Journal of Engineering and Applied Sciences 13 (21): 9204-9212, 2018
ISSN: 1816-949X

Solving the Traveling Tournament Problem Based on the Simulated Annealing and Tabu Search Algorithm

[1]Jihyun Kim, [2]Jaehyun Han and [3]Sukjae Jeong
[1]School of Business Administration, Kwangwoon University, 526 Nuri Hall, 20 Kwangwoon-ro, Nowon-gu, 01897 Seoul, South Korea
[2]School of Business Administration, Kwangwoon University, 533 Nuri Hall, 20 Kwangwoon-ro, Nowon-gu, 01897 Seoul, South Korea
[3]School of Business Administration, Kwangwoon University, 531 Nuri Hall, 20 Kwangwoon-ro, Nowon-gu, 01897 Seoul, South Korea

Abstract: The professional sports league has been one of the biggest business industries. And both each team manager and league manager have interested in good game schedule for the benefit. Sports League Scheduling Problem (SLSP) study had started in operation research and many researchers have been worked over the last two decades. The traveling tournament problem is recently proposed by Goerigk and Westphal which is a well-known combinatorial optimization problem aimed at developing optimal schedules for sport leagues. The TTP is considered a difficult problem in that its constraints appear to be simple but are difficult to satisfy and the objective of minimizing the total travel distance is difficult to achieve. Thus, it has been considered a challengeable combinatorial optimization problem for both theoretical and practical reasons. In this study, we present a hybrid heuristic algorithm using the Tabu search and simulated annealing procedures for solving the traveling tournament problem. Computational experiments using a hybrid approach on benchmark sets give results comparable to or better than current best known solutions.

Key words: Combinatorial optimization problem, double round-Robin tournament, integer programming, simulated annealing, Tabu search, traveling tournament problem

INTRODUCTION

Professional sports leagues form a huge global industry and popular teams make huge profits. In the US, television networks can pay more than $400 million per year for nationally televised baseball games. For example, Manchester United, a publicly traded British soccer team, commands more than £100 million for overseas broadcasting rights alone. Sport league scheduling has received considerable attention in recent years because these applications involve significant television networks and generate challenging combinatorial optimization problems. As shown in Table 1, feasibility constraints imposed by rules, venues and other regulations applied to the sport make the resulting combinatorial optimization problem a very difficult one.

In manufacturing, having and following an appropriate production schedule is critical. Similarly, optimal game schedules are of critical importance to sports leagues in that they allow teams to maximize profits, strike a balance between matches and venues, maximize/minimize breaks and minimize the travel time. The Traveling Tournament Problem (TTP) is an important research topic which developed for requirement of major league basketball in the United Status. The problem is to minimize distances traveled by league which each sport team takes. In research years, some methods have been presented for solving the TTP. However, such studies have had considerable difficulty developing optimal game schedules because of diverse objectives and complex constraints.

Table 1: Elements of sports scheduling

Match rules	Venues	Other regulations
Round-Robin	Neutral venues	Limits the number of consecutive games
Double round-Robin	Home/away venues	Break-time regulations
Double round-Robin mirrored)	Special venues for events	Prevention of repeat games

Anagnostopoulos *et al.* (2006) presented an advanced simulated annealing algorithm for the TTP that include strategic oscillation and reheats to balance the exploration of the feasible and infeasible regions and to escape local minima at very low temperatures. Lim *et al.* (2006) used a interaction strategy between simulated annealing and hill-climbing method to divide the search space into a timetable space and a team assignment space.

Corresponding Author: Sukjae Jeong, School of Business Administration, Kwangwoon University, 531 Nuri Hall, Kwangwoon-ro, Nowon-gu, 01897 Seoul, South Korea

In their research, the timetable space is explored by a simulated annealing algorithm while the team assignment space is explored by a hill-climb algorithm. Di Gaspero and Schaerf (2007) have proposed a Tabu search for the TTP, focused on the definition of the neighborhood. The neighborhood employed in the search is a composition that has been designed starting from basic neighborhood structures, according to an analytical study of the cardinality, the overlapping and the average quality of the components. Irnich (2010) proposed a fast solution method for the TTP on the decomposition into master and pricing problems. He has showed that this problem can be reformulated as an ordinary shortest-path problem over an expanded network and the proposed algorithm can improve many lower bounds of knowingly hard TTP instances from the literature. Yamaguchi *et al.* (2011) proposed an approximation algorithm for TTP with constraints such that both the number of consecutive away games and that of consecutive home games are at most k. A key of the proposed algorithm is to construct an almost shortest Hamilton cycle passing all the venues and finds a permutation of teams such that the above cyclic order corresponds to the obtained Hamilton cycle. Easton *et al.* (2001) defined the TTP's objective as the minimization of the total travel distance for all teams in the league. In addition, Easton *et al.* (2002) presented optimal solutions for leagues composed of 4, 6 or 8 teams by using a branch and price algorithm based on integer and constraint programming. Benoist *et al.* (2001) examined some hybrid algorithms combining Lagrangian relaxation and constraint programming with round-Robin assignment and travel optimization. Russell and Urban (2006) considered the problem of scheduling sports games through several venues not associated with any of the teams by using constraint programming. Anagnostopoulos *et al.* (2006) achieved good results with a simulated annealing algorithm for the TTP that explores both feasible and infeasible schedules and searches for large neighborhoods with complex moves. Lee *et al.* (2006) suggested a heuristic algorithm based on the Tabu search procedure, introducing better solution features (alternation and intimacy) and proposing a strategic search method.

In this study, we present a heuristic algorithm based on Simulated Annealing (SA) and Tabu search (TA) procedures, for solving the practical as well as large size problems that face many teams.

MATERIALS AND METHODS

Problem description

Definition of traveling tournament problem: The TTP is generally defined as a problem that is presented in the form of round-Robin tournament among teams, so that, every team plays ever other team. Such a tournament has n-1 slots during which n/2 games are played. For each game, one team is denoted the home team and its opponent is the away team. As suggested by the name, the game is held at the venue of the home team. A double round-Robin tournament has 2 (n-1) slots and has every pair of teams played twice, once at home and once away for each team (Easton *et al.*, 2001).

Table 2: Characteristics of the TTP in sports scheduling

Matching rules	Venues	Other regulation
Double-round-Robin	Home/away venues	Consecutive limit no. Repeat game regulation

In the TTP, d_{ij} represents the distance between the homes of teams i and j given by a n×n symmetric matrix d. Each team begins the tournament at its home site to which it must return at the end of the tournament. In addition, when a team plays an away game, the team travels from one away venue to another directly. The cost to each team is the total distance traveled over the tournament.

Constraints of the TTP: The conditions which the TTP is assumed are categorized by elements of sports league scheduling. As shown in Table 2, the match rule which determine the number of games is the double-round robin. And the TTP assume that each team has a own home venues. And it contains special regulations optionally witch are consecutive limit generally, 3 home/away game and 2 team make a game each home venues continually. These characteristics become constraints of TTP. These constraints seem to be simple but it is hard to satisfy, so, it has been known a difficult problem to solve even for very small cases.

There are two basic requirements for solving the TTP. The first requirement is to find the feasible game schedules. The second is the minimization of the total travel distance for all the teams in the league.

The key to the TTP is a conflict between minimizing travel distances and feasibility constraints on the home/away pattern. A solution to the TTP must satisfy the following representative constraints:

Double round-Robin constraints: Each pair of teams, for example, a pair of teams, A and B, play exactly twice-once at A's home site and once at B's home site. Thus, there is the exit total 2(n-1) rounds and n/2 games are played in each round.

Consecutive constraints: For each team, no more than three consecutive home or three consecutive away games is allowed. In other word, more than three consecutive home stands or road trips are forbidden by these constraints.

No-repeat constraints: For any pair of teams, for example a pair of teams, A and B, after A and B play at A's home site, A and B cannot immediately play at B's home site in

Table 3: Example of the above constraints for the TTP

T/R	1	2	3	4	5	6	7	8	9	10
1	3	-3	4	5	**-4**	**-5**	**-6**	**-2**	6	2
2	6	4	-5	-4	3	-6	5	1	-3	-1
3	-1	1	-6	6	-2	**4**	**-4**	5	2	-5
4	5	-2	-1	2	1	**-3**	**3**	-6	-5	6
5	-4	**6**	**2**	-1	-6	1	-2	-3	**4**	**3**
6	-2	-5	3	-3	**5**	**2**	**1**	**4**	-1	-4

Row 1 satisfies no-repeat constraint. Row 3 and 4 satisfies double round-robin constraint. Row 5 and 6 satisfies consecutive constraint

next round. The example of above constraints is shown in Table 3. A schedule is represented by a timetable indicating the competing teams. Each row corresponds to a team and each column, a round. The opponent of team i in round t is given by the absolute value of the element (I, t). If (I, t) is positive, the game takes place at I's team home and at the opponent's home otherwise. Anagnostopoulos *et al.* (2006) designed this representation of schedules. In Table 3, the blocked square of team 1 and 6 is an example of violating consecutive constraints and the blocked square of team 3 and 4 is an example of violating no-repeat constraints. Finally, the blocked square of team 5 means double round-robin constraints.

Mathematical model for the TTP: This study's proposed mathematical model is as follows:

Decision variable:

$$Y_{i,j,k,t}=\begin{cases}1 & \text{if team } i \text{ moves from team } j\text{'s home to team } k\text{'s home in round } t\\ 0 & \text{otherwise}\end{cases}$$

$$\text{Minimize}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{t=1}^{2n-2} d_{jk}\times Y_{i,j,k,t} \tag{1}$$

$$\text{Subject to}\sum_{j=1}^{n}\sum_{t=1}^{2n-2} Y_{i,j,i,t}=n-1 \qquad \forall_i \tag{2}$$

$$\sum_{j=1}^{n}\sum_{j=1}^{2n-2} Y_{i,j,k,t}=1 \qquad \forall_i, k(k\neq i) \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} Y_{i,j,k,t}\leq 2 \qquad \forall_i, j, t(i\neq j) \tag{4}$$

$$\sum_{j=1}^{n} Y_{i,j,k,t}\leq\sum_{j=1}^{n} Y_{k,j,k,t} \qquad \forall_i,\ k,\ t \tag{5}$$

$$\sum_{j=1}^{n} Y_{i,j,k,t}\sum_{j=1}^{n} Y_{i,k,j,t+1} \qquad \forall_i, k, t \tag{6}$$

$$\sum_{j=1}^{n} Y_{i,j,t,1}=1 \qquad \forall_i \tag{7}$$

$$\sum_{j=1}^{n} Y_{i,j,i,2n-1}=1 \qquad \forall_i \tag{8}$$

$$Y_{i,k,i,t}+Y_{k,k,i,t}\leq 1 \qquad i,k,t(i\neq k) \tag{9}$$

$$\sum_{t}^{t+3} Y_{i,i,i,t}\leq 3 \qquad \forall_i, t=1, 2, \ldots, 2n-5 \tag{10}$$

$$\sum_{\substack{j=1\\ \neq i}}^{n}\sum_{\substack{k=1\\ \neq i}}^{n}\sum_{t}^{t+3} Y_{i,j,k,t}\leq 3 \qquad \forall_i, t=1, 2, \ldots, 2n-5 \tag{11}$$

The objective function 1 is the same as formal models. Equation 2 is the constraint for the number of (n-1) games that must be played at the home site and Eq. 3 necessitates that each team must play only once against all the teams at their home sites. Equation 4 represents that no more than two teams can make use of one venue simultaneously and equation 5 means that if an opponent visits, the home team must be at its own home site. Thus, according to Eq. 4 and 5, a venue must accommodate either two teams simultaneously or no team. Equation 6 is a moving sequence constraint indicating that if team i is at team j's home site in round t then team i must start at team j's home in round t+1. Equation 7 and 8 are the constraints indicating that all teams must start at their home site at the beginning of the tournament and return to their home site at the end of the tournament. Equation 9 is the no-repeat constraint ensuring that a match between a pair of teams does not immediately follow a match at the home site of one of the two teams. Finally, Eq. 10 and 11 are the "no consecutive games" constraints preventing more than three consecutive home or away games. This mathematical model does not provide optimal solutions even for a small number of teams. That is the model can find an optimal solution for up to only four teams. In this regard, the present paper proposes a heuristic method that can address this size problem.

Proposed SA-Tabu search algorithm: A simulated annealing algorithm (TTSA) of Anagnostopoulos *et al.* (2006) defines an effective neighbor mechanism and shows good performance in terms of solution quality. However, it is slow because of random searching. In this regard, this study proposes a hybrid algorithm based on the simulated annealing and Tabu search procedures that can perform well in terms of both solution quality and time consumption.

The search process is based on SA which has a random search process. On the other hand, random searching can produce a feasible solution. Then a local search based on the Tabu search procedure is started by

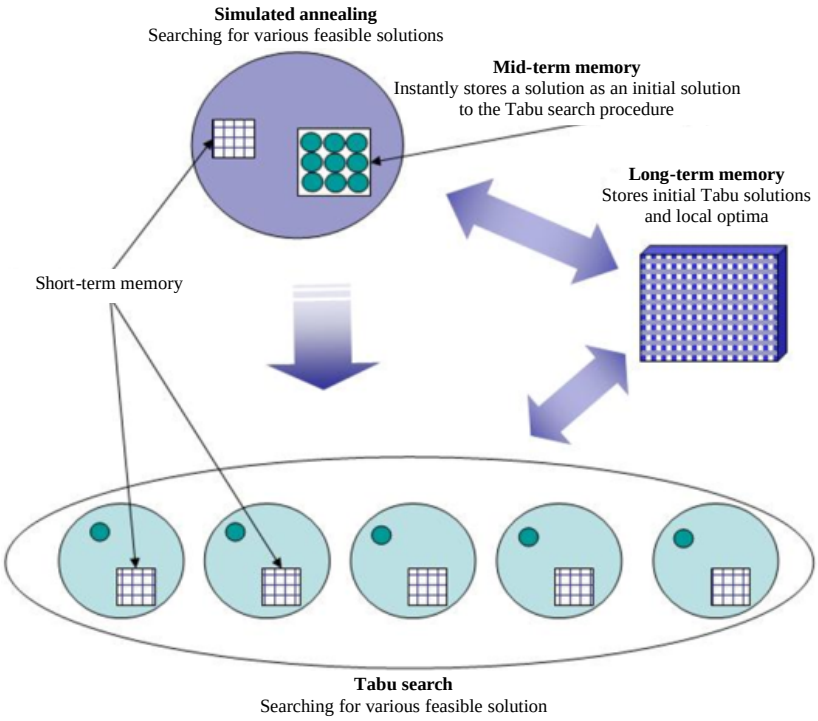

Fig. 1: SA-Tabu search process

using a feasible solution from simulated annealing. It is expected that additional feasible solutions can be found through the Tabu search procedure. That is, SA is designed for shallow and global searching and TS is for further local searching. In other words, SA is for diversification whereas TA is for intensification (Fig. 1).

Over all searching process is based on Simulated annealing. Simulated annealing has random search process. While random searching, a good feasible solution may be founded. Then local search based on Tabu search process is started with a feasible solution founded by simulated annealing. It is expected that more good feasible solution is founded by Tabu search. That is simulated annealing derives Tabu search processes. And each process performs their searching in parallel. Simulated annealing is designed to have shallow and global searching and Tabu search is designed to have further local searching. In other words, Simulated annealing performs diversification and Tabu search performs intensification.

Each process has some memory, including short-term memory. Short-term memory prevents each process from re-visiting a visited solution. On the other hand, long-term memory provides a solution that is used as the initial solution for TA procedure. Long-term memory is shared by SA and Tabu and thus, a solution visited or initiated by one process is never visited by another which allows each process to escape the local optimum.

Neighborhood generation: The neighborhood generation mechanism is based on the TTSA. The neighborhood of a schedule S is a set of (possibly infeasible) schedules that can be obtained by applying one of 5 types of moves. The first 3 types have a simple intuitive meaning whereas the remaining two generalize them. In this regard, the present paper suggests a process for resolving infeasible solutions.

Swap homes (S, T_i, T_j): This move swaps the home/away roles of team T_i and T_j. In other words, if team T_i plays at home against team T_j in round k and away at team T_j's home in round l then swap homes (S, T_i, T_j) is the same as the Schedule (S), except that team T_i now plays away against team T_j in round l and at home against team T_j in round l. Table 4 illustrates swap homes (S, T_2, T_4).

Swap rounds (S, R_k, R_l): This move simply swaps round R_k for round R_l. Table 5 illustrates swap rounds (S, R_3, R_5).

Swap teams (S, T_i, T_j): This move swaps the schedules of team T_i and T_j (except, of course when they play against each other). Table 6 illustrates swap teams (S, T_2, T_5).

Table 4: Swap homes

T/R	1	2	3	4	5	6	7	8	9	10
1	6	-2	4	3	-5	-4	-3	5	2	-6
2	5	1	-3	-6	**-4**	3	6	**4**	-1	-5
3	-4	5	2	-1	6	-2	1	-6	-5	4
4	3	6	-1	-5	**2**	1	5	**-2**	-6	-3
5	-2	-3	6	4	1	-6	-4	-1	3	2
6	-1	-4	-5	2	-3	5	-2	3	4	1
1	6	-2	4	3	-5	-4	-3	5	2	-6
2	5	1	-3	-6	**4**	3	6	**-4**	-1	-5
3	-4	5	2	-1	6	-2	1	-6	-5	4
4	3	6	-1	-5	**-2**	1	5	**2**	-6	-3
5	-2	-3	6	4	1	-6	-4	-1	3	2
6	-1	-4	-5	2	-3	5	-2	3	4	1

Home/away role is swapped for teams 2 and 4

Table 5: Swap rounds

T/R	1	2	3	4	5	6	7	8	9	10
1	6	-2	**4**	3	**-5**	-4	-3	5	2	-6
2	5	1	**-3**	-6	**4**	3	6	-4	-1	-5
3	-4	5	**2**	-1	**6**	-2	1	-6	-5	4
4	3	6	**-1**	-5	**-2**	1	5	2	-6	-3
5	-2	-3	**6**	4	**1**	-6	-4	-1	3	2
6	-1	-4	**-5**	2	**-3**	5	-2	3	4	1
1	6	-2	**-5**	3	**4**	-4	-3	5	2	-6
2	5	1	**4**	-6	**-3**	3	6	-4	-1	-5
3	-4	5	**6**	-1	**2**	-2	1	-6	-5	4
4	3	6	**-2**	-5	**-1**	1	5	2	-6	-3
5	-2	-3	**1**	4	**6**	-6	-4	-1	3	2
6	-1	-4	**-3**	2	**-5**	5	-2	3	4	1

Rounds 3 and 5 for all teams are swapped

Table 6: Swap teams

T/R	1	2	3	4	5	6	7	8	9	10
1	6	-2	4	3	-5	-4	-3	5	2	-6
2	5	**1**	**-3**	**-6**	**4**	**3**	**6**	**-4**	**-1**	-5
3	-4	5	2	-1	6	-2	1	-6	-5	4
4	3	6	-1	-5	-2	1	5	2	-6	-3
5	-2	**-3**	**6**	**4**	**1**	**-6**	**-4**	**-1**	**3**	2
6	-1	-4	-5	2	-3	5	-2	3	4	1
1	6	**-5**	4	3	**-2**	-4	-3	**2**	**5**	-6
2	5	**-3**	**6**	**4**	**1**	**-6**	**-4**	**-1**	**3**	-5
3	-4	**2**	**5**	-1	6	**-5**	1	-6	**-2**	4
4	3	6	-1	**-2**	**-5**	1	**2**	**5**	-6	-3
5	-2	**1**	**-3**	**-6**	**4**	**3**	**6**	**-4**	**-1**	2
6	-1	-4	**-2**	**5**	-3	**2**	**-5**	3	4	1

Schedules of teams 2 and 5 are swapped

Note that, in addition to the changes in lines 2 and 5, the corresponding lines of the opponents of T_i and T_j must also be changed. As a consequence, there are four values per round (column) that are changed (except when T_i and T_j meet).

However, these three moves are not sufficient for exploring the entire search space and as a consequence they lead to suboptimal solutions for a large number of teams. These issues can be addressed by considering two moves that are more general. Although, these moves do not follow the interpretation of the first three they are similar in terms of their structure and they can dramatically enlarge the neighborhood, providing a more connected search space. More precisely, these moves reflect partial swaps that is they swap a subset of the schedule in round

Table 7: Partial swap rounds (S, T_2, R_2, R_9)

T/R	1	2	3	4	5	6	7	8	9	10
1	6	-2	2	3	-5	-4	-3	5	4	-6
2	5	**1**	-1	-5	4	3	6	-4	**-6**	-3
3	-4	5	4	-1	6	-2	1	-6	-5	2
4	3	6	-3	-6	-2	1	5	2	-1	-5
5	-2	-3	6	2	1	-6	-4	-1	3	4
6	-1	-4	-5	4	-3	5	-2	3	2	1
1	6	**4**	2	3	-5	-4	-3	5	**-2**	-6
2	5	**-6**	-1	-5	4	3	6	-4	**1**	-3
3	-4	5	4	-1	6	-2	1	-6	-5	2
4	3	**-1**	-3	-6	-2	1	5	2	**6**	-5
5	-2	-3	6	2	1	-6	-4	-1	3	4
6	-1	**2**	-5	4	-3	5	-2	3	**-4**	1

Partially swaps schedule for rounds 2 and 9 of team 2 and updates the rest

Table 8: Patial swap teams (S, T2, T4, R9)

T/R	1	2	3	4	5	6	7	8	9	10
1	6	-2	4	3	-5	-4	-3	5	2	-6
2	5	1	-3	-6	4	3	6	-4	**-1**	-5
3	-4	5	2	-1	6	-2	1	-6	-5	4
4	3	6	-1	-5	-2	1	5	2	**-6**	-3
5	-2	-3	6	4	1	-6	-4	-1	3	2
6	-1	-4	-5	2	-3	5	-2	3	4	1
1	6	-2	**2**	3	-5	-4	-3	5	**4**	-6
2	5	1	**-1**	**-5**	4	3	6	-4	**-6**	**-3**
3	-4	5	**4**	-1	6	-2	1	-6	-5	**2**
4	3	6	**-3**	**-6**	-2	1	5	2	**-1**	**-5**
5	-2	-3	6	**2**	1	-6	-4	-1	3	**4**
6	-1	-4	-5	**4**	-3	5	-2	3	**2**	1

Partially swap round 9 schedule of teams 2 and 9 and update the rest

i and round j for a subset of the schedule for team T_i and team T_j. These moves are beneficial in that they are not as global as the "macro" moves swap teams and swap rounds. As a consequence, they may facilitate a better trade-off between feasibility and optimality by increasing the feasibility of solution in one part of the schedule while not reducing it in another. Further, they are more "global" than the "micro" move swap homes.

Partial swap rounds (S, T_i, R_k, R_l): This move considers team T_i and swaps its games in round k and round l. Then the rest of the schedule for these rounds is updated in a deterministic manner to produce a double round-Robin tournament. Table 7 illustrates partial swap rounds (S, T_2, R_2, R_9).

Partial swap teams (S, T_i, T_j, R_k): This move considers round k and swaps the games of team T_i and team T_j. Then the rest of the schedule for these teams (and their opponents) is updated to produce a double round-robin tournament. Table 8 illustrates partial swap teams (S, T_2, T_4, R_9).

Design of simulated annealing: The design of the Simulated Annealing (SA) procedure is based on the TTSA. When this procedure finds a feasible solution for the list of best solutions, it yields a Tabu list with this

solution. SA starts with a random solution satisfying the double round-Robin constraint and it has a high initial temperature which makes any solutions acceptable.

SA has shot-term, mid-term and long term memory and stores the current solution in its short team memory which prevents it from revisiting solutions for cycle free searching. However, it has small short-term memory (approximately 10). If the simulated annealing procedure invokes the Tabu search procedure, it records the solution to its mid term memory. If a solution already exists in its mid term memory then the solution is deleted from the mid term memory and transferred to its long term memory. In this way, solutions in its long term memory are never revisited. It has unlimited mid term and long term memory. The characteristics of the simulated annealing procedure are summarized as follows (Algorithm 1 shows sample code):

Cost function: As in Anagnostopoulos *et al*. (2006), the simulated annealing procedure makes use of an adaptive modification of weights for any violation of consecutive or no-repeat constraints. More specifically, the weight of each component is allowed to vary according to the so called "shifting penalty" mechanism in which the weight is divided (multiplied) by the factor α if for k consecutive iterations, all the constraints of that component are satisfied. This mechanism constantly changes the shape of the cost function in an adaptive manner, thereby allowing the simulated annealing procedure to pass through infeasible states and visit states that have a structure different from those of previously visited states.

Stop criteria: The simulated annealing procedure is stopped when long term memory is not updated for a specified number of iterations when the temperature is half the temperature at which the last best feasible or infeasible solution is found or when the phase exceeds the number of iterations.

Initial temperature: This sets to a very high temperature to accept any solutions.

Cooling schedule: This refers to a geometric cooling schedule such as $T = \beta \cdot T$. For temperature to cool rapidly, β is set to 0.9.

Algorithm 1; Sample code for Simulated annealing:

```
1. find random schedule S
2. bestFeasible←∞; nbf←∞
3. bestInfeasible←∞; nbi←∞
4. reheat←0; counter←0; phase←0; lastUpdate←0
7. while phase≤max Phase or T≤best Temperature do
8.   counter←0
9.   while counter≤counterLimit and lastUpdate≤updateLimit do
10.    select a random move m from neighborhood (S)
11.    let S' be the schedule obtained from S with m
12.    if C (S')<C (S) or
13.      nbv (S') = = 0 and C (S')< bestFeasible or
14.      nbv(S')>0 and C(S')<bestInfeasible
15       and S' is not in Short-term memory
16.    then
17.      accept←true
18.    else
19.      accept←true with probability exp (-ΔC/T)
20.           false otherwise
21.    if accept then
22.      S←S'
23.      Record S in Short-term memory
24.    if nbv (S') == 0 then
25.      nbf←min (C (S), bestFeasible)
26.     if C (S) is acceptable bestFesaibleList then
27.       if S is not in Long-term memory then
28.          if S is in Mid-term memory then
29.             Delete S in Mid-term memory
30.             Record S in Long-term memory
31.          else
32.             Record S in Mid-term memory
33.             Make tabu process with S
34.             lastUpdate←0
35.        else
36.          lastUpdate++
37.       else
38.         lastUpdate++;
39.     else
40.       nbi←min(C(S), bestInfeasible);
41.     if nbf < bestFeasible or nbi < bestInfeasible then
42.       reheat←∞0; counter←∞0; phase←∞0
43.       bestTemperature←T
44.       bestFeasible←nbf
45.       bestInfeasible←∞nbi
46.       if nbv (S) == 0 then
47.         ω←ω/α
48.       else
49.         ω←ω*α
50.     else
51.       counter++
52.   phase++
53.   T←T*β
```

Design of the Tabu search procedure: The design of the Tabu search procedure is based on the heuristic method proposed by Lee *et al*. (2006) and focuses on an intensive local search near feasible solutions. The terms "alternation" and "intimacy", introduced by Lee *et al*. (2006), present good solution features and allow for a strategic search solution space. The Tabu search can be summarized as follows:

Alternation: Alternation refers to the number of home away or away home conversions for one team. The elite schedule has few conversions.

Intimacy: Intimacy refers to the number of home-home patterns and away-away patterns for two adjacent rounds and is used for evaluating each pair of columns (rounds).

Table 9: Alternation of team i

T/R	1	2	3	4	5	6	7	8	9	10
.										
.										
i	-2	-4	-1	6	5	4	-6	1	2	-5
.										
.										

Alteration of values

Table 10: Intimacy of round t and round t+1

T/R	t	t+1
1	-6	4
2	4	-5
3	-5	-6
4	-2	-1
5	3	2
6	1	3

Alteration of values

In a specific team's schedule, if the team's alternation is large then it must travel frequently. In addition, between two specific adjacent rounds, low intimacy needs to be improved through neighborhood searches. As shown in Table 9, team I's alternation is 4 {(-1, 6), (4, -6), (-6, 1), (2, -5)} and as shown in Table 10, its intimacy between round t and round t+1 is 4 {(-5, -6), (-2, -1), (3, 2), (1, 3)}. In the TTP, generating all the neighbors of the current solution requires too much time. Thus, alternation and intimacy are used to search for good neighbors in an effective manner. Each team's alternation is evaluated and then three teams with the highest alternation values are selected. In addition, for each adjacent pair of rounds each round's intimacy is evaluated and then, until the final three or four rounds, a minimum number of round pairs are deleted among the candidates. The design of the Tabu search procedure is summarized as follows:

Neighborhood sampling: After evaluating the alternation and intimacy values for the current solution, three max-alternation teams and three or four min-intimacy rounds are selected. Then the neighbors are generated by applying swap homes and swap teams using four max-alternation teams and swap rounds using four min-intimacy rounds. Finally, partial swap rounds and partial swap teams find the neighbors by using both four max-alternation teams and four min-intimacy rounds.

Movement control: After the neighborhood sampling, the best solution that is not already in short term, mid term or long term memory is chosen. If the chosen solution is not an improvement then the procedure generates 100 neighborhoods randomly and searches for a better solution not in the Tabu list. If successful, the current solution can be moved to improved solution. If not, it can be the local optimum. Finally, the procedure stores this solution in long term memory and moves to an unimproved solution.

Tabu list: There are two types of Tabu lists: short term and long term list. The short term list maintains a record of solution movements and the long term list does a record of the initial solutions used in the Tabu process and local optima.

Stop criteria: The stop criterion is based on the max number of iterations, since, the last improvement.

Other features: If a generated neighborhood is not feasible then it is not considered. The sample code for Tabu search is shown in algorithm 2.

Algorithm 2: Sample code for the Tabu search procedure:

```
01. find random schedule S
02. bestFeasible←∞; lastUpdate←0
03. while lastUpdate≤updateLimi do
04.     record S to short-term memory
05.     generate neighbor samples set N from S and sorting N
06.     for each S' in N
07.         if S' is feasible and S' is not in Tabu lists then
08.             NS←S'; break
09.     end for
10.     if C (NS)≥C (S) then
11.         generate neighbor samples set N' from S and sorting N'
12.         for each S' in N'
13.             if S' is feasible and S' is not in tabu lists then
14.                 NS'←S'; break
15.         end for
16.     if C(NS')<C(S) then
17.         S←NS'
18.     else
19.         record S to Long-term memory
20.         S←NS
21.     if C (S)<bestFeasible then
22.         lastUpdate←0
23.         bestFeasible←S
24.     else
25.         lastUpdate++
```

RESULTS AND DISCUSSION

The effectiveness of the proposed algorithm was analyzed by using two different instances. The first set was composed of small size problems based on the actual distance between Major League Baseball (MLB) teams in

Table 11: Results for 10 runs

			Proposed algorithm				Computational time (sec)			
Instance	Best known	LB	Min	Max	Mean	SD	Min	Max	Mean	SD
NL6	23916	22969	23916	23916	23916	0	75	780	183.4	63.7
NL8	39721	38760	39721	39721	39721	0	325	3403	1320.3	611.1
NL10	59436	56506	59436	61608	60186.2	582.3	3107	1102	5815.5	2443.4
NL12	110729	107483	110729	115192	113874.4	1667.8	7420	30240	23323	9085.2
NL14	188728	182797	188728	207343	196638.1	5070.9	17682	60520	39330.9	13256
NL16	261687	248852	276520	285767	279211.1	4444.9	47343	100340	65203.7	12542
NFL16	231483	223800	238581	241973	240172.2	3421.4	101428	113599	103412	9023.4
NFL18	282258	272834	299192	311546	300243.3	5321.6	212586	276362	23234.4	17231.7
NFL20	332041	316721	342947	361878	357214.3	4123.5	265732	478318	30124.4	20143.3
NFL22	402534	378813	418086	439626	419215.9	6342.1	293760	411264	34768.9	24171.2
NFL24	463657	431226	480528	500017	496123.4	7002.7	311040	435456	36861.5	29134.8
NFL26	536792	495982	573596	590497	588211.1	9012.4	336960	438048	38891.4	30123.7
NFL28	609788	560697	650923	682132	672132.3	9103.2	339661	452332	39932.2	31232.2
NFL30	739697	688875	847011	860234	850179.3	10234.1	346464	485050	40135.2	40312.7
NFL32	914620	836031	1020966	1038706	102921.6	9045.4	363880	527626	41234.7	47737.9

*Best solution is presented in the cited website (http://mat.gsia.cmu.edu/TTP/)

Table 12: Comparison of the computational time (sec)

	Simulated annealing (Anagnostopoulos *et al*., 2006)		Tabu search		Proposed algorithm	
Instance	Min	Mean	Min	Mean	Min	Mean
NL6	NA	NA	308	379.1	75	183.4
NL8	596.6	1639.3	1411	2204.5	325	2020.3
NL10	8084.2	40268.6	3427	6135	3107	5815.5
NL12	28526	68505.3	7329	26751.2	7420	23323
NL14	418358.2	233578.4	15674	41600.1	17682	39330.9
NL16	344633.4	192086.6	48294	71088.3	47343	65203.7

the National League. The other set was composed of large size problems based on the National Football League (NFL). For each instance the proposed algorithm was run 10 times and the best solution and the total running time were recorded. Table 11 shows the results for the 10 runs.

In terms of solution quality, our solutions were the same as those for NL6-NL10. When comparing with our solutions and values of instance more than 10 teams in MLB, the differences were insignificant.

In terms of the computation time, our solution was evaluated by results of Anagnostopoulos *et al.* (2006) and Lee *et al.* (2006), respectively. Table 12 provides a comparison of the computation time among the three approaches. As shown in Table 11, the average computation time for the proposed algorithm was less than that indicated in Anagnostopoulos *et al.* (2006) and Lee *et al.* (2006).

CONCLUSION

The sports league scheduling problem has attracted considerable attention from both researchers and practitioners in recent years because it involves substantial revenues for television networks and generates challenging combinatorial optimization problems. The present paper considers the Traveling Tournament Problem (TTP) (Easton *et al.*, 2001). The TTP schedules a double round-robin tournament to minimize the total distance traveled by teams and involves various feasibility and optimality issues and thus, it is a very difficult problem to solve.

SUGGESTIONS

In this regard, this study proposed an algorithm based on meta heuristic approaches to obtain good solutions in an efficient manner. The results indicate that the proposed algorithm shows good performance in terms of computational efficiency in certain cases. This suggests that the proposed algorithm and its underlying mechanism are effective and practical.

ACKNOWLEDGEMENTS

The research reported in this study was conducted during the sabbatical year of Kwangwoon University in 2015. This research was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2017S1A5B8060156).

NOTATION

- k, i, j = Teams (team = 1, 2, ... , n)
- t = Round (round = 0, 1, 2, ... , 2n-1)
- d = Matrix of the distance between team i and team j

REFERENCES

Anagnostopoulos, A., L. Michel, P.V. Hentenryck and Y. Vergados, 2006. A simulated annealing approach to the traveling tournament problem. J. Scheduling, 9: 177-193.

Benoist, T., F. Laburthe and B. Rottembourg, 2001. Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems. Proc. CPAIOR., 1: 15-26.

Di Gaspero, L. and A. Schaerf, 2007. A composite-neighborhood TABU search approach to the traveling tournament problem. J. Heuristics, 13: 189-207.

Easton, K., G. Nemhauser and M. Trick, 2001. The Traveling Tournament Problem Description and Benchmarks. In: Principles and Practice of Constraint Programming, Walsh, T. (Ed.). Springer, Berlin, Germany, ISBN:978-3-540-42863-3, pp: 580-584.

Easton, K., G. Nemhauser and M. Trick, 2002. Solving the travelling tournament problem: A combined integer programming and constraint programming approach. Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT'02), August 21-23, 2002, Springer, Gent, Belgium, ISBN:978-3-540-40699-0, pp: 100-109.

Irnich, S., 2010. A new branch-and-price algorithm for the traveling tournament problem. Eur. J. Oper. Res., 204: 218-228.

Lee, J.H., Y.H. Lee and Y.H. Lee, 2006. Mathematical modeling and Tabu search heuristic for the traveling tournament problem. Proceedings of the International Conference on Computational Science and its Applications, May 8-11, 2006, Springer, Glasgow, Scotland, UK., pp: 875-884.

Lim, A., B. Rodrigues and X.Zhang, 2006. A simulated annealing and hill-climbing algorithm for the traveling tournament problem. Eur. J. Oper. Res., 174: 1459-1478.

Russell, R.A. and T.L. Urban, 2006. A constraint programming approach to the multiple-venue, sport-scheduling problem. Comput. Oper. Res., 33: 1895-1906.

Yamaguchi, D., S. Imahori, R. Miyashiro and T. Matsui, 2011. An improved approximation algorithm for the traveling tournament problem. Algorithmica, 61: 1077-1091.