Optimizing Major League Baseball Schedules with the Traveling Tournament Problem

Gavin Bulthuis

April 30, 2024

Abstract

Baseball is the game of America's past time and is one of the biggest leagues in the United States. Major League Baseball (MLB) is America's largest and most popular baseball league. The MLB's regular season spans from early April to early September and each team plays a total of 162 games. Playing 162 games can present a lot of fatigue for players and the expenses of travel can be extremely expensive. This paper will utilize a scheduling algorithm known as the Traveling Tournament problem that will generate a schedule that hopes to minimize traveling distance for each MLB team.

1 Problem Description

Have you ever wondered how much MLB teams are forced to travel over the course of a lengthy 162 game season? The answer to this question simply is too much. The question we look to answer through this study is whether or not these traveling distances can be minimized and if we can optimize schedules using this strategy. With that comes the Traveling Tournament Problem, an artificial intelligence problem that is part of the famous Traveling Salesman Problem family. The Traveling Salesman Problem in the simplest terms finds the minimum distance from a starting node to goal node. The Traveling Tournament Problem on the other hand aims to create an optimal scheduling using a round robin tournament scheduling style that minimizes the total distance of travel for all teams. A round robin tournament can best be defined as a tournament in which each team plays each other at least one and potentially more than once depending on set constraints and rules. The MLB essentially uses a round robin based scheduling approach when building their schedules.

1.1 MLB Schedule Structure

For context, the MLB has 30 teams, 2 leagues (15 teams each), and 6 divisions (5 teams each). The league is split in half to the American League and National League and then divided by three so that there are three five team divisions (East, Central, West). The 162 game schedule can be best understood by breaking it down into smaller pieces. Each team plays a total of 52 games against their division or 13 games against each team. These 13 games are played across 4 series with 7 of them being home and 6 being

away, or vice versa. Teams also play 64 games against other non-divisional teams in their league with the home to away ratio being even at 32 games a piece. They play 6 games against 6 of the teams and 7 games against the other 4 teams. Now in terms of the last 46 games, they are played against teams of the opposite league. Since the teams are in the opposite league, the matchup importance is not as prevalent as it doesn't effect playoff races or anything of sort. Each team plays one 3 game series against 14 of the 15 teams in the opposite league and one 4 game series against a "geographical rival". A geographical rival can best be explained as an outer league team that has a similar geographical location to your team. Great examples of this include the New York Yankees versus the New York Mets, the Chicago Cubs vs the Chicago White Sox, and the Minnesota Twins vs the Milwaukee Brewers. All of these teams either share cities or borders with each other, making them geographical rivals.

There are some basic scheduling rules or constraints that make this problem quite difficult, especially for the number of teams and games. Each teams plays 162 games, which are split into 52 series and 13 of them happen during weekends. This makes complete sense especially when it comes to fan attraction and revenue by scheduling series on weekends. Single series and four series home/away stands should be minimized. This is basically saying that teams should aim to play two or three series at home or away to minimize traveling. Single series could have teams traveling back and forth from home to away, which is clearly not optimal for minimizing traveling distance. The four series stands are not common because they present more of a fatigue or advantage problem. Teams traveling for four straight series find themselves all over the country for multiple weeks and playing four home series allows teams to have minimized travel and more rest which certainly is an advantage. Another constraint is that there are no doubleheaders in the original schedule and those are only used when games are postponed for weather or other reasons. For player safety, no team is allowed to play twenty consecutive games without rest and the season must be completed in a total of 178 to 183 days.

The complexity of scheduling for 162 games for 30 teams with all of these constraints is no easy task. This is why there are not many available sources that analyze full season MLB scheduling. Fortunately, we were able to find one source that does optimize a 2010 MLB schedule. Throughout the study we also will study smaller instances of the problem that may calculate optimal schedules for divisions or leagues instead of the whole league.

1.2 Significance of MLB Traveling Tournament Problem

There are plenty of reasons to why this problem is interesting and is being attempted to be solved in a multitude of ways. Major League Baseball scheduling is among the most complex schedules for all sports across the world given the size and number of games. If there was a proven method that completely optimized the schedules for the league, it can almost be guaranteed that the MLB would want to implement that method. A completely optimized schedule would reduce the MLB's carbon footprint and likely decrease player fatigue. Having to travel across the country for six months, would drive the MLB's carbon footprint up. Currently, the MLB actually has the least emissions per game among the United States major sports (National Football League, National Hockey League, Major League Soccer, National Basketball Association). This is likely

due to the MLB playing series for multiple days in the same city, whereas the rest of the leagues are traveling back and forth for each game (Sim, 2024). With an optimized schedule, this number could be even lower and the MLB could continue to be the most carbon friendly league in the United States. In terms of player health, playing a full 162 game schedule takes a huge toll on their physical health and mental health while also having to deal with increased injury risk and travel fatigue. Physically, playing baseball day after day for six months in a sport that requires constant explosive and powerful movements can not be understated. These type of movements repeated over and over can lead to fatigue, strains, and injuries over time. On the mental side, players must stay mentally focused at all times as baseball is a game of split second decisions and is truly a game against yourself. Slumps are extremely common in baseball and failure is more common than success and players have a constant pressure to perform. Travel and jet lag may be the worst symptom that players must battle through as constantly changing climates and time zones can disrupt sleep patterns and cause excess fatigue, slow recovery, and affect performance (Kovacs, 2023). If the MLB schedules were optimized to minimize traveling distance, players would most likely be healthier, have improved performance, and the league would likely reduce it's carbon emissions. These factors all have a major impact as to why the TTP is important to Major League Baseball.

2 Literature Review

The scheduling of Major League Baseball (MLB) games is a complex task, involving numerous constraints and objectives. One critical aspect of MLB scheduling is the minimization of traveling distance for teams throughout the season. To address this challenge, we turn to the Traveling Tournament Problem (TTP), a well-known problem in artificial intelligence. By using TTP strategies, we aim to construct an optimal MLB schedule that minimizes traveling distance while adhering to league scheduling constraints.

This literature review provides an overview of existing research on the TTP, focusing on its application to MLB scheduling. Our objective is to analyze TTP methods and algorithms, identify knowledge gaps, and propose strategies for building efficient MLB schedules. This review seeks to find the most successful TTP methods that lead to optimal MLB schedules.

2.1 Traveling Tournament Problem Constraints

The TTP is based on scheduling a double round-robin tournament that satisfies certain constraints to find minimized distances. These constraints are extremely important to follow or else sub-optimal solutions may result. The first constraint is that each team must play every other team twice. Following that, a team can't play the same team consecutively. This makes sense because playing a team multiple times consecutively is a significant net positive when trying to minimize distance and also never happens in sports. Thirdly, visiting teams on a road trip travel to their next destination and don't travel back home. Road trips are often essential to help minimize travel distance for teams. The final constraint is that teams will not play more than two series at, or

away from home in a row (Chatterje, 2012).

2.2 Analyzing Current Research

One method for solving the TTP uses simulated annealing. This method splits the constraints into hard and soft constraints. A hard constraint can best be defined as a constraint that is always satisfied, whereas a soft constraint can sometimes be violated (Anagnostopoulos et al., 2006). If too many constraints are violated, we could be driven away from truly optimal solutions which is a limitation to this study. The complexity of the TTP makes it an extremely hard problem to solve and gaps may form when creating solutions for it (Thielen, 2010). The algorithm is performed in a neighborhood with $O(n^3)$ size and dynamically adjusts its objective function to balance the time spent in reasonable and unreasonable regions of the neighborhood. It does this while also attempting to escape local minima (Van Hentenryck, n.d). The results of one of these studies show that for even numbered team tournaments from four to sixteen, the total distance was improved or matched in all cases (Anagnostopoulos et al, 2006). The other study using a similar simulated annealing algorithm found major decreases in total traveling distance for tournament sizes from twelve to twenty teams (Van Hentenryck, n.d).

Another study covered simulated annealing but had a different approach and added a hill-climbing aspect to the algorithm. This implementation divided the problem into a timetable space and a team assignment space. Simulated annealing was used on the timetable space and hill-climbing was used on the team assignment space. The simulated annealing aspect searches for team assignments that have a higher chance of outputting better schedules and the hill-climbing aspect generates and improves team assignments by searching for assignments that result in smaller total distances for the given timetable (Lim et al., 2006). The results of this approach showed a 7.45 percent gap between the previous minimum distance and the new minimum distance for tournament sizes four to twenty. All of the attempts to solve the TTP with some version of simulated annealing in the research conducted showed clear reductions in the total distance traveled for teams competing in the tournaments.

This problem can also be approached by finding optimal team match-ups and then super matches from those original match-ups. Optimal team match-ups are found using a minimum maximal matching strategy that takes all of the edges from a graph and takes the smallest pairing (Chatterje, 2012). The super matches are formed using four teams matched from the minimum maximal strategy and swapping them until they all play each other (Chatterje, 2012). This should find four teams that are relatively close in location and have them play each other back-to-back so that travel distance is minimized. This strategy found improvements in all tournaments for size $n \leq 32$ teams and was tested on the Indian Premier Soccer League (8 teams) and found a fifteen percent decrease in traveling distance. This is quite significant and shows this method to be effective in solving the TTP.

Another common strategy used to solve the TTP was found in a couple of resources that used integer programming. In one of the sources, integer programming was combined with constraint programming to come to a solution. The constraint programming techniques are used to generate variables that are then combined with integer programming techniques to find optimal solutions (Easton et al., n.d). This approach may not

be the most successful as it was only performed in the National League (half of the MLB) and was only found optimal in cases with subsets of four and six teams. On the other hand, an approach that combined integer programming with a local search improved the best solution for ten out of fifteen teams per league and saw anywhere between a 0.1-3.3 percent difference in total distance. In this implementation, integer programming found the driving distance for each day. This was calculated and then divided into a home/away optimizing pattern (Goerigk, 2016). While this occurs, a local search would be conducted that selects the most optimal node to visit for each iteration. These two methods were repeated and combined for each instance of a new match and were continually optimized for the entirety of the scheduling (Goerigk, 2016). Given these two studies, we can find examples of both success and failure using integer programming to solve the TTP.

The final approach to solving the TTP that was studied used a variable neighborhood search. The steps to solving this problem included using a distance matrix to find the distances from each MLB team's stadium to every other team's stadium. The algorithm created multiple neighborhoods to cover each team's location across the United States. The problem is solved by taking a team in a certain neighborhood and having it select the best solution in its current neighborhood and then either continuing to take the best solution in the current neighborhood or a different neighborhood until completion (Liang et al., 2021). This may be the best method found to solve the TTP using an MLB schedule as twenty-two out of thirty teams found minimized distances and the average gap between total distances was 6.02 percent. This was also the most recent study of the ones covered in this review, making it a strong candidate for our implementation of the TTP on an MLB schedule.

In conclusion, the Traveling Tournament Problem (TTP) emerges as a valuable asset in creating optimal schedules for Major League Baseball (MLB) teams. Through this literature review, we have explored how the TTP, can be used to minimize traveling distances and enhance the overall efficiency of MLB schedules.

By leveraging various methodologies, such as simulated annealing or variable neighborhood search, researchers have demonstrated the effectiveness of TTP-inspired approaches in MLB scheduling or scheduling in general. These algorithms could offer the MLB the opportunity to optimize its schedules while considering factors such as player fatigue and competition balance.

Moving forward, continued research and innovation in TTP-based scheduling hold promise for further improving MLB schedules. Optimizing an MLB schedule could not only minimize traveling distances but also enhance player performance and fan engagement.

In essence, the TTP could serve as a valuable tool for the MLB to construct schedules that strike a balance between efficiency and competitiveness. By using TTP-inspired approaches, the MLB can unlock new opportunities to enhance the quality and competitiveness of their schedules, ultimately improving the experience for players, fans, and others.

3 Approach to Solving the TTP

Given that some of the research analyzed above didn't produce consistent results, we decided to pivot to a different method to solve the TTP proposed by Douglas Moody. Moody is a member of City University of New York's Graduate Center and School of Computer Science and has done lots of work on the TTP. The reasoning behind choosing Moody's approach, is that he is one of the only TTP researchers that attempts to perform the problem on a full MLB schedule and follows MLB scheduling guidelines in his solution. Moody's approach is two phased involving tiling and a local search phase. The tiling phase creates initial solutions and the local search then removes the hard constraint violations to improve the quality of the solution (Moody, 2010). To understand more how the problem works, we will describe more in-depth how the tiling phase and local search phase work.

3.1 Tiling Phase

Tiling can best be explained as the road trips that teams are taking in the TTP. Tiles contain blocks representing individual contests where one tile can be equated to a road trip of three opponents with three blocks. A teams' entire schedule can be described as a sequence of tiles with home stands being spaces between them. Figure 1 below is an image showing this representation more clearly (Moody, 2010).

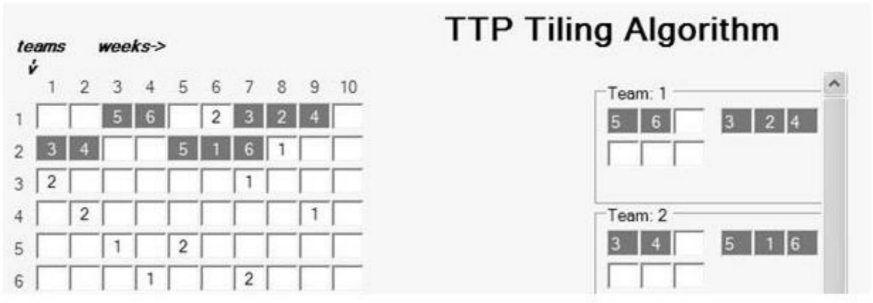

Figure 1: TTP Tiling Algorithm

The reasoning for using tiles is to minimize the distance for a team while respecting the constraints that involve the rest of the teams in the league. The tiles are placed in a scheduling format of n rows that represent each team and 2n-2 columns representing weeks in a season. As tiles are placed in the scheduling grid, the rest of the grid slowly fills in to maintain the scheduling consistency. When tiles are unable to be placed, they break into their individual blocks and are placed according to their constraints. If a block fails to place, we relax the constraints and allow for some solution to be output (Moody, 2010).

Tiles are created for each team using a Minimum Spanning Tree. The creation begins by selecting a team to represent the root node and the edges to the rest of the nodes in the tree represent the stadium distances. The tree is searched for the smallest edge and that node is connected to the root node. Next, the team that is closest to the root team or it's first opponent is added to the tree until all of the teams have been

accounted for (Moody, 2019). Figure two shows an example of this on a small sample of six teams from Moody's paper.

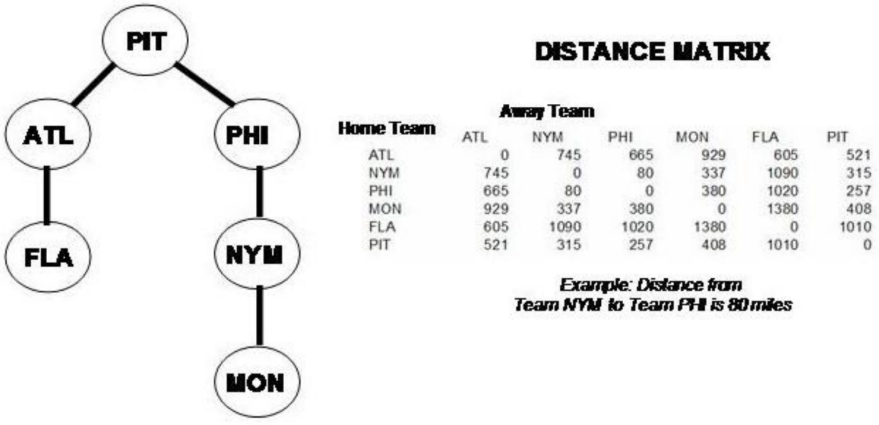

Figure 2: Minimum Spanning Tree for Pittsburgh Pirates in 6 Team Sample

After each team has it's own Minimum Spanning Tree, the tree is then collapsed to create tiles. The collapsing merges child nodes into their parent. The tiles are created once a parent has 3 teams as child nodes and the parent uses a greedy approach to decide which children is used first. The tiling process creates three block tiles that are given a cost that measures the impact on the objective function of dividing the tiles. This cost can be calculated by adding the total miles of the round trip to each opponent in the tile and subtracting that by the actual distance to be traveled within tile teams. This ultimate cost helps decided where the tiles will be located in the scheduling grid (Moody, 2019).

In terms of tile placement, the highest cost tiles are placed first (assuming it doesn't violate constraints). If constraints are violated, the teams are rotated in the tile until it is able to be placed. Once all of the tiles are placed and no more can be placed, we divide the remaining tiles into individual games to be placed inside of the schedule. In order to do this, the constraints home and away rules must be relaxed. If a game can't be placed the algorithm back tracks until completion, even if optimization is slightly reduced (Moody, 2019).

3.2 Local Search Phase

After the tiling phase, the algorithm enters a local search phase. This phase is implemented to remove any violations of hard constraints. When all tiles are able to be placed, our solutions are near optimal. If this is not the case, the solution isn't as great and the singular games that are placed drives the miles higher. The objective of the local search is to swap placements of sets of games while maintaining the objective function of the solution. There are few different types of swaps that can take place including a home/away swap, round swap, and partial round swap. A home/away swap swaps two games the involve the same two teams where each team is home in one game and away in the other. A round swap takes all games from a two week period and moves them around. Finally, the partial swap, takes 2 games in different weeks and add teams playing these 2 teams to the swap set. After this, the process continues until

all teams in the sample size are in the swap set and all games are moved between weeks until completion (Moody, 2019).

After performing the local search, schedule improvement is analyzed between the potential swaps and the swap with the largest improvement is implemented to find a new schedule distance. The process is redone until no improvement is found and the best schedule is selected (Moody, 2019).

We will use Moody's Tiling and Local Search Algorithm on a full MLB schedule to find the total minimized traveling distance for the entire league. Our solution will be compared to MLB schedules over the years to see how effective this version of the TTP is compared to others mentioned in our Literature Review section.

4 Experiment Design

Now that we have decided which algorithm that will be used to complete our research, it is time to design our experiment. To test the optimality of Moody's tiling and local search algorithm we are going to run his algorithm on an entire MLB schedule with the total amount of miles traveled for all teams being the solution in each instance over the past decade.

4.1 RobinX

Moody's research didn't include the exact code used to solve the problem, but we have found files through a TTP problem website that represents his results and shows his methods and processes. This website is called RobinX and it stores queries of TTP instances in XML format. RobinX stores hundreds of different instances that represent differnt approaches to solve the TTP. This allows for TTP researchers across the world to compare answers and methods, ultimately looking for the best solutions to different TTP problems. Essentially, Moody solved the problem using his algorithm and uploaded his solution to RobinX for all to see his solution and what constraints and inputs he used to arrive to this conclusion. We were unable to test the actual code itself due to access, but in RobinX we are given the exact algorithms and processes that were implemented to derive answers and are able to use that to explain the problem and compare solutions.

4.2 Baseball Savant

Baseball Savant is a MLB database that stores all types of statistics and numbers. We will be using Baseball Savant to help calculate by hand the total number of miles traveled per MLB season. The database shows how many miles each team traveled per season and we will manually add up all 30 teams for the last 10 years and compare it to the solution we find. Figure 3 shows a travel map from Baseball Savant for the 2024 MLB season.

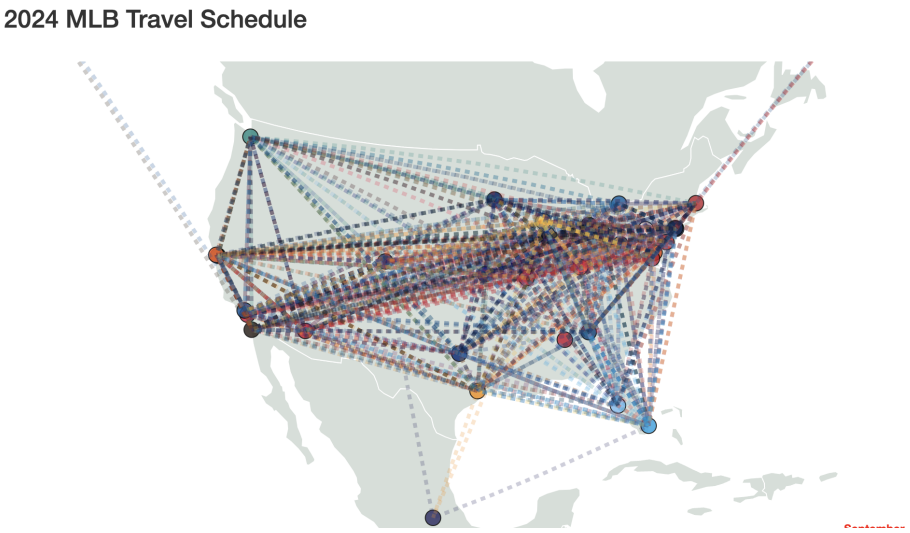

Figure 3: 2024 MLB Travel Map

4.3 Setting up the Experiment

Now that it has been determined that the goal of this study is to analyze the results from our solution versus MLB schedules from the past decade, we can break down step by step how we will get our answer from Moody's algorithm.

From the XML file of Moody's solution, we are given instances of each game in the season, a 30x30 distance matrix of each distance from stadium to stadium, the teams, number of series in the season, basic scheduling constraints, and separation constraints.

The approach to solving this problem starts by defining each game that will be in played in the schedule and inputting them line by line until all games are defined. Next, the distance matrix is input that shows the distance from each team's stadium to every other stadium in the league. The distance matrix is initially formatted so you can easily find the distance from team X to team Y if you find their intersection. In the XML form, the matrix is converted into an extremely long list. In the lines representing the schedules and distances, the teams are not defined yet and are shown as "team 1" or "team 25". This is why in the file there are lines that assign each team to the number they represent. This allows for easy viewing of which team is which when trying to decipher the lines. Series slots are then defined for all 52 series that each team plays in a season. This makes it easy to determine which teams are competing and during what time period of the season. The basic scheduling constraints that are used can best be defined as capacity constraints. These constraints assure that no team plays more than 3 consecutive series at home or on the road. Finally, the separation constraints assure that a team doesn't play back-to-back series against each other.

In the implementation we are using, all 2,430 games are scheduled. Next, the distance matrix is then implemented in line format and it can be seen how far each stadium is from each other. Figures four and five below show examples of how these lines are formatted in the instance file.

```
<game noHome="0" team1="26" team2="23"/>
<game noHome="0" team1="23" team2="26"/>
<game noHome="0" team1="27" team2="23"/>
<game noHome="0" team1="27" team2="23"/>
<game noHome="0" team1="23" team2="27"/>
<game noHome="0" team1="28" team2="23"/>
```

Figure 4: Example of Game Instances in XML Format

```
<distance dist="541" team1="27" team2="8"/>
<distance dist="242" team1="27" team2="9"/>
<distance dist="727" team1="27" team2="3"/>
<distance dist="1063" team1="27" team2="5"/>
<distance dist="1010" team1="27" team2="21"/>
```

Figure 5: Example of Distance Instances in XML Format

4.4 Limitations to Experiment

Due to the TTP not being the most common Artificial Intelligence problem, there are some limitations that must be noted from our study. There are a couple of MLB scheduling constraints that are not used to compute the answer to our problem. This includes there being 13 weekend series per team and that no team can play on twenty consecutive days. Also, Moody's computations were done on a schedules from 2010 where the MLB scheduling constraints might have been slightly different than they are currently. The total teams and games remain the same as current day, so we are not extremely worried about that playing a large role in determining if this algorithm generates optimal MLB schedules. The 2010 comparison will be our most important to study, but we will consider all schedules since then.

4.5 Process of Analyzing Results

In order to best analyze our results from the experiment, we will build tables that compare the solutions from our algorithm and the MLB's scheduling algorithm. We have no clue what the MLB's approach to creating MLB schedules is, but we want our solutions to best theirs in order to prove that MLB schedules can be optimized and could be a beneficiary for the league. These visuals should allow for easily understanding which approach is more optimal for minimizing traveling distance.

5 Analysis of Results

To test the experiment we calculated the total miles traveled for every team combined for the last 14 seasons and compared this to Moody's best solution. Considering our "best solution" using the algorithm was performed on a 2010 schedule, we looked at the total distance from current day 2024 to 2010. It must be noted that 2020 will be excluded from this experiment due to the COVID-19 pandemic. During this season the MLB was only able to play 60 games compared to the usual 162 making it irrelevant for this study. We utilized Baseball Savant to manually calculate the total distance

traveled for all teams in the league and cross referenced the Moody solution. It was found that over the 13 years of data, that our solution performed better than the MLB schedule did. Figure six below showcases the comparison year-by-year and it can be clearly seen that our algorithm outperformed the MLB's.

Instance: MLB_DIV5	Teams: 30	# in Total Miles
Year	MLB Algorithm	Moody's Best Solution
2024	1,061,373	959,944
2023	1,074,621	959,944
2022	999,933	959,944
2021	981,419	959,944
2020	EXCLUDED	EXCLUDED
2019	1,057,476	959,944
2018	925,357	959,944
2017	973,398	959,944
2016	988,759	959,944
2015	898,548	959,944
2014	997,559	959,944
2013	984,132	959,944
2012	962,504	959,944
2011	988,546	959,944
2010	986,012	959,944

Figure 6: Total Miles Comparison (Our Result vs MLB Result)

While our algorithm produced a more optimal solution in 11 out of 13 instances, it is also important to truly look at how much better his solutions were by calculating the average gap between victories. It can be computed by finding the differences in solutions, adding all of the differences up, and finally dividing them all by the 11 instances. It was found that on average, our algorithm produced better outcomes by 48,764 miles. This number is notable because when looking at the total miles traveled per team on the Baseball Savant website, the team that travels the most per season is typically somewhere between 52,000 and 47,000 miles. The average distance among all teams lies within the 32,000 to 28,000 range, which almost makes our solution better by 2 teams. Hypothetically, we could use our solution on an expanded MLB schedule with two more teams and still be tied or better than the MLB solutions.

To analyze our algorithm we also believed it was important to compare our results to the best results that were found in our review of other TTP literature. The best results from our literature review were found using a variable neighborhood search. This TTP was also performed on a Major League Baseball schedule, but used a subset of 22 of the 30 teams instead of all them in our study. That study found an average gap of 6.02 percent between solutions. Calculating the average gap percentile of our solution, we find an average gap of 3.18 percent. It is unclear which study optimizes schedules better due to the differing number of teams, but either way both improve the total distances that MLB teams are forced to travel in a given season.

Given that our algorithm performs better in most cases, we can conclude that MLB

schedules can be optimized using a tiling and local search approach. This also means that there is a sufficient method of scheduling that can work towards reducing player fatigue and carbon emissions. Reducing total travel by an average of 48,764 miles using our algorithm is a drastic amount and reduces travel per team by about 1,625 miles. For reference, 1,625 miles would save the Kansas City Royals a whole trip to visit the Los Angeles Dodgers (1,626 mile distance between the two). This may not seem super significant, but long distance match-ups like these only happen once per season. Another important perspective is to look how this difference effects divisional match-ups. Teams play thirteen games against all four teams in their division meaning four traveling periods. If we look at division rival St. Louis Cardinals and Chicago Cubs who have a 304 mile distance between them, this algorithm saves teams almost 5 trips so that each time these teams play the miles aren't accumulated. Overall, our algorithm using Moody's tiling and local search phases would help the league financially, the climate, and most importantly player safety and recovery.

6 Conclusion

To conclude this project, it is safe to say that the TTP is one of the most challenging problems to solve in all of Artificial Intelligence. Creating a 162 game schedule for 30 teams that is also optimal requires a lot of work. From our implementation, we can conclude that Moody's tiling and local search algorithm performs better than the MLB's in about 85 percent of seasons dating back to 2010. Due to this, MLB scheduling can be optimized using this approach to minimize the total traveling distance for teams. With the distance optimization, we can expect improved player performance due to less travel fatigue. This can also be said about a carbon emissions decrease from less plane travel. Along with these benefits, the MLB will save money overall from having less travel expenses per team. In terms of future research, there are always more optimal solutions available, we believe it would be beneficial to attempt this implementation using other popular Artificial Intelligence algorithms until a significantly better solution is found. But for now, we can confidently say that our solution performs better than the MLB's and could be something that the MLB needs to take into account before assembling schedules each year.

References

[AMHV06] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. Technical report, Brown University, 2006.

[ENTNA] Kelly Easton, George Namhauser, and Michael Trick. The traveling tournament problem description and benchmarks. Technical report, Georgia Institute of Technology and Carnegie Mellon, N/A.

[GW16] Marc Goerigk and Stephan Westphal. A combined local search and integer programming approach to the traveling tournament problem. In *Annals of Operations Research*, 2016.

[htta] *Baseball Savant, https://baseballsavant.mlb.com/.*

[httb] *RobinX, https://robinxval.ugent.be/RobinX/index.php.*

[HVNA] Pascal Van Hentenryck and Yannis Vergados. Traveling tournament scheduling: A systematic evaluation of simulated annealing. Technical report, Brown University, N/A.

[Kov23] Mark Kovacs. Kovacs institute, 2023. The Path To 162: One Of The Hardest Things To Do in Baseball.

[LLCC21] Yun-Chia Liang, Yen-Yu Lin, Angela Hsiang-Ling Chen, and Wei-Sheng Chen. Variable neighborhood search for major league baseball scheduling problem. *Sustainability*, 2021.

[LRZ06] A. Lim, B. Rodrigues, and X. Zhang. A simulated annealing and hill-climbing algorithm for the traveling tournament problem. In *European Journal of Operational Research*. Association of European Operational Research Societies, 2006.

[PAT10] *An efficient and robust approach to generate high quality solutions for the Traveling Tournament Problem*, 2010.

[Sim24] Josh Sim. NFL averages highest per-game carbon footprint of major US sports leagues. *SportsPro Media*, 2024.

[TW10] Clemens Thielen and Stephan Westphal. Complexity of the traveling tournament problem. Technical report, University of Kaiserslauten, 2010.

[ENTNA] [Sim24] [PAT10] [LRZ06] [httb] [htta] [TW10] [AMHV06] [LLCC21] [HVNA] [PAT10] [GW16] [Kov23]