2007−MPS−65 (2)
2007/6/25

巡回トーナメント問題に対するタブーサーチ・アルゴリズム

住川 裕岳, 宮代 隆平, 中森 眞理雄
東京農工大学大学院工学府

概要： 本研究では，スポーツスケジューリング問題の一種である巡回トーナメント問題を扱う．巡回トーナメント問題とは，ホーム&アウェイ形式の二重総当りリーグ戦を行うスポーツにおいて，各チームの移動距離の総和を最小化した試合日程を構築する問題である．この問題では，巡回セールスマン問題の難しさに加え，あるチームの対戦順序が他チームの対戦順序に影響を与えており，問題の難易度を増している．これまでの研究により，巡回トーナメント問題に対してはシミュレーテッド・アニーリングが有効であることが示されていたが，本研究ではタブーサーチを用いて最適化を行った．計算機実験の結果，既存のアルゴリズムによる結果に匹敵する質の良い解が得られた．

A Tabu Search Algorithm for the Traveling Tournament Problem

Hirotake Sumikawa, Ryuhei Miyashiro and Mario Nakamori
Graduate School of Technology
Tokyo University of Agriculture and Technology
Koganei, Tokyo 184-8588, JAPAN

Abstract. The traveling tournament problem is a well known benchmark problem in sports scheduling. This problem has both an optimization aspect like the traveling salesman problem and a feasibility aspect as in many scheduling/timetabling problems. Since the traveling tournament problem was established, a number of researchers have tackled the problem with various optimization techniques. Recent researches indicated that simulated annealing algorithms are effective for the traveling tournament problem, and few results by tabu search are reported so far. In this manuscript, we propose a tabu search algorithm for the traveling tournament problem. Our computational experiments show that the proposed algorithm generates good solutions, which are competitive with solutions by simulated annealing algorithms.

1 Introduction

The traveling tournament problem is a famous benchmark problem in sports scheduling, a recent topic in combinatorial optimization. This problem has not only an optimization aspect like the traveling salesman problem but also a feasibility aspect as in many scheduling/timetabling problems. These aspects make the traveling tournament problem much difficult. For instance, there is an unsolved instance of the traveling tournament problem of 10 cities, whereas 1,000 cities instances of the traveling salesman problem can be easily solved nowadays.

Since the traveling tournament problem was established in 2001 [3], a number of researchers have tackled the problem with various optimization techniques, such as integer programming, constraint programming and several metaheuristics. In particular, simulated annealing algorithms are known to be effective for the traveling tournament problem [1, 7, 9]. On the contrary, there have been proposed few algorithms using tabu search [5].

In this paper, we propose a tabu search algorithm for the traveling tournament problem. Our computational experiments show that the proposed algorithm produces solutions competitive with the previous solutions generated by simulated annealing based algorithms.

2 The Traveling Tournament Problem

The traveling tournament problem was proposed by Easton et al. [3] in 2001. In addition, several benchmark instances of the traveling tournament problem have been released on the website [8].

The traveling tournament problem is to find a *double round-robin tournament* whose total traveling distance is minimized. A double round-robin tournament is a schedule satisfying the following constraints:

(C1) Each team plays every other team twice, once at its home and once at away.

(C2) Every team plays one game in every round.

	1	2	3	4	5	6
1	@3	@4	@2	4	2	3
2	4	3	1	@3	@1	@4
3	1	@2	@4	2	4	@1
4	@2	1	3	@1	@3	2

Figure 1: A double round-robin tournament

(C3) Each game is held at the one of the home of the playing two teams.

In this paper, we assume that each team has its home cities, and the distance matrix among the cities is given. Due to the constraints (C1) and (C2), the number of rounds is $2(n-1)$, where n is the number of teams in a tournament.

Figure 1 is a schedule of a double round-robin tournament of 4 teams. Each row is indexed by teams, and each column is indexed by rounds. Each row shows opponents of the team corresponding to the row. Each entry with @ means that the game is held at the home of the opponent, i.e., away game for the team corresponding to the row. Each team stays/returns its home cities before/after the tournament, respectively. Thus, for example, team 2 in Figure 1 stays at its home until round 3, goes to the home of team 3, to the home of team 1, to the home of team 4, and then returns to its home.

In addition to the constraints (C1)–(C3), the following constraints are also considered for practical reasons.

(C4) No team plays more than three consecutive games at away.

(C5) No team plays more than three consecutive games at home.

(C6) No pair of teams plays against each other in two consecutive rounds.

The *total traveling distance* of a double round-robin tournament is the sum total of the traveling distance of all teams. The *traveling tournament problem* is defined as follows: for a given number of teams n and a distance matrix among their homes, to find a schedule of a double round-robin tournament that satisfies the constraints (C1)–(C6) and minimizes its total traveling distance.

The benchmark instances of the traveling tournament problem are described in the webpage [8]. There are some class of instances, NLn, NFLn, CIRCn and CONn. The instance class NLn consists of instances NL4, NL6, NL8, NL10, NL12, NL14 and NL16, where each number means the number of teams. In NLn instances, the distance matrices come from the real distances among the home cities of the Major League Baseball teams.

It is difficult to obtain optimal solutions of NLn instances. For the NL8 instance, an optimal solution was obtained by using 20 PCs with 4 days of computational time [4]. The NL10 instance is not solved yet, and the best objective value and lower bound obtained are 59436 [1] and 57500 [8], respectively. In this paper, we concern NL8 and NL10 instances with a tabu search algorithm, which is described in the next section.

3 Algorithm

In this section, we propose an algorithm for the traveling tournament problem. Our algorithm employs tabu search. In Section 3.1, we explain how to create an initial solution for tabu search. In Section 3.2, the neighborhoods for tabu search are described. In Section 3.3, the procedures of our tabu search algorithm are shown.

3.1 Initial Solution

To generate an initial solution for tabu search, we make use of the circle method [2], which is a well known algorithm to create a round-robin tournament.

In this paper, we define a *single* round-robin tournament as a tournament satisfying the constraints (C2), (C3) and (C1′) defined as follows, instead of (C1):

(C1′) Each team plays every other team once.

A single round-robin tournament created by the circle method is as follows:

in round r $(r = 1, 2, \ldots, n-1)$,

- team n plays team r
  - at the home of team n if r is even,
  - at the home of team r otherwise,
- team i plays team j satisfying $i + j \equiv 2r \bmod (n-1)$
  - at the home of team j if $r \equiv i+2, i+4, \ldots, i+n-2 \bmod (n-1)$,
  - at the home of team i otherwise.

By substituting teams, the circle method can produce $n!$ different single round-robin tournaments.

Using the circle method, we create a double round-robin tournament as follows. First, we create a single round-robin tournament by randomly substituting teams of a tournament obtained by the circle method. Then, we create another single round-robin tournament without assignment of home/away. Such a tournament can easily obtained by removing assignment of home/away from a single round-robin tournament. Then, we concatenate these two schedules to form a double round-robin tournament. In particular, we should assign home/away to games in the latter slots so as to satisfy the constraint (C1).

We called the above procedure the *modified circle method*. Note that schedules created with this method satisfy the constraints (C1)–(C3), but do not necessarily satisfy the constraints (C4)–(C6).

3.2 Neighborhoods

The neighborhoods used in our algorithm are as follows. The neighborhoods (N1)–(N5) are proposed in [1], and the neighborhood (N6) is newly proposed in this paper.

(N1) Swap Homes:
If teams i and j play at the i's home and j's home in the rounds r_1 and r_2, respectively, swap these two games. (As a result, teams i and j play at the i's home and j's home in the rounds r_2 and r_1, respectively.)

(N2) Swap Rounds:
Swap rounds r_1 and r_2 of a schedule. (In other words, swap two columns of a schedule.)

(N3) Swap Teams:
Replace all opponents of teams i but j for all opponents of teams j but i. (In other words, swap two rows of a schedule.)

(N4) Partial Swap Rounds:
Swap two opponents of team i in rounds r_1 and r_2, and accordingly fix corresponding games to satisfy the constraint (C1)–(C3). (See [1] for detail.)

(N5) Partial Swap Teams:
Swap the opponents of teams i and j in round r, and accordingly fix corresponding games to satisfy the constraint (C1)–(C3). (See [1] for detail.)

(N6) Random Swap Rounds:
All rounds are permuted at random. This neighborhood is used when the search procedure cannot find a better solution after a long iteration (see the Section 3.3 in detail).

3.3 Tabu Search

Our algorithm is based on tabu search. Tabu search is a metaheuristic algorithm for combinatorial optimization problems. (See [6] for tabu search.)

First, we described the basic operation of the algorithm. Our algorithm searches the neighborhoods (N1)–(N5) of a current schedule, moves to the schedule of the best objective value, and then continues to search the neighborhoods. After 20,000,000 moves, the algorithm terminates. Note that an initial solution satisfies the constraints (C1)–(C6), and moves using the neighborhoods (N1)–(N5) do not produce schedules violating the constraints (C1)–(C3). However, the constraints (C4)–(C6) can be violated by this algorithm. If a schedule violates the constraints (C4)–(C6), we add a penalty to the objective value.

Next, we explain tabu lists in the proposed algorithm. To avoid cycling, we keep the objective value and the number of violation of the constraints (C4)–(C6) in a short term memory, whose length is $n \times n$. In addition, we use a long term memory to avoid using a same neighborhood many times.

Finally, our algorithm force to move to another solution from a current solution by using the neighborhood (N6), if there is no improvement in the objective value after the fixed number of moves in a search process. This jump is to escape from a local optimal solution.

4 Computational Experiment

To show the effectiveness of the proposed algorithm, we performed computational experiments as follows. The computational environment is as follows: CPU Celeron 2.53 GHz, RAM 512 MB. The algorithm is implemented in C++ programming language.

We used the benchmark instances NL8 and NL10. As mentioned, the optimal value of NL8 is 39721, and NL10 is not solved yet. For each instance, we performed ten runs of the proposed tabu search procedure. Tables 1 and 2 show the results of experiments and CPU time, respectively. To evaluate our results, Tables 1 and 2 also show the results of [5] and [9]. In [5], an algorithm using tabu search was executed on a 1.5 GHz, AMD Athlon PC running Linux. The algorithm in [9] using simulated annealing which are known to be effective for the traveling tournament problem were on an AMD Athlon 64 at

Table 1: The objective values of obtained schedules

n	old best	min	mean	max	lower bound	gap (%)	min [5]	min [9]
8	39721	39721	39721.0	39721	39721	0.00	–	39721
10	59436	59583	59702.2	59963	57500	0.24	59583	59436

Table 2: CPU time

n	min	mean	s. d.	min [5]	min [9]
8	77.58	529.76	461.56	–	1169.00
10	582.81	15915.39	18168.39	689	2079.06

2 GHz.

For NL8 instance, we obtained an optimal solution in all of ten runs of the algorithm. For NL10 instance, the objective value of the best obtained solution is 59583, whose gap to the best known solution is only 0.25%.

5 Conclusion

We proposed a tabu search algorithm for the traveling tournament problem. Our computational experiments obtained an optimal solution for 8 teams instance (NL8), and a solution whose gap to the known best is 0.25% for 10 teams instance (NL10). These experiments show that the proposed tabu search algorithm generates good solutions, competitive with solutions generated by simulated annealing algorithms.

Acknowledgement

This work was partially supported by the Grant-in-Aid for Young Scientists (B) 18710127 from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, Y. Vergados: A simulated annealing approach to the traveling tournament problem. Journal of Scheduling, 9(2), pp. 177–193, 2004.

[2] D. de Werra: Geography, games and graphs, Discrete Applied Mathematics, 2, pp. 327–337, 1980.

[3] K. Easton, G. Nemhauser, M. Trick: The traveling tournament problem: description and benchmarks, Lecture Notes in Computer Science, 2239 (2001), Springer, Berlin, pp. 580–585.

[4] K. Easton, G. Nemhauser, M. Trick: Solving the traveling tournament problem: a combined integer programming and constraint programming approach. Lecture Notes in Computer Science, 2740 (2003), Springer, Berlin, pp. 100–109.

[5] L. D. Gaspero, A. Schaerf: A tabu search approach to the traveling tournament problem. Proceedings of the Sixth Metaheuristics International Conference (MIC 2005), pp. 278–283, 2005.

[6] F. Glover, M. Laguna: Tabu Search, Kluwer, Boston, 1997.

[7] A. Lim, B. Rodrigues, X. Zhang: A simulated annealing and hill-climbing algorithm for the traveling tournament problem. European Journal of Operational Research, 174(3), pp. 1459–1478, 2005.

[8] M. Trick: Challenge Traveling Tournament Instances. http://mat.gsia.cmu.edu/TOURN/

[9] P. Van Hentenryck, Y. Vergados: Traveling tournament scheduling: a systematic evaluation of simulated annealling. Lecture Notes in Computer Science, 3990 (2006), Springer, Berlin, pp. 228–243.