 246–257.

Westphal, S.; and Noparlik, K. 2014. A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, 218(1): 347–360.

Williamson, D. P.; and Shmoys, D. B. 2011. *The design of approximation algorithms*. Cambridge university press.

Xiao, M.; and Kou, S. 2016. An Improved Approximation Algorithm for the Traveling Tournament Problem with Maximum Trip Length Two. In *MFCS 2016*, volume 58 of *LIPIcs*, 89:1–89:14.

Yamaguchi, D.; Imahori, S.; Miyashiro, R.; and Matsui, T. 2011. An improved approximation algorithm for the traveling tournament problem. *Algorithmica*, 61(4): 1077–1091.

Zhao, J.; and Xiao, M. 2021. 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 2021*, 4206–4212.

Zhao, J.; and Xiao, M. 2023. The Linear Distance Traveling Tournament Problem Allows an EPTAS. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*, 12155–12162. AAAI Press.

Zhao, J.; Xiao, M.; and Xu, C. 2022. Improved Approximation Algorithms for the Traveling Tournament Problem. In *MFCS 2022*, volume 241 of *LIPIcs*, 83:1–83:15.