An Algorithm for the Bipartite Linear Distance Traveling Tournament Problem

William Thrasher
Florida State University
wjt1321@my.fsu.edu

Abstract

I apply a linear distance heuristic to the n-team Bipartite Traveling Tournament problem, which treats the problem as if the n teams are located on a straight line, reducing the number of variables. I then generate an algorithm to solve this variation of the problem. I then compare the results of my algorithm with results presented in previous papers. My algorithm does not present an optimal solution to the problem, but does provide a good starting point and generates a solution very quickly.

1 Introduction

The Traveling Tournament Problem (TTP) is a problem in the area of sports scheduling inspired by the difficulties inherent in optimizing the regular season schedule for Major League Baseball. It is a simple to state and has minimal data requirements, yet is a good benchmark problem for exploring various approaches to problem solving. The goal of the TTP is to determine the optimal schedule for an n-team sports league in a double round-robin tournament, minimizing the sum of the total distance traveled by the teams. [Easton *et al.*, 2001]

There are variations of the TTP, two of which should be noted as they directly apply to this paper. First, the Linear Distance Traveling Tournament Problem (LD-TTP) is a heuristic in which one assumes that the teams are located on a straight line. This problem reduces the complexity of the TTP enought that all possible solutions can be discovered without any use of computing for the cases of $n = 4$ and $n = 6$. It also allows for the easier generation of approximate solutions to larger datasets. [Hoshino and ichi Kawarabayashi, b] Another variation to consider is the Bipartite Traveling Tournament Problem (BTTP). In this variation, instead of modeling intra-league play, the problem models inter-league play, in which each member of a league must play each member of the opposite league twice. This is an NP-complete problem that seems a natural progression from the TTP. [Hoshino and ichi Kawarabayashi, a]

In this paper, I combine the LD-TTP and the BTTP and create an algorithm to assist in generating an approximate solution to this Linear Distance Bipartite Traveling Tournament Problem (LD-BTTP). This technique does not easily generate an optimal schedule, but generates a schedule surprisingly quickly, with minimum computation time, and serves as a starting point towards solving this problem.

2 Background/Prior Work

There have been a number of attempts to solve the TTP and its many variants. Some of the solutions are precise algorithms, while some only generate approximate solutions. In addition, there are three sets of test instances widely used as benchmarks for the traditional TTP [Kendall *et al.*, 2010], although none are appropriate for the BTTP. In order to understand the problem proposed in this paper, an understanding of the TTP, LD-TTP ,and BTTP are useful.

2.1 The Traveling Tournament Problem

The Traveling Tournament Problem is a problem simulating a round-robin tournament in which every team plays every other team. For each game, one team is the home team and the other is the away team. Given n teams, distances between team sites are given by an n by n matrix. When teams travel to an away game, it travels from its home site to the away venue unless the team's previous game was an away game. In this game, the team travels from the previous away venue. A home stand is defined as consecutive home games for a team. An away trip is defined as consecutive away games for one team. In general, the length of every home stand and away trip is between L and U inclusive and the goal of the problem is to minimize total distance traveled. In addition to these constraints, there are often additional constraints. For the purposes of this paper, we will be using the No Repeaters constraint, which means that no team may play the same team twice in a row. [Easton *et al.*, 2001]

Following the lead of Hoshino and Kawarabayashi, I will assume values of L and U to be $L = 1$ and $U = 3$. This means that no home stand or away trip can last more than 3 games. [2012]

2.2 The Linear Distance Traveling Tournament Problem

The Linear Distance Traveling Tournament Problem is a relaxation of the rules in the TTP where we assume the n teams are on a straight line. This greatly reduces the complexity of the problem and makes it much easier to find solutions to the optimal tournament schedule. In this variant, it is important to

note that each team is only associated with two distances and to travel from one team to another, it is often required to pass through the locations of other teams. The specific order of the teams on the line is produced through a "line of best fit" or an informal check by looking at the map of the teams. This variant is a natural heuristic when the teams are connected by a train line running in one direction. This models the actual context of sports leagues in many countries. The paper originally outlining this variant of the problem produced an algorithm that provides a good approximation of the optimal solution and proved that for $n = 4$ and $n = 6$, the LD-TTP could produce an optimal solution. This algorithm produced feasible solutions no matter the non-linearity of the benchmark data sets. [Hoshino and ichi Kawarabayashi, b]

2.3 The Bipartite Traveling Tournament Problem

In many sports today, inter-league play has been added to the season, which adds an additional level of complexity to the problem of scheduling teams. This led to the introduction of the Bipartite Traveling Tournament Problem modeling interleague play. In this variant, there are $2n$ teams, with n teams in each league. The distance matrix in this case is a $2nX2n$ distance matrix, where each entry is the distance between two home stadiums. There are two leagues, which will be called X and Y, with $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$. This follows all the constraints of the original TTP, but instead of playing each team within its league, each team must play every team in the opposite league. All other constraints remain the same.

In addition, there is a restriction of the BTTP called BTTP*. In this variant, the teams in each league all play at home or all play on the road. This uniformity constraint reduces the number of possible solutions, but allows us to more quickly generate approximate solutions. By definition, both BTTP and BTTP* are NP-complete. [Hoshino and ichi Kawarabayashi, a]

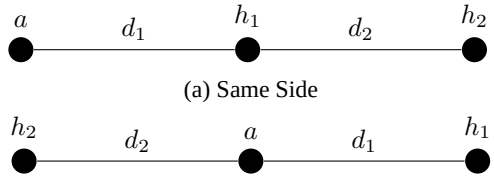
3 The Linear Distance Bipartite Traveling Tournament Problem

In this paper, I combine the two previously discussed variants on the TTP. I assume $2n$ teams, with n teams in each league, with a $2nX2n$ distance matrix. There are twoi leagues, as in the BTTP. However, I also applied the Linear Distance Heuristic to each data set. Rather than separating the leagues into two separate lines, I placed the entire dataset on a single line, with no regard to the league each team was in. To do this, I used an informal check, although for some data sets, the informal check would approximate a "line of best fit." Also, for the purposes of deriving a simpler algorithm, I followed the restrictions of the BTTP* variant, so each league was either all at home or all on the road. As a reminder, no team was allowed to repeat games by playing a team at home and then playing the same team away (or vice versa). Also, no home stand or away trip was allowed to be longer than three games.

In examining this problem, I noticed one fact that was vital in creating a simple algorithm for this problem. In any given away trip, the team visited first does not affect the total distance traveled, as long as all teams in one direction are visited before all teams in the opposite direction of the line and the away team does not reverse direction except at the sites that are the greatest distance from the away team's site. This distance is, in fact, twice the distance between the two involved sites that are furthest away from each other. I will prove this in the following section.

/subsectionFirst Site Does Not Matter

Take an away team a and 1-3 home teams h_1, h_2, and h_3. For one-game away trips, the away team will travel the distance between a and the home team's site and then return to the away team's site over the same route. This is obviously twice the difference between the home team and the away team.

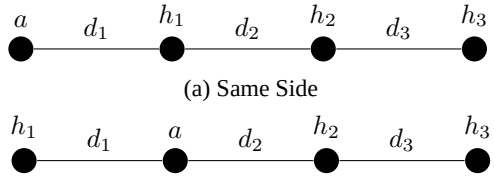


(a) Same Side

(b) Opposite Side

Figure 1: Two-Game Away Trip Configurations

For two-game away trips, it is slightly more difficult, as the teams could be in two different orders. In the first case, the two home teams are on the same side, as in Figure 1a. In this case, the away team will travel to one team first, I will arbitrarily choose h_1 and distance d_1, then travel back across that same distance in order to get to the other team (in this case, h_2, adding $d_1 + d_2$ to the distance traveled), and then travel back to the away team's site (in this case, adding another d_2 for a total of $d_1 + d_1 + d_2 + d_2$ or $2(d_1 + d_2)$). This means that the total distance traveled will be twice the distance between the two home team sites. If the two teams are on the same side, as in Figure 1b, then it is easy to see that the distance traveled will be twice the distance between the away team's site and the site of the furthest home team. The away team will travel one of the following two paths: $a - h_1 - h_2 - a$ or $a - h_2 - h_1 - a$, both of which have distance $2(d_1 + d_2)$.



(a) Same Side

(b) Opposite Sides

Figure 2: Three-Game Away Trip Configurations

For three-game away trips, it is very similar. There are again two possible configurations. The first, as shown in Figure 2a, is all 3 home sites are in the same direction from the away team. If team a visits the teams in order from left to right, it will travel the distances in the following order: $d_1, d_2, d_3, d_3, d_2, d_1$. This totals to $2(d_1 + d_2 + d_3$. If it vis-

its team h_2 first, then, remembering the caveat that it cannot change directions at a "middle" site like this one, it will travel the distances in the same order, although it will visit the sites in the order $h_2, h_3, h1$. If it visits team h_3 first, the distances will again be traveled in the same order, although the site order is now h_3, h_2, h_1. In the other configuration, as in Figure 2b, is one team in one direction and the other two teams in the opposite direction. In this case, if a visits h_1 first, it will encounter the distances as $d_1, d_1, d_2, d_3, d_3, d_3$ as it visits the teams in the order h_1, h_2, h_3. If visiting team h_2 first, the distance order shall be $d_2, d_3, d_3, d_2, d_1, d_1$. An away trip with h_3 first follows the order h_3, h_2, h_1. Each of those cases has a distance of $2(d_1 + d_2 + d_3$.

3.1 Other Factors

Adding the extra constraint of the BTTP* allows me to use this information to treat groups of 3 or in a manner similar to individual teams. As the order of visitation does not change the distance, assigning one set to another set gives a computationally inexpensive methord of calculating the distance traveled for multiple sets of games.

The next factor to consider, then, is how to choose both the number of teams in each group and the exact number of teams in each group. Also, one must consider whether the groups should be the same for sets of home and away games. As an example of what I mean by that last sentence, if there were two leagues and one had teams $x_1, x_2, x_3, and x_4$, then x_1 and x_2 could be grouped together when they were playing home games, but x_1 and x_3 could be grouped together when playing away games.

It seems logical that teams that are "adjacent" on the imaginary line that we have placed teams on would be closer together than teams that were not adjacent, as an intelligently drawn line would be unlikely to skip over a nearby team to add a team a further distance away. This would, of course, depend partially on the linearity of the team locations in the real world as well, so it is safe to assume that any schedule drawn with this assumption would be better for more linear leagues. Given that, it makes the most sense to make groups of adjacent teams, so that a team will theoretically not be forced to travel across large portions of the line as often. In order to reduce trips across large portions of the line, it would also make sense to have as few away trips as possible, as multiple away trips to adjacent teams means more backtracking for the traveling team and thus a larger total distance traveled. Thus, I chose to go with groups of 3 adjacent teams, when possible.

4 Solving the Problem

In order to test my solution, I created a program that uses the following procedure. First, the program reads in the data, stores each team, an index for each team, and the distance along the line to the next team. From there, the program splits each league into a number of groups by simply assigning the first three teams from a given league in the line to a group, the second three teams to a group, and so on. To calculate the distance, the program loops over each group, matching every group in one team with every group in another team, and calculates the distance for each team in the group to travel to each team in the matched group. It does this by simply finding the difference between the team with the lowest index and the team with the highest index (the two teams that are farthest apart) and doubling it. It adds these distances together and prints out a sample schedule and the total distance.

In order to test this against benchmarks, I used two sets of data with optimal and near-optimal solutions previously offered, the 12-team Nippon Professional Baseball (NPB) league and the 30-team National Basketball Association (NBA). [Hoshino and ichi Kawarabayashi, a; 2011]. For each group, I used the team distances presented in *Scheduling Bipartite Tournaments to Minimize Total Travel Distance* by Hoshino and Kawarabayashi.

4.1 NPB Benchmark Test

The NPB provided a fairly ideal situation for testing. The team locations are fairly linear, so deciding on a team order was not difficult and if the linearity of the teams affects my solution, there will not be much distortion. The two leagues are also not divided in an obvious way geographically, so there is nice integration in the linear path. The order of the teams and the symbol used to represent the teams are shown below in Figure 3. Teams that are in **bold** type are members of the Pacific League and teams that are in *italics* are in the Central League.

Fukoaka(p_1**)** — *Hiroshima(*c_1*)* — *Hanshin(*c_2*)* — **Orix(**p_2**)** — *Chunichi(*c_3*)* — **Seibu(**p_3**)** — *Yokohama(*c_4*)* — *Tokyo(*c_5*)* — *Yomiuri(*c_6*)* — **Chiba Lotte(**p_4**)** — **Tohoku(**p_5**)** — **Hokkaido(**p_6**)**

Figure 3: Nippon Professional Baseball League Linear Order

Upon running the program, the schedule in Figure 1 was generated. In this schedule, home games are presented in bold. This schedule had a final total distance of 52,838 km. This is not an optimal schedule at all. The optimal schedule had a travel distance of 42,950 km. The actual 2010 NPB inter-league schedule had a total travel distance of 51,134. [Hoshino and ichi Kawarabayashi, a] So, my algorithm produced a solution that was approximately 3% worse than the actual schedule and was a huge 23% worse than the optimal solution. While this is bad, it is not terrible and there was one definite advantage to my program over one that considered nearly every possible solution. Hoshino and Kawarabayashi used a Maplesoft program to generate all possible solutions and find the optimal solutions. This program generated its solutions in 34,716 seconds (just under 10 hours). [Hoshino and ichi Kawarabayashi, a] In contrast, my program took less than one second to complete.

Interestingly, for this data, it has been shown that the lower bound of a team's travel time would actually occur when that team played its road sets in either two blocks of three (as done here) or three blocks of two. [Hoshino and ichi Kawarabayashi, a] Because of this, I re-ran the program using blocks of two instead of three for this set of data, and the total distance traveled was worse.

	1	2	3	4	5	6	7	8	9	10	11	12
p_1	c_1	c_2	c_3	c_1	c_2	c_3	c_4	c_5	c_6	c_4	c_5	c_6
p_2	c_2	c_3	c_1	c_2	c_3	c_1	c_5	c_6	c_4	c_5	c_6	c_4
p_3	c_3	c_1	c_2	c_3	c_1	c_2	c_6	c_4	c_5	c_6	c_4	c_5
p_4	c_4	c_5	c_6	c_4	c_5	c_6	c_1	c_2	c_3	c_1	c_2	c_3
p_5	c_5	c_6	c_4	c_5	c_6	c_4	c_2	c_3	c_1	c_2	c_3	c_1
p_6	c_6	c_4	c_5	c_6	c_4	c_5	c_3	c_1	c_2	c_3	c_1	c_2
c_1	p_1	p_2	p_3	p_1	p_2	p_3	p_4	p_5	p_6	p_4	p_5	p_6
c_2	p_2	p_3	p_1	p_2	p_3	p_1	p_5	p_6	p_4	p_5	p_6	p_4
c_3	p_3	p_1	p_2	p_3	p_1	p_2	p_6	p_4	p_5	p_6	p_4	p_5
c_4	p_4	p_5	p_6	p_4	p_5	p_6	p_1	p_2	p_3	p_1	p_2	p_3
c_5	p_5	p_6	p_4	p_5	p_6	p_4	p_2	p_3	p_1	p_2	p_3	p_1
c_6	p_6	p_4	p_5	p_6	p_4	p_5	p_3	p_1	p_2	p_3	p_1	p_2

Table 1: NPB Proposed Schedule

4.2 NBA Benchmark Test

The NBA schedule is much less ideal than the NPB. The most obvious distance at first glance is that there are more teams (30, as opposed to 12). However, there are two other differences worth pointing out. First, the NBA is divided into the Eastern Conference and the Western Conference, so when creating a linear solution, the conferences are almost completely separate. Second, the NBA teams are not geographically linear in any way, so there is no obvious ordering and if the order of the teams affects the solution (as is likely), then finding the ideal order would be difficult.

I have listed below, in Figure 4, the ordering I chose for the teams and the abbreviations used. Teams in **bold** type are in the Eastern Conference, teams in *italics* are in the Western Conference.

Portland Tralblazers(PT) — Sacramento Kings(SK) — Golden State Warriors(GW) — Los Angeles Clippers(LC) — Los Angeles Lakers(LL) — Phoenix Sunds(PS) — Utah Jazz(UJ) — Denver Nuggets(DN) — Oklahoma City Thunder(OT) — Dallas Mavericks(DM) — San Antonio Spurs(SS) — Houston Rockets(HR) — New Orleans Hornets(NH) — Memphis Grizzlies(MG) — **Indiana Pacers(IP) — Chicago Bulls(CU) — Milwaukee Bucks(MB)** *— Minnesota Timberwolves(MT) —* **Detroit Pistons(DP) — Cleveland Cavaliers(CC) — Toronto Raptors(TR) — Boston Celtics(BC) — New York Knicks(NK) — New Jersey Nets(NN) — Philadelphia Sizers(PS) — Washington Wizards(WW) — Charlotte Bobcats(CB) — Atlanta Hawks(AH) — Orlando Magic(OM) — Miami Heat(MH)**

Figure 4: National Basketball Association Linear Order

Upon running the program, a schedule (not shown) was generated. This schedule had a final total distance of 123,375 mi. Unfortunately, this is more than double the optimal distance and the near-optimal distance shown in a previous paper. [Hoshino and ichi Kawarabayashi, a] This shows that my algorithm does not provide a good solution for the NBA schedule, at least wit the order I chose to look at.

5 Conclusion

This algorithm could use a lot of work, but the results with the Nippon Professional Baseball league are encouraging enough that I believe a method for determining near-optimal results for the Bipartite Traveling Tournament Problem using the Linear Distance heuristic. I believe that this algorithm works best when the team sites are more linear in the real world and, in fact, it might work very well when looking at sports leagues in countries where the main method to travel from site to site is from set train lines (or similar).

I believe that the ordering of the teams may have a large effect on the schedule produced from this algorithm, so a good next step would be to attempt to develop a method for generating the best ordering for a given league. It also seems likely that the method of grouping the teams would have a large effect on the final results, so another future project could be to determine how best to group the teams in home stands and away trips.

Besides the previously mentioned ideas, it may also be of use to look at this problem without the constraint taken from the BTTP*. If not all teams in a league have to be home at the same time, there is more flexibility in the scheduling, although it is definitely more complex.

Overall, I feel that this algorithm is a good first step in exploring the Linear Distance heuristic when applied to the Bipartite Traveling Tournament Problem and future research could build on this and, hopefully, determine whether this heuristic is helpful when applied to this variant of the problem.

References

[Easton *et al.*, 2001] Kelly Easton, George Nemhauser, and Michael Trick. The traveling tournament problem description and benchmarks. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 580–584, 2001.

[Hoshino and ichi Kawarabayashi, a] Richard Hoshino and Ken ichi Kawarabayashi. The inter-league extension of the traveling tournament problem and its application to sports scheduling. In *Proceedings of the Twenty-Fifth*

AAAI Conference on Artificial Intelligence, pages 977–984, San Francisco, California, month =. AAAI Press.

[Hoshino and ichi Kawarabayashi, b] Richard Hoshino and Ken ichi Kawarabayashi. The linear distance traveling tournament problem. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1770–1778, Palo Alto, California, month =. AAAI Press.

[Hoshino and ichi Kawarabayashi, 2011] Richard Hoshino and Ken ichi Kawarabayashi. Scheduling bipartite tournaments to minimize total travel distance. *Journal of Artificial Intelligence Research*, 42:91–124, October 2011.

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