

An Improved Neighbourhood for the Traveling Tournament Problem

Glenn Langford*

July 3, 2010

Abstract

The Traveling Tournament Problem (TTP) is a challenging combinatorial optimization problem that has attracted the interest of researchers around the world. This paper proposes an improved search neighbourhood for the TTP that has been tested in a simulated annealing context. The neighbourhood encompasses both feasible and infeasible schedules, and can be generated efficiently. For the largest TTP challenge problems with up to 40 teams, solutions found using this neighbourhood are the best currently known, and for smaller problems with 10 teams, three solutions found were subsequently proven optimal.

1 Introduction

The Traveling Tournament Problem (TTP) proposed in [2] requires scheduling a double round-robin tournament while satisfying a set of constraints. The best schedules also minimize the total travel distance of the tournament participants.

Similar to the Traveling Salesman Problem (TSP), the TTP requires searching for a shortest tour through a number of destinations. However, unlike the TSP the TTP requires the creation of interlocking travel schedules across multiple participants, with tight constraints on feasible solutions.

Effective neighbourhood designs have been developed for the TTP using tabu search and simulated annealing in [3] and [1]. However, it is desirable to have a neighbourhood design that affects fewer games at each transition, especially as the number of teams increases. In a markov chain context,

*e-mail: glenn.langford@gmail.com

we might expect such a neighbourhood to allow a sequence to reach quasi-equilibrium faster. In addition, a larger neighbourhood might also help to ensure that a given search procedure is not trapped in a local minimum.

The rest of this paper is organized as follows: a brief description is given of some known neighbourhood structures encompassing feasible and infeasible schedules. This background discussion is a slightly different presentation of the techniques used in [1]. A proposal for an improved neighbourhood is outlined, and its properties examined. Finally, a summary of experimental results using a simulated annealing approach is presented.

2 Background of the Traveling Tournament Problem

In a TTP schedule, each of n participants or teams must play every other opponent twice; once at the team's home location, and once at the opponent's location. Thus, there are $2n - 2$ games in each team's schedule. TTP challenge problems specify a distance matrix D defining the travel distance between all teams. Traveling between the locations of two teams t_i and t_j incurs a travel cost of $D_{ij} == D_{ji}$. By definition, every team must return home at the end of the schedule; if t_i is away at the location of some team t_x in the last game of the schedule, D_{xi} is added to the total travel distance for t_i .

TTP schedules commonly have two significant "soft" constraints, beyond the double round-robin requirement. A schedule satisfying these constraints is said to be *feasible*, while a schedule which contains one or more violations to these constraints is *infeasible*. The *At Most* constraint requires that no more than three consecutive home or away games are allowed for any team. The *No Repeat* constraint requires that no team play the same opponent in consecutive rounds. (The mirror constraint is not considered in this paper, which is discussed in [4]).

Figure 1 shows an example (shortest) schedule for the Galaxy 10 problem, discovered using the techniques described later.

2.1 Basic Neighbourhood

Following the definition and syntax outlined in [1], a basic TTP neighbourhood is reviewed on which later work is based.

A *move* is a transformation which can be applied to one schedule to reach one or more other schedules. Schedules may be feasible or infeasible,

SOL	ERI	ARA	GEM	PSC	CEP	AND	CNC	PEG	HOR	
---	---	---	---	---	---	---	---	---	---	
@CEP	ARA	@ERI	@CNC	@PEG	SOL	HOR	GEM	PSC	@AND	
@GEM	AND	@PSC	SOL	ARA	@CNC	@ERI	CEP	HOR	@PEG	
@CNC	@PEG	AND	CEP	HOR	@GEM	@ARA	SOL	ERI	@PSC	
CEP	@ARA	ERI	CNC	PEG	@SOL	@HOR	@GEM	@PSC	AND	
ERI	@SOL	GEM	@ARA	@CEP	PSC	PEG	@HOR	@AND	CNC	
ARA	CNC	@SOL	@HOR	@AND	PEG	PSC	@ERI	@CEP	GEM	
@PSC	GEM	@CNC	@ERI	SOL	HOR	@PEG	ARA	AND	@CEP	
@PEG	CEP	@GEM	ARA	AND	@ERI	@PSC	HOR	SOL	@CNC	
@AND	@PSC	CEP	HOR	ERI	@ARA	SOL	PEG	@CNC	@GEM	
CNC	@AND	PSC	PEG	@ARA	@HOR	ERI	@SOL	@GEM	CEP	
PSC	@CEP	CNC	@AND	@SOL	ERI	GEM	@ARA	@HOR	PEG	
AND	PSC	@HOR	@PEG	@ERI	CNC	@SOL	@CEP	GEM	ARA	
@ARA	HOR	SOL	@PSC	GEM	@PEG	@CNC	AND	CEP	@ERI	
HOR	@CNC	PEG	AND	CEP	@PSC	@GEM	ERI	@ARA	@SOL	
PEG	@GEM	HOR	ERI	@CNC	@AND	CEP	PSC	@SOL	@ARA	
@HOR	PEG	@CEP	PSC	@GEM	ARA	CNC	@AND	@ERI	SOL	
@ERI	SOL	@AND	@CEP	@HOR	GEM	ARA	@PEG	CNC	PSC	
GEM	@HOR	@PEG	@SOL	CNC	AND	@CEP	@PSC	ARA	ERI	
404	416	477	463	423	435	452	500	462	503	4535

Figure 1: A tournament schedule for the Galaxy 10 problem comprising $2*10 - 2 = 18$ rounds which satisfy the At Most and No Repeat constraints. The “@” symbol designates an away game, and if t_i plays away to t_j in a round, then by definition t_j is at home to t_i in that same round.

but are always complete double round-robin tournaments at the completion of a move. Multiple types of moves may be defined. The union of schedules reachable in a single move, across all types of moves, defines the *neighbourhood*.

Given a TTP with n teams, (n even) with a set of teams $1 \leq t_i \leq n$ and rounds $0 \leq r_i < 2n - 2$, a neighbourhood can be constructed from the following set of moves:

1. *swapHomes*(t_i, t_j) reverses the home/away status whenever t_i plays t_j . This must occur exactly twice in the schedule for t_i and t_j in the same rounds. All other teams remain unchanged.
2. *swapRounds*(r_k, r_l) swaps all games in round r_k with r_l across all teams.
3. *swapTeams*(t_i, t_j) swaps the schedule of teams t_i and t_j , except when the two teams play each other. The schedules of all the other teams are also updated to reflect the swap. A variant of *swapTeams* combines *swapTeams*(t_i, t_j) with *swapHomes*(t_i, t_j) which produces a new schedule which is violation neutral with respect to the previous schedule (it neither increases nor decreases the number of constraint violations in the schedule).
4. *partialSwapRounds*(t_i, r_k, r_l) starts with a swap of the games in rounds r_k and r_l for team t_i . Suppose t_i 's opponents in these two games are t_x and t_y ; we then continue to swap the same rounds for t_x and t_y , making note of their opponents in the swapped rounds. The process continues until all opponents in any swapped rounds have had the same swap performed; this ensures that a double round-robin tournament is produced that is consistent.
5. *partialSwapTeams*(t_i, t_j, r_k) starts with a swap of the game between t_i and t_j in round r_k . Suppose that t_i swapped away a home game against t_x in that round, and received a home game with t_y in return. From the perspective of t_i , the schedule now has a duplicate home game with t_y : one in round r_k , and another in some other round, r_l . The swap of games between t_i and t_j then continues in round r_l to trade away the duplicated game, and the swapping procedure continues. Swapping stops only when a home game with t_x is received by t_i in a swap. This procedure ensures that a valid double round-robin tournament is preserved, with (hopefully) fewer changes than *swapTeams*(t_i, t_j).

Round	Team Schedules		Swap sequence
	A	B	
0	B	@A	
1	C	@H	1
2	D	@C	
3	E	@F	9
4	F	@G	5
5	G	@E	3
6	H	@D	7
7	@B	A	
8	@C	D	
9	@D	E	8
10	@E	F	4
11	@F	C	10
12	@G	H	6
13	@H	G	2

Table 1: An example schedule fragment (highly infeasible!) for two teams, A and B, part of an 8 team schedule. The schedules for the other six teams (C through H) are elided. The games shown are prior to executing $\text{partialSwapTeams}(A, B, 1)$. The first game swap would take place in round 1, where team A would receive '@H'. The next game to be swapped is thus round 13, where A has a duplicate '@H' game. The home game against team C would be returned to team A after 10 swaps (this game was lost in the first swap), and the partialSwapTeams swap sequence is then complete.

To illustrate the machinery in *partialSwapTeams*, Table 1 shows a progression of swaps in a sample schedule for the move *partialSwapTeams*($A, B, 1$). While round 1 is the starting point shown here, note that the resulting schedule would be the same if any round in that move’s swap sequence was selected as the starting round. This is true in general for *partialSwapTeams*, and as a result the *partialSwapTeams* effective neighbourhood size is substantially smaller than the number of *partialSwapTeams* moves for a given schedule.

3 An Improved Neighbourhood

This section outlines a new strategy for extending *partialSwapTeams*, creating a larger accessible neighbourhood with improved acceptance rates.

The strategy is based on preconditioning a schedule to reduce the number of games swapped by the *partialSwapTeams* procedure. For example, Table 2 illustrates the scope of change similar to Table 1, but with a perturbation first applied to the schedule in the form of a *swapHomes* move. This perturbation supposes we are prescient enough to precondition schedules with some chosen *swapHomes* move prior to executing *partialSwapTeams*.

Simulating the same *partialSwapTeams*($A, B, 1$) move as before on the altered schedule, only 5 rounds are involved in exchanges between A and B rather than 10. Of course, this gain was not achieved without cost, since the *swapHomes* move also disturbed the schedule. This can be partially reversed if we simply repeat the same *swapHomes* move as an additional step.

3.1 Lookahead Partial Swap Teams

Generalizing from this example, consider a new compound move called *lookAheadPartialSwapTeams*, or L-PST, sketched in Algorithm 1. This move is constructed by preconditioning a schedule with *swapHomes* before executing the usual *partialSwapTeams* procedure.

Data: (t_i, t_j, r)
 select look ahead opponent t_l ;
 $swapHomes(t_j, t_l)$;
 $partialSwapTeams(t_i, t_j, r)$;
 $swapHomes(t_j, t_l)$;

Algorithm 1: Sketch of basic Lookahead Partial Swap Teams

The value of L-PST depends on the selection of a *look ahead opponent*

Round	Team Schedules		Swap sequence
	A	B	
0	B	@A	
1	C	@H	1
2	D	@C	
3	E	@F	4
4	F	@G	
5	G	E	3
6	H	@D	
7	@B	A	
8	@C	D	
9	@D	@E	*
10	@E	F	
11	@F	C	5
12	@G	H	
13	@H	G	2

Table 2: A schedule fragment for two teams, A and B, part of an 8 team schedule. The schedule is derived from Table 1 with the home/away status reversed for team E in the schedule for team B. In other words, $swapHomes(B, E)$ has been executed as a preconditioning step. If we then execute $partialSwapTeams(A, B, 1)$ as before, the number of swapped rounds is reduced to 5 from 10. Round 9 is shown in the swap sequence with a '*' since it has been changed; however, if $swapHomes(B, E)$ is repeated after $partialSwapTeams(A, B, 1)$ then round 9 is returned to its original game assignments.

t_l that reduces the number of game swaps and overcomes the added cost of *swapHomes*.

The search for a look ahead opponent starts by simulating the generic *partialSwapTeams*(t_i, t_j, r) procedure, collecting all of the opponents of t_j into a list, called a *swaplist*. For example, looking at the swap sequence shown in Table 1, the swaplist constructed from the opponents of team B is:

$$\{ @H, G, @E, F, @G, H, @D, E, @F, C \} \quad (1)$$

To identify candidates for the look ahead opponent, a swaplist is scanned for duplicate teams with a sufficiently large gap between them; any team that occurs twice in the swaplist is a candidate look ahead opponent. The gap size represents the benefit of the *swapHomes* preconditioning step. For example, team H is a good candidate since there is a gap of 5 between @H and H. The same gap length exists for teams E and F.

$$\{\overbrace{@H, G, @E, F, @G, H}^5, @D, E, @F, C\} \quad (2)$$

Note that choosing either H, E, or F as the look ahead opponent generates three different resulting schedules, increasing the size of the neighbourhood that can be reached in one move.¹ These three new schedules are all achieved with fewer total game swaps than the original *partialSwapTeams* procedure. In each case, the number of expected game swaps is 5, which is predicted by the length of the swaplist less the look ahead opponent gap size. In practice, selection of a look ahead opponent is based on finding the largest gap in the swaplist; ties can be resolved by coin tossing.

Of course, it is possible that no duplicate opponents will be found in a given L-PST swaplist, and therefore the preconditioning strategy is not available. In this case, the usual *partialSwapTeams* procedure is followed. Algorithm 2 shows a version of L-PST with this feature.

3.2 Early Exit Strategy

It is possible to further curtail the number of game swaps with a strategy termed *early exit*. Recall that the generic *partialSwapTeams*(t_i, t_j, r) procedure first swaps a game between t_i and t_j in round r . Team t_i loses some game against an opponent t_x with this first swap. Ordinarily, if this round

¹The opponents in the swaplist that occur within the gap framed by the look ahead opponent are skipped by the *partialSwapTeams* portion of the L-PST procedure, therefore selecting a different look ahead opponent will result in a different schedule.

Data: (t_i, t_j, r)
simulate generic $partialSwapTeams(t_i, t_j, r)$ procedure to produce a
swaplist for t_j ;
select look ahead opponent t_l ;
if t_l available **then**
 $swapHomes(t_j, t_l)$;
 $partialSwapTeams(t_i, t_j, r)$;
 $swapHomes(t_j, t_l)$;
else
 $partialSwapTeams(t_i, t_j, r)$;
end

Algorithm 2: Basic Lookahead Partial Swap Teams algorithm

was t_i 's home game against t_x (for example), swapping would continue until a home game against t_x is received from t_j . However, it is possible to stop earlier, if an *away* game against t_x is received instead. Stopping at this point leaves the schedule in a state which is not double round-robin; however, it can be repaired by noticing that t_i has two away games against t_x , and flipping one of the two games at random to be a home game restores the double round-robin property for t_i . t_j has a similar problem, with two home games against t_x , one of which must be flipped to an away game. Of course, the procedure must also correct the two corresponding games for t_x .

In general, the early exit strategy can be followed in a $partialSwapTeams(t_i, t_j, r)$ move when t_i regains a game which has the original round r opponent, but with the wrong home/away status.

3.3 Combining Look Ahead with Early Exit

The lookahead and early exit strategies can be combined to reduce the number of game swaps. For example, suppose that the L-PST simulation step in Algorithm 2 produced the following swaplist:

$$\{\overbrace{A, B, C, D, E}^5, @A, @F, B, C, D, E, F\} \quad (3)$$

Choosing B, C, D, or E as the look ahead opponent, each with a gap of 6, is certainly a big improvement. But it is also possible to choose A (with a gap of 5), and exit early with @F rather than continuing with 5 additional swaps to get F. In general, to choose the look ahead opponent penalties are provisionally added to the $swapHomes$ preconditioning step and the early

exit (when applicable), and the shortest overall path is constructed to either the final opponent in the swaplist, or an earlier match against the same opponent.

The complete version of L-PST is summarized in Algorithm 3.

Data: (t_i, t_j, r)
simulate generic $partialSwapTeams(t_i, t_j, r)$ to create a swaplist for t_j ;
select look ahead opponent t_l ;
if t_l available **then**
 $swapHomes(t_j, t_l)$;
 $partialSwapTeams(t_i, t_j, r)$;
 $swapHomes(t_j, t_l)$;
else
 $partialSwapTeams(t_i, t_j, r)$;
end
if early exit was performed **then**
 repair schedules for t_i, t_j and opponent to restore double round-robin property;
end

Algorithm 3: Lookahead Partial Swap Teams with optional early exit

4 Experiments

The shortest known schedules for the TTP challenge problems NL10, CIRC10, and Galaxy10 were all discovered by the author using Lookahead PST. These schedules were recently proven optimal in [6]. Note that the optimal schedule was not found in every run; problems of this small size execute relatively quickly and were used for test purposes with dozens of runs for each.

L-PST was also used to find all of the current best known schedules for the larger Galaxy problems. The results for the largest problems are summarized in Table 3.

All results were obtained using a single desktop computer, using a simplified version of the TTSA algorithm without reheats [1], starting from random initial schedules. Other aspects of the TTSA algorithm including the objective function were also modified. The proposal distribution used in TTSA is not described in [1]; as an indication of the relative importance of L-PST, in the largest challenge problem (Galaxy with $n=40$) the initial

Problem size	Previous best	L-PST best	Improvement
36	207,117	177,090	14.5%
38	253,279	214,546	15.3%
40	304,689	258,899	15.0%

Table 3: Summary of results for the largest Galaxy challenge problems.

proposal distribution used suggests L-PST for approximately 66 % of all move attempts.

5 Conclusion

An improved search neighbourhood for the Traveling Tournament Problem is proposed which uses a derivative of the *partialSwapTeams* move. New mechanisms were outlined which identify shortcuts to reduce the number of game swaps. With a reduced number of game swaps, on average, the goal is to avoid becoming trapped in a local minimum during a search procedure. The proposed Lookahead Partial Swap Teams or L-PST algorithm used in a simulated annealing context appears to be effective in finding optimal schedules for problems up to 10 teams, and very good, short distance schedules for a wide range of larger challenge problems. Whether the solutions to larger problems are close to optimality or not remains to be proven.

References

- [1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, Y. Vergados. *A Simulated Annealing Approach to the Traveling Tournament Problem*, Journal of Scheduling, Volume 9, Issue 2 pp. 177-193. Kluwer Academic Publishers, 2006.
- [2] Kelly Easton, George Nemhauser, Michael Trick. *The Traveling Tournament Problem Description and Benchmarks*, Principles and Practices of Constraint Programming (CP 2001), Volume 2239 of Lecture Notes in Computer Science, pp. 580-585. Springer Berlin/Heidelberg, 2001.
- [3] Luca di Gaspero, Andrea Schaerf. *A Composite-Neighborhood Tabu Search Approach to the Traveling Tournament Problem*, Journal of Heuristics, Volume 13, Issue 2 pp. 189-207. Kluwer Academic Publishers, 2007.

- [4] Celso Ribeiro, Sebastián Urrutia. *Heuristics for the Mirrored Traveling Tournament Problem*, European Journal of Operational Research, 2007.
- [5] Michael Trick. <http://mat.gsia.cmu.edu/TOURN/>, 2010.
- [6] David Uthus. <http://www.cs.auckland.ac.nz/~duth001/>, 2010.