Balancing the Traveling Tournament Problem for Weekday and Weekend Games

Richard Hoshino
Quest University Canada
Squamish, British Columbia, Canada

Ken-ichi Kawarabayashi
National Institute of Informatics, Tokyo
JST-ERATO Kawarabayashi Large Graph Project

Abstract

The Traveling Tournament Problem (TTP) is a well-known NP-complete problem in sports scheduling that was inspired by the application of optimizing schedules for Major League Baseball to reduce total team travel. The techniques and heuristics from the n-team TTP can be extended to optimize the scheduling of other sports leagues, such as the Nippon Professional Baseball (NPB) league in Japan. In this paper, we describe the additional scheduling constraints required by the NPB league, such as the requirement that each team play the same number of weekend home games, weekday home games, weekend road games, and weekday road games. We fully solve this TTP-variant for the case $n = 6$, and conclude the paper by presenting the official 2013 NPB Central League Schedule, where we helped this Japanese baseball league reduce total team travel by over six thousand kilometres.

Introduction

Nippon Professional Baseball (NPB) is Japan's largest and most well-known professional sports league, with over 22 million fans each season, and annual revenues topping one billion U.S. dollars. In terms of actual attendance, the NPB ranks second in the world among all professional sports leagues, ahead of the National Football League, the National Basketball Association, and the National Hockey League.

The NPB is split into the six-team Pacific League and the six-team Central League. Each team plays 144 games during the regular season, with 120 *intra*-league games (against teams from their own league) and 24 *inter*-league games (against teams from the other league). To complete these $\frac{1}{2} \times 12 \times 144 = 864$ games, the teams travel long distances from city to city, primarily by airplane or bullet-train. During the 2012 regular season, these twelve teams traveled a total of $280,000$ kilometres (Hesse 2012), the equivalent of seven trips around the Earth.

Sports scheduling has emerged as a growing field of AI research in the past decade (Kendall et al. 2010), especially since the introduction of the Traveling Tournament Problem (TTP) by the head schedulers of Major League Baseball

(Easton, Nemhauser, and Trick 2001). The TTP is an NP-complete problem involving an n-team sports league, where the objective is to produce a double round-robin schedule that minimizes the total distance traveled by all n teams. The TTP is the exact framework for many sports leagues around the world, such as college basketball in the USA and soccer in South America, where each pair of teams plays twice, with one game held at each team's home arena / stadium.

Our research program (Hoshino and Kawarabayashi 2013) was motivated by our hope that graph theory could help the NPB become more efficient and effective, to save money, time, and greenhouse gas emissions. By reducing the NPB scheduling problem to a shortest-path problem, we determined the distance-optimal inter-league schedule (Hoshino and Kawarabayashi 2011a) as well as the distance-optimal intra-league schedule (Hoshino and Kawarabayashi 2011c) for both the Pacific and Central Leagues. Combined, our proposed 864-game regular season tournament requires $210,000$ kilometres of total team travel, representing a potential reduction of 25%. After our results were published, the NPB invited the authors to help design the Central League's 2013 intra-league schedule (Hesse 2012).

In this paper, we describe how we solved this scheduling problem by adapting the TTP to fit the exact specifications of this Japanese league, including the strict requirement that every team play the same number of home and away games on weekdays and weekends – this "revenue-balancing" rule is in place because most NPB stadiums are half-empty on Tuesdays but filled to capacity on Saturdays. Such a requirement is a natural condition in equitably scheduling a tournament: if there are several "big-money" weekends that coincide with major holidays (e.g. Memorial Day and Independence Day), then each team should be assigned half of these games at home, to ensure an equitable distribution.

This paper proceeds as follows: we first describe the n-team Traveling Tournament Problem, and then define the variant specific to the NPB context. We then apply Dijkstra's shortest path algorithm to fully solve this TTP-variant for the case $n = 6$, and generate the distance-optimal schedule for the NPB Central League. We conclude the paper by discussing our consultation work for the NPB, and compare the 2012 and 2013 Central League schedules: this year's schedule requires 12 fewer trips, and the total travel distance is reduced by over $6,000$ kilometres.

The Traveling Tournament Problem

Let there be n teams in a sports league, where n is even. Let D be the $n \times n$ *distance matrix*, where entry $D_{i,j}$ is the distance between the home stadiums of teams t_i and t_j. By definition, $D_{i,j} = D_{j,i}$ for all $1 \leq i, j \leq n$, and all diagonal entries $D_{i,i}$ are zero.

Team	t_1	t_2	t_3	t_4	t_5	t_6
t_1	0	323	488	808	827	829
t_2	323	0	195	515	534	536
t_3	488	195	0	334	353	355
t_4	808	515	334	0	37	35
t_5	827	534	353	37	0	7
t_6	829	536	355	35	7	0

Table 1: Distance matrix for the NPB Central League (in km)

For example, the distance matrix for the NPB Central League is given in Table 1. The six teams (Hiroshima, Hanshin, Chunichi, Yokohama, Yomiuri, Tokyo) are labelled t_1 to t_6, respectively. The locations of each team's home stadium is given in Figure 1 below.
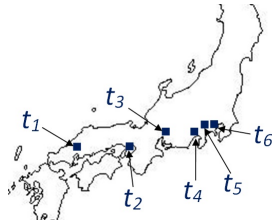

Figure 1: The Central League teams on a map of Japan.

In the TTP, a double round-robin schedule is sought, where each pair of teams plays twice during a tournament lasting $2(n - 1)$ days, with each team having one game scheduled per day. As the context for our paper is baseball, we will now use *sets* rather than *days* to refer to the length of a tournament. Unlike other sports (e.g. football, soccer, hockey, basketball) where a team visits another city to play a single match, professional baseball leagues always involve a team visiting another city to play three or four games. To avoid any confusion, we will now re-define the TTP to the scheduling of $2(n - 1)$ sets, where each set consists of a fixed number of games played on consecutive days.

The objective is to minimize the total distance traveled by the n teams, with the requirement that each team begins the tournament at home, and returns home after having played their last away set. When a team is scheduled for a road trip consisting of multiple away sets, the team doesn't return to their home city but rather proceeds directly to their next away venue. In many ways, the TTP is a variant of the well-known Traveling Salesman Problem, asking for an optimal schedule linking venues that are close to one another.

In the (standard) TTP, we have the following constraints:

(a) The *each-venue* condition: Each pair of teams must play two sets, once in each other's home venue.

(b) The *at-most-three* condition: No team may have a home stand or road trip lasting more than three sets.

(c) The *no-repeat* condition: A team cannot play against the same opponent in two consecutive sets.

To illustrate, Table 2 lists the first ten sets of the NPB Central League for the 2013 season, where home teams are marked in bold. We see that this is a double round-robin schedule satisfying all of the above conditions. In the NPB, each set consists of three games.

Team	1	2	3	4	5	6	7	8	9	10
t_1	t_5	t_6	t_2	t_4	t_3	t_4	t_5	t_6	t_3	t_2
t_2	t_6	t_3	t_1	t_5	t_4	t_5	t_6	t_3	t_4	t_1
t_3	t_4	t_2	t_5	t_6	t_1	t_6	t_4	t_2	t_1	t_5
t_4	t_3	t_5	t_6	t_1	t_2	t_1	t_3	t_5	t_2	t_6
t_5	t_1	t_4	t_3	t_2	t_6	t_2	t_1	t_4	t_6	t_3
t_6	t_2	t_1	t_4	t_3	t_5	t_3	t_2	t_1	t_5	t_4

Table 2: First ten sets of the 2013 Central League Schedule.

Define a *block* to be a feasible solution of the TTP, i.e., a tournament lasting $2(n - 1)$ sets. We say that a block consists of two *rounds*, with the first round being the first $n - 1$ sets and the second round being the last $n - 1$ sets. In a multi-round tournament with k blocks, there are $2k$ rounds and $2k(n - 1)$ sets. In the NPB, each of the $n = 6$ teams play $2k = 8$ rounds of intra-league matches, corresponding to $2k(n - 1) = 40$ sets of three games.

In addition to the requirement of multiple blocks ($k = 4$ for the NPB, compared to $k = 1$ for the TTP), scheduling an NPB tournament requires several additional constraints. First, the NPB requires each pair of teams to play exactly one set during each $(n-1)$-set round, which is stronger than the *each-venue* condition that only requires the two sets to be played sometime during a two-round block. As in the *each-venue* condition, these two sets must be played at different locations, with one set held at each team's home venue. We note that this condition is similar but not identical to the more stringent "mirrored" condition of Latin American soccer leagues (Ribeiro and Urrutia 2004), which has the rule that if team i hosts team j in set s (where $1 \leq s \leq n - 1$), then team j hosts team i in set $s + n - 1$.

Secondly, the NPB schedule requires a balance in the number of home and away sets played by each team at any point in the season. More formally, for each ordered pair (i, s) with $1 \leq i \leq n$ and $1 \leq s \leq 2k(n - 1)$, define $H_{i,s}$ and $R_{i,s}$ to be the number of home and away sets played by team i within the first s sets. (By definition, $|H_{i,s} + R_{i,s}| = s$.) The NPB requires that $|H_{i,s} - R_{i,s}| \leq 2$ for all pairs (i, s). For example, under this requirement a team cannot start or end a season with three consecutive home sets, ensuring that no team gains a momentum-increasing advantage at a key point in the season.

When we met with the NPB head scheduler, we learned that the league tries to avoid three-set home stands and three-set road trips at all costs, as they have found in the past that long home stands lead to diminished attendance, while long road trips cause player fatigue.

And most importantly, each team must play the same number of weekend home games for reasons of competitive fairness and revenue balance. NPB teams don't play on Monday, and each three-game set takes place on weekdays (Tuesday, Wednesday, Thursday) or on weekends (Friday, Saturday, Sunday). The 40 intra-league sets are slotted so that there are 20 weekday sets and 20 weekend sets. The NPB requires each team to play 10 weekend home sets, 10 weekday home sets, 10 weekend road sets, and 10 weekday road sets. This is denoted by the four-tuple $(10, 10, 10, 10)$.

Block #1 and Block #4 consist of five weekend sets and five weekday sets. Due to a five-and-a-half week break for inter-league play (between sets 13 and 14) as well as a half-week break for the All-Star Game (between sets 21 and 22), Block #2 has six weekends and Block #3 has four weekends. NPB rules require a specific four-tuple structure within each block, as described in condition (g) below.

To summarize, modeling the NPB scheduling problem requires four additional constraints:

(d) The *each-round* condition: Each pair of teams must play exactly once per round, with their matches in rounds $2t-1$ and $2t$ taking place at different venues (for all $1 \le t \le k$).

(e) The *diff-two* condition: $|H_{i,s} - R_{i,s}| \le 2$ for all (i, s) with $1 \le i \le n$ and $1 \le s \le 2k(n-1)$.

(f) The *at-most-two* condition: No team may have a home stand or road trip lasting more than two sets.

(g) The *weekday-weekend* condition: in Blocks #1 and #4, each team must have one $(3, 2, 2, 3)$ four-tuple and one $(2, 3, 3, 2)$ four-tuple. In Block #2, each team must have a $(3, 2, 3, 2)$ four-tuple, and in Block #3, each team must have a $(2, 3, 2, 3)$ four-tuple.

By definition, conditions (a) and (b) are made redundant by conditions (d) and (f), respectively.

We now present an algorithm for solving this multi-round variant of the TTP, by reformulating it as a shortest path problem on a directed graph. The first part of our algorithm handles conditions (a) through (e), and the full details appear in a previous paper (Hoshino and Kawarabayashi 2011c). Here, we make a slight fix to our Dijkstra-based algorithm by replacing the *at-most-three* condition with *at-most-two*, thus handling the first six conditions. The second part of our algorithm is completely new, and addresses condition (g).

Shortest-Path Algorithm: conditions (a)-(f)

Our idea is to create a source node and a sink node and link them to numerous vertices in a graph whose (weighted) edges represent the possible blocks that can appear in an optimal schedule. We then apply Dijkstra's Algorithm to find the path of minimum weight between the source and the sink, which is a well-known $O(|V| \log |V| + |E|)$ graph search algorithm that can be applied to any graph or digraph with non-negative edge weights.

By definition, a *block* is a two-round tournament schedule satisfying the above conditions, with each of the n teams playing $2(n - 1)$ sets of games. To solve our NPB scheduling problem, we first enumerate the complete set of blocks that can appear in a distance-optimal tournament. We then

introduce a simple "concatenation matrix" to check whether two pre-computed blocks can be joined together to form a multi-block schedule, without violating any of the scheduling constraints. As we will explain, to determine whether two (feasible) blocks B_1 and B_2 can be concatenated, it suffices to check just the last two columns of B_1 and the first two columns of B_2.

Each column of a block represents a set consisting of $\frac{n}{2}$ different *matches*, with each match specifying the two teams as well as the stadium/venue. Thus, a match identifies the home team and away team, not just each team's opponent. For any column, there are $\binom{n}{n/2}$ ways to select the home teams. Also there are $\binom{n}{n/2} \cdot \left(\frac{n}{2}\right)!$ ways to specify the matches of any column, since there are $\left(\frac{n}{2}\right)!$ ways to map any choice of the $\frac{n}{2}$ home teams to the unselected $\frac{n}{2}$ away teams to decide the set of $\frac{n}{2}$ matches. Hence, there are $m = \binom{n}{n/2}^2 \cdot \left(\frac{n}{2}\right)!$ different ways we can specify the matches of the first column *and* the home teams of the second column. For $n = 6$, we have $m = \binom{6}{3}^2 \times 3! = 2400$.

There are m ways that the first two columns of a block can be chosen as described above, with the first column listing matches and the second column listing home teams. Now use any method, such as a lexicographic ordering, to index these m options with the integers from 1 to m. By symmetry, there are m different ways we can specify the last two columns of a block, with the last column listing matches and the second-last column listing home teams. Thus, we use the same scheme to index these m options. To avoid confusion, we write the home teams column in binary form, with **1** representing a home game and 0 representing an away game.

For example, $(t_2, \boldsymbol{t_1}, \boldsymbol{t_5}, \boldsymbol{t_6}, t_3, t_4)^T$ is one of the 120 possible options for the matches column, and $(\mathbf{1}, 0, 0, \mathbf{1}, 0, \mathbf{1})^T$ is one of the 20 possibilities for the home teams column. We remark that if we listed the column of *opponents* rather than the column of matches, there would be only $\frac{120}{2^3} = 15$ unique columns, corresponding to the 15 perfect matchings of the complete graph K_6.

There exists some integer q (with $1 \le q \le 2400$) that is the index of the instance where the home teams column is $(\mathbf{1}, 0, 0, \mathbf{1}, 0, \mathbf{1})^T$ and the matches column is $(t_2, \boldsymbol{t_1}, \boldsymbol{t_5}, \boldsymbol{t_6}, t_3, t_4)^T$. Similarly, there exists some r (with $1 \le r \le 2400$) that is the index of the instance where the two columns are $(t_5, t_6, \boldsymbol{t_4}, t_3, \boldsymbol{t_1}, \boldsymbol{t_2})^T$ and $(\mathbf{1}, \mathbf{1}, 0, \mathbf{1}, 0, 0)^T$. In the block given in Table 2, the last two columns have index q and the first two columns have index r.

For each pair (u_1, u_2), with $1 \le u_1, u_2 \le m$, define C_{u_2, u_1} to be the $n \times 4$ *concatenation matrix* where the first two columns list the home teams and matches with index u_2, and the next two columns list the matches and home teams with index u_1. For the indices q and r from the previous paragraph, we have

$$C_{q,r} = \begin{bmatrix} \mathbf{1} & t_2 & t_5 & \mathbf{1} \\ 0 & \boldsymbol{t_1} & t_6 & \mathbf{1} \\ 0 & \boldsymbol{t_5} & \boldsymbol{t_4} & 0 \\ \mathbf{1} & \boldsymbol{t_6} & t_3 & \mathbf{1} \\ 0 & t_3 & \boldsymbol{t_1} & 0 \\ \mathbf{1} & t_4 & \boldsymbol{t_2} & 0 \end{bmatrix}.$$

Note that $C_{q,r}$ has no row with three consecutive home sets, no row with three consecutive away sets, and no row with the same opponent appearing in Columns 2 and 3. As we describe in Theorem 1 below, these three properties are a necessary and sufficient condition for whether two feasible blocks can be concatenated to produce a multi-block schedule satisfying all the conditions from (a) to (f). Therefore, we can simply create four copies of the block in Table 1, concatenate them together to form a 40-set schedule, and the resulting tournament will automatically satisfy the first six scheduling constraints. (Alas, it does not satisfy condition (g), the weekday-weekend balancing constraint.)

Before we proceed with Theorem 1, let us explain the role of the concatenation matrix in the construction of our directed graph. Let G consist of a source vertex v_{start}, a sink vertex v_{end}, and vertices $x_{t,u}$ and $y_{t,u}$ defined for each $1 \leq t \leq k$ and $1 \leq u \leq m$.
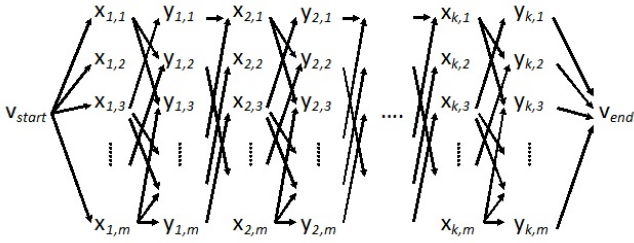

Figure 2: Converting NPB scheduling into a shortest path problem.

We now describe how these edges are connected, with a pictorial representation of G in Figure 2. For notational simplicity, denote $v_1 \rightarrow v_2$ as the directed edge from v_1 to v_2.

(i) For each $1 \leq u \leq m$, add the edge $v_{start} \rightarrow x_{1,u}$.

(ii) For each $1 \leq u \leq m$, add the edge $y_{k,u} \rightarrow v_{end}$.

(iii) For each $1 \leq t \leq k$, and for each $1 \leq u_1, u_2 \leq m$, add the edge $x_{t,u_1} \rightarrow y_{t,u_2}$ iff there exists a (feasible) block for which the first two columns have index u_1 and the last two columns have index u_2.

(iv) For each $1 \leq t \leq k - 1$, and for each $1 \leq u_1, u_2 \leq m$, add the edge $y_{t,u_2} \rightarrow x_{t+1,u_1}$ iff the concatenation matrix C_{u_2,u_1} has no row with three consecutive home sets, no row with three consecutive away sets, and no row with the same opponent appearing in Columns 2 and 3.

The following theorem (Hoshino and Kawarabayashi 2011c) shows that the k-block NPB scheduling problem can be reformulated in a graph-theoretic context, for any $k \geq 1$.

Theorem 1 *Every feasible solution of the NPB scheduling problem can be described by a path from v_{start} to v_{end} in graph G. Conversely, any path from v_{start} to v_{end} in G corresponds to a feasible intra-league NPB schedule.*

Having constructed our digraph, we now assign a *weight* to each edge using the distance matrix so that the shortest path (i.e., path of minimum total weight) from v_{start} to v_{end} corresponds to the optimal tournament schedule that minimizes the total distance traveled by the n teams.

For any block, we define its *in-distance* to be the total distance traveled by the n teams within that block, i.e., starting from set 1 and ending at set $2(n - 1)$. Note that the in-distance does not include the distance traveled by the teams heading *to* the venue of set 1 or *from* the venue of set $2(n - 1)$. We will use this definition in part (C) below:

(A) For each $1 \leq u \leq m$, the weight of edge $v_{start} \rightarrow x_{1,u}$ is the total distance traveled by the $\frac{n}{2}$ teams making the trip from their home city to the venue of their set 1 opponent.

(B) For each $1 \leq u \leq m$, the weight of edge $y_{k,u} \rightarrow v_{end}$ is the total distance traveled by the $\frac{n}{2}$ teams making the trip from the venue of their opponent in set $2k(n - 1)$ back to their home city.

(C) For each $1 \leq t \leq k$, and for each $1 \leq u_1, u_2 \leq m$, the weight of edge $x_{t,u_1} \rightarrow y_{t,u_2}$ is the *minimum* in-distance of a block, selected among all blocks for which the first two columns have index u_1 and the last two columns have index u_2.

(D) For each $1 \leq t \leq k - 1$, and for each $1 \leq u_1, u_2 \leq m$, the weight of edge $y_{t,u_2} \rightarrow x_{t+1,u_1}$ is the total distance traveled by the teams that travel from their match in set $2t(n - 1)$ to their match in set $2t(n - 1) + 1$, where the last two columns of the tth block have index u_2 and the first two columns of the $(t + 1)^{\text{th}}$ block have index u_1.

To illustrate (D), consider the 20-set schedule formed by concatenating two copies of Table 2. Then the last two columns of the first block (sets 9 and 10) have index q and the first two columns of the next block (sets 11 and 12) have index r. When we concatenate these two blocks, the weight of edge $y_{1,q} \rightarrow x_{2,r}$ is the total distance traveled by the teams from their matches in set 10 to their matches in set 11. This sum equals $D_{2,5} + D_{2,6} + D_{4,3} + D_{3,5} + D_{4,6}$, the distances traveled by teams t_1, t_2, t_4, t_5, and t_6, respectively.

By this construction, we have produced a weighted digraph. In part (C), suppose there exist two blocks B and B' for which the first two columns have index u_1 and the last two columns have index u_2. If the in-distance of B is less than the in-distance of B', then block B' cannot be a block in an optimal solution, since we can just replace B' by B to create a feasible solution with a lower objective value. This trivial observation, based on Bellman's Principle of Optimality, allows us to assign the *minimum* in-distance as the weight of edge $x_{t,u_1} \rightarrow y_{t,u_2}$, for all $1 \leq u_1, u_2 \leq m$. As a result, we have a digraph G on $2mk + 2$ vertices and at most $2m + (2k - 1)m^2$ edges, with a unique weight for each edge. Combined with the previous theorem, we have established the following.

Theorem 2 *Let $P = v_{start} \rightarrow x_{1,p_1} \rightarrow y_{1,q_1} \rightarrow x_{2,p_2} \rightarrow y_{2,q_2} \rightarrow \ldots \rightarrow x_{k,p_k} \rightarrow y_{k,q_k} \rightarrow v_{end}$ be a shortest path in G from v_{start} to v_{end}, i.e., a path that minimizes the total weight. For each $1 \leq t \leq k$, let B_t be the block of minimum in-distance selected among all blocks for which the first two columns have index p_t and the last two columns have index q_t. Then the multi-block schedule $S = B_1, B_2, \ldots, B_k$, created by concatenating the k blocks consecutively, is an optimal solution for the NPB scheduling problem.*

Shortest-Path Algorithm: condition (g)

In the previous section, we described a four-step procedure to produce a weighted digraph G. Parts (A), (B), (D) are easy to implement once we specify the distance matrix of a particular n-team instance (e.g. Table 1 for the NPB Central League). For part (C), we need to ensure that the weight of each edge $x_{t,u_1} \to y_{t,u_2}$ is the *minimum* in-distance of a block for which the first two columns have index u_1 and the last two columns have index u_2. To accomplish this, we need to enumerate *all* possible ten-set blocks that satisfy the seven conditions, and then apply the $n \times n$ distance matrix to determine the correct weight of each edge.

Following the standard three-phase approach (Rasmussen and Trick 2007), we first generate double round-robin home-away pattern (HAP) sets in the form of an n by $2(n-1)$ matrix, then convert these HAP sets into timetables which are assignments of matches to time slots, and finally convert timetables into feasible $2(n-1)$-set schedules (i.e., blocks) by assigning each team in $\{t_1, t_2, \ldots, t_n\}$ a unique row in the matrix.

We need to repeat this procedure for each of the four block positions: sets $1-10$, $11-20$, $21-30$, and $31-40$. As an example, the pattern for the first block is EDEDEDEDED (where E is a week**E**nd and D is a week**D**ay), and the pattern for the second block is EDEEDEDEDE due to the break for inter-league games between sets 13 and 14.

For example, Table 3 provides a valid HAP (with $n = 6$) for the first block, which produces many possible feasible timetables, including the one shown in Table 4. Then the timetable in Table 4 can be converted into the block schedule given in Table 2, via the mapping $\{\#1, \#2, \#3, \#4, \#5, \#6\} \to \{t_1, t_2, t_3, t_4, t_5, t_6\}$.

	1	2	3	4	5	6	7	8	9	10
Week-End/Day	E	D	E	D	E	D	E	D	E	D
Team #1	0	1	1	0	0	1	1	0	1	0
Team #2	0	1	0	1	1	0	1	0	0	1
Team #3	1	0	0	1	1	0	0	1	0	1
Team #4	0	1	0	1	0	0	1	0	1	1
Team #5	1	0	1	0	1	1	0	1	0	0
Team #6	1	0	1	0	0	1	0	1	1	0

Table 3: A feasible HAP for the first block (sets 1 to 10).

	1	2	3	4	5	6	7	8	9	10
Week-End/Day	E	D	E	D	E	D	E	D	E	D
Team #1	#5	**#6**	**#2**	#4	#3	**#4**	**#5**	#6	**#3**	#2
Team #2	#6	**#3**	#1	**#5**	**#4**	#5	**#6**	#3	#4	**#1**
Team #3	**#4**	#2	#5	**#6**	**#1**	#6	#4	**#2**	#1	**#5**
Team #4	#3	**#5**	#6	**#1**	#2	#1	**#3**	#5	**#2**	**#6**
Team #5	**#1**	#4	**#3**	#2	**#6**	**#2**	#1	**#4**	#6	#3
Team #6	**#2**	#1	**#4**	#3	#5	**#3**	#2	**#1**	**#5**	#4

Table 4: Timetable corresponding to the above HAP.

By a direct combinatorial enumeration, we determine all possible HAPs and timetables that satisfy the constraints of the NPB scheduling problem, repeating the analysis for each of the four block positions. We find that there are 1960 feasible timetables for Blocks #1 and #4, 624 timetables for Block #2, and 736 timetables for Block #3.

Each feasible timetable yields 6! different blocks, any of which can appear in an optimal solution to the NPB scheduling problem. For example, to calculate the weights of all edges of the form $x_{2,u_1} \to y_{2,u_2}$, we enumerate all $624 \times 6!$ possible options for Block #2, and find the weight of the block with minimum in-distance.

Recall the *weekday-weekend* requirement given in condition (g) earlier: each of the $624 \times 6!$ options for Block #2 has the property that *every* team has the $(3, 2, 3, 2)$ four-tuple that respectively counts weekend home sets, weekday home sets, weekend road sets, and weekday road sets. And each of the $1960 \times 6!$ options for Block #1 has the property that every team has either a $(3, 2, 2, 3)$ or $(2, 3, 3, 2)$ four-tuple.

However, in order for the final tournament schedule (i.e., the concatenation of four separate blocks) to be a feasible solution to the NPB problem, each team's final four-tuple must be $(10, 10, 10, 10)$. Thus, if some team's Block #1 four-tuple is $(3, 2, 2, 3)$, then that team's Block #4 four-tuple must be $(2, 3, 3, 2)$. To ensure this, we partition the $1960 \times 6!$ options for Block #1 into $\binom{6}{3} = 20$ cases, for each of the ways that three fixed teams among $\{t_1, t_2, \ldots, t_6\}$ can have a $(3, 2, 2, 3)$ four-tuple, while the other three have a $(2, 3, 3, 2)$ four-tuple. We repeat this process for Block #4.

Thus, we can match up each of the 20 cases for Block #1 to exactly *one* of the 20 cases for Block #4, knowing that any feasible path from v_{start} to v_{end} must necessarily satisfy all seven conditions of the NPB scheduling problem. In other words, we need to run Dijkstra's Algorithm twenty times, with each iteration being run on a directed graph whose edge weights are determined from $1960 \times \frac{6!}{20}$ options for Block #1 and $1960 \times \frac{6!}{20}$ options for Block #4.

All code was written and compiled using Maplesoft 13 using a single Toshiba laptop under Windows with a single 2.10 GHz processor and 2.75 GB RAM. Based on the distance matrix in Table 1, Maplesoft ran Dijkstra's Algorithm twenty times to produce the following distance-optimal schedule for the NPB Central League, in just under three hours. The total travel distance is 66, 122 kilometres.

Team	1–5	6–10	11–15	16–20
t_1	t_2 t_3 t_5 t_4 t_6	t_4 t_2 t_3 t_5 t_6	t_5 t_2 t_3 t_6 t_4	t_6 t_4 t_3 t_2 t_5
t_2	t_1 t_4 t_3 t_6 t_5	t_6 t_1 t_4 t_3 t_5	t_4 t_1 t_5 t_3 t_6	t_3 t_6 t_5 t_1 t_4
t_3	t_6 t_1 t_2 t_5 t_4	t_5 t_6 t_1 t_2 t_4	t_6 t_4 t_1 t_2 t_5	t_2 t_5 t_1 t_4 t_6
t_4	t_5 t_2 t_6 t_1 t_3	t_1 t_5 t_2 t_6 t_3	t_2 t_3 t_6 t_5 t_1	t_5 t_1 t_6 t_3 t_2
t_5	t_4 t_6 t_1 t_3 t_2	t_3 t_4 t_6 t_1 t_2	t_1 t_6 t_2 t_4 t_3	t_4 t_3 t_2 t_6 t_1
t_6	t_3 t_5 t_4 t_2 t_1	t_2 t_3 t_5 t_4 t_1	t_3 t_5 t_4 t_1 t_2	t_1 t_2 t_4 t_5 t_3

Team	21–25	26–30	31–35	36–40
t_1	t_4 t_3 t_2 t_6 t_5	t_6 t_5 t_3 t_2 t_4	t_6 t_5 t_3 t_2 t_4	t_6 t_4 t_5 t_3 t_2
t_2	t_6 t_4 t_1 t_5 t_3	t_5 t_3 t_4 t_1 t_6	t_5 t_3 t_4 t_1 t_6	t_5 t_6 t_3 t_4 t_1
t_3	t_5 t_1 t_6 t_4 t_2	t_4 t_2 t_1 t_6 t_5	t_4 t_2 t_1 t_6 t_5	t_4 t_5 t_2 t_1 t_6
t_4	t_1 t_2 t_5 t_3 t_6	t_3 t_6 t_2 t_5 t_1	t_3 t_6 t_2 t_5 t_1	t_3 t_1 t_6 t_2 t_5
t_5	t_3 t_6 t_4 t_2 t_1	t_2 t_1 t_6 t_4 t_3	t_2 t_1 t_6 t_4 t_3	t_2 t_3 t_1 t_6 t_4
t_6	t_2 t_5 t_3 t_1 t_4	t_1 t_4 t_5 t_3 t_2	t_1 t_4 t_5 t_3 t_2	t_1 t_2 t_4 t_5 t_3

Table 5: Optimal solution to the NPB scheduling problem.

The 2013 NPB Central League Schedule

Nippon Professional Baseball is divided into the six-team Central League and the six-team Pacific League. While each league is officially part of the NPB they run as two separate entities, each with its own director and staff. In September 2012, the authors met with the director of the NPB Central League, who doubles as its chief scheduler. (Unfortunately we were unable to meet with the Pacific League officials.)

Within the Central League, the scheduling process works as follows: first, the league asks each of the six teams to submit dates in which their home stadium is not available (e.g. due to concerts, trade shows, and other events) as well as preferred home dates and match-ups against rival teams. All of this information is then considered by the league, from which a list of "hard constraints" is produced. In producing the 2013 Central League intra-league schedule, there were 47 hard constraints, *all* of which needed to be satisfied.

Of the seven league-wide constraints given in the NPB scheduling problem, conditions (a), (b), (c), (e), (g) are hard constraints, while (d) and (f) may be relaxed under exceptional circumstances. For example, there is an annual high school baseball tournament that takes place each summer in the home stadium of the Hanshin Tigers (team t_2). As a result, this team must play on the road in three consecutive sets sometime during Block #3, thus violating *at-most-two*, condition (f). Similarly, condition (d), *each-round*, can be violated if the Central League scheduler cannot find a feasible block satisfying all of the hard constraints.

Note that the elimination of constraint (d) no longer allows for the balanced block structure based on the one-factorization of the complete graph K_6; however, even when this constraint is removed for certain blocks, we can still apply Dijkstras Algorithm, since two blocks X and Y can be joined by simply checking the 6×4 concatenation matrix formed by the last two columns of Block X and the first two columns of Block Y. Therefore, our shortest-path approach is still valid, though the nice symmetric and balanced structure no longer applies.

Define a *trip* to be any pair of consecutive sets not occurring in the same city (i.e., any situation where that team doesn't play at home in sets s and $s + 1$, and therefore has to travel from one venue to another.) Then Table 5 requires a total of 195 trips and 66, 122 kilometres, which is significantly better than the 2012 NPB Central League intra-league schedule which required 206 trips and 86, 364 kilometres. Furthermore, our theoretically-best schedule satisfies conditions (a) through (g), while the 2012 schedule violated conditions (d) and (f) multiple times – naturally, by having extra 3-set road trips, one can reduce the number of total trips.

To illustrate this key point, it is possible to create a Central League intra-league schedule with only 170 trips and 57, 836 kilometres (Hoshino and Kawarabayashi 2011b) satisfying conditions (a) through (e), i.e., all but the *at-most-two* and *weekend-weekday* balancing conditions. These final two conditions significantly increase the total travel distance, but as we can see, the schedule given in Table 5 is much closer distance-wise to 57, 836 km than 86, 364 km.

While the intra-league schedule of Table 5 is theoretically the best possible, it naturally does not satisfy *all* 47 Central League constraints for the 2013 season. However, we started by applying Dijkstra's Algorithm to produce the distance-optimal schedule, to give us a baseline of what could be achieved. When we met with the Central League chief scheduler, he inputted the 47 constraints one by one, thus removing the large majority of the possible blocks. For example, 13 of the 47 team constraints related to sets scheduled in Block #1, and once these hard constraints were added, the $1960 \times 6!$ options for Block #1 reduced to just 32 choices, including the 10-set block provided in Table 2.

Furthermore, due to stadium unavailability, teams t_2 and t_4 both required a three-set road trip in Block #3, forcing us to re-do our analysis by enumerating all possible HAP sets and timetables that satisfy the six scheduling conditions excluding *at-most-two*. Even with this relaxation, there were no blocks that satisfied all the hard constraints while only having two violations of the *at-most-two* condition; all choices for Block #3 required a minimum of one 3-game home stand and four 3-game road trips.

After all 47 constraints were added, it was a simple matter to run our shortest-path algorithm on the set of possible blocks. Naturally, the number of possible 10-game blocks reduced dramatically from our analysis in the previous section, where no such constraints were in place.

As a result, Maplesoft computed the shortest path in just four minutes (instead of three hours). We made some extensions to our code and generated the entire collection of 40-set intra-league schedules satisfying all 47 hard constraints, conditions (a) through (e), as well as the weekend-weekday balancing condition (g).

Upon the Central League's request, we restricted our analysis to the subset of 180 intra-league tournament schedules with the fewest violations of the *at-most-two* condition, all of which had two 3-set home stands and five 3-set road trips. We printed off the top seven tournaments, ordered by travel distance, with the best being a schedule that required 194 trips and 76, 598 kilometres of total team travel.

After our final consultation meeting, the Central League met with the team representatives one last time, and a few additional adjustments were made to the schedule just before the Official Release, to ensure revenue-maximizing matches of rival teams on major weekends. As a result, the final schedule is more inefficient that the one we proposed, with 194 trips, four 3-set home stands, six 3-set road trips, and 80, 006 kilometres of total travel.

Nevertheless, we were able to play a valuable role in helping the Central League produce an intra-league tournament schedule that reduced total travel by over 6, 000 kilometres and required 12 fewer trips, as compared to last year's schedule.

Of course, the exact same graph-theoretic methods will work to optimize the Pacific League whose six teams are spread throughout Japan; all that is required is the 6×6 distance matrix and the set of hard constraints for this league.

We look forward to partnering with the NPB once again, and hope to have the opportunity to help this league produce future regular-season schedules that will result in annual win-wins for the people of Japan: both economically and environmentally.

In conclusion, we remark that this weekday-weekend balancing requirement is important to other sports leagues. A natural question is whether the ideas in this paper can be applied to optimize the scheduling for Major League Baseball, especially as MLB recently approved a major realignment (into 2 leagues of fifteen teams, with inter-league games spread throughout the season). Perhaps the combinatorial approaches described in this paper can be scaled to help MLB devise schedules for both the 15-team American League and the 15-team National League to simultaneously reduce travel while ensuring a fair and equitable distribution of weekend home games for all thirty teams.

References

Easton, K.; Nemhauser, G.; and Trick, M. 2001. The traveling tournament problem: description and benchmarks. *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming* 580–584.

Hesse, S. 2012. Canadian uses math to green Japanese basebal. [Online; accessed 21-January-2013].

Hoshino, R., and Kawarabayashi, K. 2011a. The inter-league extension of the traveling tournament problem and its application to sports scheduling. *Proceedings of the 25th AAAI Conference on Artificial Intelligence* 977–984.

Hoshino, R., and Kawarabayashi, K. 2011b. The multi-round balanced traveling tournament problem. *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)* 106–113.

Hoshino, R., and Kawarabayashi, K. 2011c. A multi-round generalization of the traveling tournament problem and its application to Japanese baseball. *European Journal of Operational Research* 215:481–497.

Hoshino, R., and Kawarabayashi, K. 2013. Graph theory and sports scheduling. *Notices of the American Mathematical Society* to appear.

Kendall, G.; Knust, S.; Ribeiro, C.; and Urrutia, S. 2010. Scheduling in sports: An annotated bibliography. *Computers and Operations Research* 37:1–19.

Rasmussen, P., and Trick, M. 2007. A Benders approach for the constrained minimum break problem. *European Journal of Operational Research* 177:198–213.

Ribeiro, C., and Urrutia, S. 2004. Heuristics for the mirrored traveling tournament problem. *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling* 323–342.