A Tabu Search Approach to the Traveling Tournament Problem

Luca Di Gaspero Andrea Schaerf

SOMMARIO/*ABSTRACT*

Il problema del Traveling Tournament (TTP) è un problema combinatorio che unisce le caratteristiche del problema del commesso viaggiatore e della generazione di calendari sportivi. In questo lavoro presentiamo un approccio alla soluzione del TTP basato sulla ricerca tabù che fa uso della combinazione di due relazioni di vicinato. L'algoritmo è stato analizzato sperimentalmente su alcuni benchmarks disponibili in rete e viene presentata una comparazione dei suoi risultati con approcci alternativi presentati nella letteratura sull'argomento.

The Traveling Tournament Problem (TTP) is a combinatorial problem that combines features both from the traveling salesman problem and from tournament scheduling. We propose a tabu search approach to the solution of TTP that makes use of a combination of two neighborhood relations. The algorithm has been experimentally analyzed on several sets of publicly available benchmarks and we show a comparison with previous approaches presented in the literature.

Keywords: Metaeuristiche, analisi sperimentale di algoritmi, esperienze applicative

1 Introduction

The Traveling Tournament Problem (TTP) has been proposed by Easton *et al.* [5] and consists in scheduling a double round-robin tournament, while minimizing the total traveling distance of the teams and satisfying a set of constraints. TTP is considered an interesting problem from both the theoretical and the practical side. On the one hand, it has a natural combination of features from the traveling salesman problem [12] and the tournament scheduling problem [8, 13], so that it exhibit strong feasibility issues (due to the tournament structure) together with a complex optimization part (traveling distance). On the other hand, sport scheduling problems are important for their economical value. National sport leagues mainly in Western Europe and USA represent a big business that depends also on good schedules (see [5] for a discussion).

Due to the mentioned combinations of features, TTP has shown to be difficult to solve by means of exact methods even for very small instances. For example, its solution by either Integer Programming or Constraint Programming so far has been possible only for instance up to only six teams [6]. A set of challenging instances for which many researchers have provided solutions, but in many cases the optimal one is still to be found, is available at [14]. TTP is thus a challenging and promising problem for heuristic and meta-heuristic search approaches. In fact, using local search techniques Anagnostoupulus *et al.* [1] have been able to find the best known solutions for almost all instances of the most popular testbed. In addition, Ribeiro and Urrutia [11], also using a local search technique, have obtained very good results for a restricted version of TTP (explained below).

In this ongoing work, borrowing ideas from [1] and [11], we try to tackle TTP using an approach based on tabu search and on the combination of neighbourhoods. In this paper, we report our preliminary results.

2 The Traveling Tournament Problem

The input of TTP is an (even) positive integer n and an $n \times n$ distance matrix. The output is a double round-robin tournament on n teams; that is, a tournament with $2n - 2$ rounds such that each team plays one game in every round, and each team plays each other team exactly twice: once home and once away.

A solution to the TTP problem must satisfy the following two (hard) constraints:

H1 (No repeat): A match t_1-t_2 can never be followed in the next round by the match t_2-t_1

H2 (At most): A team t cannot play more than 3 consecutive games at home or away.

The objective of the problem is to minimize the total

	Team	ATL	NYM	PHI	MON	FLA	PIT
0	ATL	0	745	665	929	605	521
1	NYM	745	0	80	337	1090	315
2	PHI	665	80	0	380	1020	257
3	MON	929	337	380	0	1380	408
4	FLA	605	1090	1020	1380	0	1010
5	PIT	521	315	257	408	1010	0

Figure 1: The NL6 instance

		Round									Total
No.	Team	0	1	2	3	4	5	6	7	8	9	Distance
0	ATL	-4	-1	+3	+2	-5	-3	-2	+5	+1	+4	4414
1	NYM	+2	+0	+5	-3	-2	+4	+3	-4	-0	-5	3328
2	PHI	-1	+5	-4	-0	+1	-5	+0	+3	+4	-3	3724
3	MON	+5	-4	-0	+1	+4	+0	-1	-2	-5	+2	3996
4	FLA	+0	+3	+2	-5	-3	-1	+5	+1	-2	-0	5135
5	PIT	-3	-2	-1	+4	+0	+2	-4	-0	+3	+1	3319

Figure 2: An optimal solution to the NL6 instance

traveling distance, assuming that a team starts the tournament at home and returns home at the end of the tournament (but not between two games). As an example, consider the instance called NL6, which has 6 teams and the distance matrix reported in Figure 1. For this instance the optimal cost is known (23916) and the solution shown in Figure 2 is an optimal one.

The solution matrix must be interpreted as follows. The sign represents the location of the match: '+' home, '-' away; the number is the opponent number. For example, the games of the first round are: 4-0, 1-2, and 3-5 (FLA-ATL NYM-PHI MON-PIT). The last column is the total traveling of each team. For example team 0 (ATL) goes at FLA, then at NYM, then home, stays home, goes to PIT, and so on. The total traveling for this team is: 605 + 1090 + 745 + 0 + 521 + 408 + 380 + 665 + 0 + 0 + 0 = 4414.

An additional constraint considered by [11] (and imposed in many sport leagues) is the so-called *mirror constraint*. The tournament must be composed by two consecutive single round-robin tournament parts which are identical except that home and away positions are inverted.

3 Tabu search for TTP

In order to apply a local search to our problem, we have to define several features. We first illustrate the search space, the cost function and the procedure for computing the initial state. Then, we define the neighborhood structure, and finally we describe the guiding meta-heuristic.

3.1 Search Space, Cost Function, and Initial State

As search space, we select the set of all correct tournament schedules, including those violating the constraints H1 and H2. That is, in any state it is not possible that a team plays zero or two games, but for example it is possible that two teams play twice back to back (violating constraint H1).

Round 1	1-6	2-5	4-3
Round 2	6-2	3-1	5-4
Round 3	3-6	4-2	1-5
Round 4	6-4	5-3	2-1
Round 5	5-6	1-4	3-2

Figure 3: The canonical pattern for $2n = 6$

The cost function is the weighted sum of the violations of H1 and H2 plus the travel distance. The weights are adjusted so that hard constraints are always more valuable than distance. However, as we will see below, the weight can vary dynamically during search, so that it is possible to explore also the infeasible region of the space.

The initial state is generated in two steps: First, we select a *tournament pattern*, i.e. a tournament in which "anonymous" numbers from 0 to $n-1$ are used as teams. Second, associate all actual teams with distinct numbers in the pattern.

The problem of determining the tournament pattern is related to the problem of finding an *1-factorization* of a complete (undirected) graph [10]. That is, given the complete graph K_{2n} we must partition it in a set of $2n-1$ sets of n arcs (called *1-factors*), such that in each set the arcs are pairwise non adjacent. Each arc represents a match and each 1-factor a round. Therefore, giving an order to the 1-factors and assigning home or away teams for each match, a 1-factorization can be turned into a tournament pattern.

The pattern we use is the so-called *canonical pattern* (see, e.g., [2]), which can be easily computed in linear time. The first half of the canonical pattern for $n = 6$ is shown in Figure 3. In order to obtain different initial states for each run, round order and home/away order are randomized.

In the current implementation, the team assignment is also done in a totally random way.

3.2 Neighborhoods

We make use of two basic neighborhood relations, which are taken from [1] and [11], with some slight modifications (we have experimented also with other neighborhood relations, but with no significative result so far).
N_1: Given a triple composed by two teams t_1 and t_2 and a round r, the opponents of t_1 and t_2 in r are exchanged.
N_2: Given a triple composed by two rounds r_1 and r_2 and a team t the opponents of t in rounds r_1 and r_2 are exchanged.

For both N_1 and N_2, in order to obtain a correct tournament structure the given exchange must be followed by a chain of exchanges that involve also other teams and rounds, that we call *recover chain*. There are different

ways to define the recover chain, and we do not discuss them here for the sake of brevity.

Due to the recover chain, it is possible that in a given state different triples lead to the same new state. This is an important issue for tabu search (not for the simulated annealing algorithm in [1]), not only for efficiency reasons but also for the effectiveness of the tabu mechanism. For this reason, we identify the condition that makes two moves equivalent, and we include a test that, during neighborhood generation, excludes all moves that would lead to the same state of a move already evaluated.

We verified that also moves from N_1 and N_2 could lead to the same state. For example, given four teams t_1, t_2, t_3, and t_4, such that the matches t_1-t_2 and t_3-t_4 are in one round and the matches t_1-t_3 and t_2-t_4 in another round. The swap of the two pairs of matches can be obtained both by a N_1 move (that involves two rounds) and by a N_2 move (that involves four teams). Therefore, we identified the conditions that make two moves equivalent, and we consider only one of them.

3.3 Solution Meta-heuristic

The search procedure is driven by a tabu search meta-heuristic [7], which considers, as the neighborhood, the set union $N_1 \cup N_2$ of the two neighborhood relations N_1 and N_2. At each step of the tabu search we evaluate all the neighbors of the current state which are obtained by applying either a move in N_1 or a move in N_2, and we select the best non-tabu one. However, in order to reduce the computational cost of the neighborhood exploration, we consider only "small" moves, i.e. those whose size of the recover chain is smaller than a given value (in the experiments, the maximum is set to 5 for N_1 and to 7 for N_2).

The tabu search meta-heuristic makes use of an adaptive modification of the weights for the violations of the constraints H1 and H2. Namely, the weight of each component is let to vary according to the so-called *shifting penalty* mechanism: if for a number of consecutive iterations all constraints of that component are satisfied (resp. not satisfied), then the weight is divided (multiplied) by a factor k (with $k = 1.02$ in our experiments). This mechanism changes continuously the shape of the cost function in an adaptive way, thus causing tabu search to visit states that have a different structure than the previously visited ones.

The stop criterion is based on the number of iterations since the last improvement (parameter *max idle iterations*). However, when the number of idle iterations reaches its maximum, the search is not stopped but it restarts from the best state found that far. This procedure is finally stopped when it could not find any improvement for a given number of rounds (parameter *max idle rounds*).

The parameters for our experiments are set as follows: 10,000 max idle iterations, 10 max idle rounds, and a dynamic tabu list whose length varies from $\frac{2}{3}n$ to n (where n is the size of the instance).

It is worth mentioning than, given the complexity of computing the size and the structure of the neighborhood [4, 9], it is not possible to assess whether the search space is connected under our neighborhoods. To shed some light on this issue, exploiting the fact that for eight teams the number of non-isomorphic tournament patterns is only six [4], we made the following experiment: we ran our algorithms from a set of initial states built from different patterns. The outcome was that from all six we reached the optimal solution, supporting the hypothesis that the search space is indeed connected under $N_1 \cup N_2$

4 Experimental Analysis

We tested our algorithm on three different sets of instances available at the website [14]. Namely, we experimented with the most popular testbed, the so-called NLx instances, the constant distance instances, named CONx, and the *circular* instances CIRCx (the x stands for the dimension of the instance). On each instance, we perform 20 runs of the solution meta-heuristic, recording the best state found during the search and the total running time of the procedure. The algorithm has been coded in C++, exploiting the EASYLOCAL++ framework [3], and the experiments have been performed on a 1.5GHz AMD Athlon PC running Linux.

In the following tables we present the performances of our meta-heuristic on the aforementioned instances. In order to give a comprehensive picture of the behavior of the algorithm and for the purpose of a more meaningful comparison with other solution techniques, in both tables we give an account of the distribution of the results by means of some descriptive statistics. In details, we report the extreme of the distributions (i.e. the minimum and the maximum), the median value and the 1st and 3rd quartile value (which account for 25% and 75% of the distribution, respectively).

Since instances with 4, 6, and 8 teams are solved optimally, and we easily obtain the very same optimal results, we decided to focus on instances with at least 10 teams.

In Tables 1–3 we show the distribution of the costs found and the running times. Furthermore, we present also the best cost found by all the runs performed during our experimentation (not necessarily with the final parameter configuration used in the reported experiments) and, in the column labeled "Best known", we report the cost of the best solution known at the time of writing.

The experiments on the NLx instances show that the results of our algorithm in terms of solution cost ranges from 0% up to about 6% of the best known solution. However, if we look at the results in term of the times needed to reach those solutions (refer to [1] for the detailed results of the current best solution method) our algorithm is, on the average, more than an order of magnitude faster than [1].

Moving to the CONx set of instances, we recall that for these instances the distances between the cities is unitary.

Table 1: Distribution of costs and running times of the meta-heuristic on the NLx instances

Instance	Best	Cost values					Best	Running times				
	known	min	1st q.le	median	3rd q.le	max	found	min	1st q.le	median	3rd q.le	max
NL10	**59,583**	**59,583**	59,975	60,293	60,475	61,133	**59,583**	689s	871s	1,445s	4,675s	8,391s
NL12	111,248	112,334	113,822	114,652	115,192	116,945	111,483	646s	1,029s	1,330s	1,748s	3,632s
NL14	189,766	194,780	197,484	199,714	201,002	207,343	190,174	1,881s	2,456s	2,937s	3,564s	4,713s
NL16	267,194	276,520	278,260	280,172	282,502	285,767	270,794	3,069s	4,863s	5,885s	7,551s	13,311s

Table 2: Distribution of costs and running times of the meta-heuristic on the CONx instances (best known results refer to the case with the mirror constraint)

Instance	Best	Cost values					Best	Running times				
	known	min	1st q.le	median	3rd q.le	max	found	min	1st q.le	median	3rd q.le	max
CON10	130	124	124	124	124	124	**124**	844s	859s	866s	874s	922s
CON12	192	181	182	182	182	183	**181**	1,579s	1,683s	1,793s	1,938s	2,182s
CON14	256	252	252	252	253	253	**252**	2,742s	2,943s	3,539s	5,049s	5,967s
CON16	342	329	330	330	330	331	**329**	4,613s	5,257s	5,809s	6,633s	9,239s
CON18	434	419	420	420	420	422	**419**	6,379s	8,756s	10,060s	11,333s	13,292s
CON20	526	522	523	523	524	524	**522**	17,170s	17,489s	17,808s	18,126s	18,445s

In this case, therefore, the problem becomes to minimize the number of trips. For this set of instances, the only available results are the ones of Urrutia and Ribeiro [11] who consider also the mirror constraint.

On this testbed we improve the best known result on all but two instances. The solution meta-heuristic has proven to be quite robust, leading to a very compact distribution of solution costs. Furthermore, the running time scales well with the instance dimension.

Finally, regarding the CIRCx instances, we improve the best knows solution of CIRC10, whereas the results on the other instances are quite significantly worse than the best known ones.

REFERENCES

[1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. Submitted for publication, available from `http://www.cs.brown.edu/~aris/pubs/ttp.pdf`, 2004.

[2] D. de Werra. Scheduling in sports. In P. Hansen, editor, *Studies on Graphs and Discrete Programming*, pages 381–395. North Holland, 1981.

[3] Luca Di Gaspero and Andrea Schaerf. EASYLOCAL++: An object-oriented framework for flexible design of local search algorithms. *Software—Practice and Experience*, 33(8):733–765, 2003.

[4] Jeffrey H. Dinitz, David K. Garnick, and Brendan D. McKay. There are 526,915,620 nonisomorphic one-factorizations of k_{12}. *Journal of Combinatorial Design*, 2:273–285, 1994.

[5] K. Easton, G. Nemhauser, and M. Trick. The traveling tournament problem description and benchmarks. In *Seventh International Conference on the Principles and Practice of Constraint Programming (CP 99)*, volume 2239 of *LNCS*, pages 580–589. Springer-Verlag, 2001.

[6] K. Easton, G. Nemhauser, and M. Trick. Solving the traveling tournament problem: A combined integer programming and constraint programming approach. In *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers*, volume 2740 of *LNCS*, pages 100–109. Springer-Verlag, 2003.

[7] Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[8] Martin Henz. Scheduling a major college basketball conference – revisited. *Operations Research*, 49(1):163–168, 2001.

[9] Charles C. Lindner, Eric Mendelsohn, and Alexander Rosa. On the number of 1-factorizations of the complete graph. *Journal of Combinatorial Theory*, Series B 20:265–282, 1976.

[10] Eric Mendelsohn and Alexander Rosa. One-factorizations of the complete graph – a survey. *Journal of Graph Theory*, 9:43–65, 1985.

[11] Celso C. Ribeiro and Sebastián Urrutia. Heuristics for the mirrored traveling tournament problem. In *Proc. of the 5th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2004)*, pages 323–342, 2004.

Table 3: Distribution of costs and running times of the meta-heuristic on the CIRCx instances

Instance	Best	Cost values					Best	Running times				
	known	min	1st q.le	median	3rd q.le	max	found	min	1st q.le	median	3rd q.le	max
CIRC10	246	244	258	258	260.5	266	**244**	581s	730s	895s	1135s	1499s
CIRC12	408	428	440	442	446.5	452	426	1,129s	1,457s	1,986s	2,350s	3,803s
CIRC14	654	682	690	699	704.5	712	682	1,994s	2,380s	2,746s	4,108s	5,545s
CIRC16	928	1,004	1,021	1,032	1,046.5	1,060	1,004	2,894s	4,308s	5,571	7,205s	9,971s
CIRC18	1,308	1,412	1,452	1,466	1,471	1,484	1,412	5,596s	7,254s	9,508s	11,106s	16,580s
CIRC20	1,842	1,962	1,970	1,980	2,000	2,034	1,962	10,556s	17,201s	22,615s	29,348s	38,129s

[12] M. Ringer, O. Reinelt, and O. Rinaldi. The traveling salesman problem. In M. Ball, T. Magnanti, C. Monma, and G.L. Nemhauser, editors, *Handbooks in Operations Research and Management Science*, chapter 7, pages 225–330. North-Holland, 1995.

[13] Andrea Schaerf. Combining local search and look-ahead for scheduling and constraint satisfaction problems. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)*, pages 1254–1259, Nagoya, Japan, 1997. Morgan-Kaufmann.

[14] Michael Trick. Challenge traveling tournament instances, web page. URL: `http://mat.gsia.cmu.edu/TOURN/`. Viewed: March 30, 2005, Updated: March 11, 2005.

5 Contatti

Luca Di Gaspero e Andrea Schaerf
Dipart. di Ingegneria Elettrica, Gestionale e Meccanica
Università degli Studi di Udine – via delle Scienze 206
I-33100, Udine
email:{`l.digaspero|schaerf`}`@uniud.it`

6 Biografia

Luca Di Gaspero si è laureato in Scienze dell'Informazione nel 1998 ed ha conseguito il Dottorato di Ricerca in Informatica nel 2003 presso l'Università di Udine, dove è attualmente Ricercatore. La sua ricerca riguarda l'applicazione della ricerca locale a problemi di scheduling.

Andrea Schaerf si è laureato con lode in Ingegneria Elettronica nel 1990 ed ha conseguito il Dottorato di Ricerca in Informatica nel 1994 all'Universita di Roma "La Sapienza". Dal 1998 è Professore Associato presso l'Università di Udine. Si interessa principalmente di algoritmi, linguaggi di specifica e strumenti software per problemi di scheduling e timetabling.