The Timetable Constrained Distance Minimization Problem

Rasmus V. Rasmussen[1] and Michael A. Trick[2]

[1] Department of Operations Research, University of Aarhus, Ny Munkegade,
Building 1530, 8000 Aarhus C, Denmark
[2] Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. We define the *timetable constrained distance minimization problem* (TCDMP) which is a sports scheduling problem applicable for tournaments where the total travel distance must be minimized. The problem consists of finding an optimal home-away assignment when the opponents of each team in each time slot are given. We present an integer programming, a constraint programming formulation and describe two alternative solution methods: a hybrid integer programming/constraint programming approach and a branch and price algorithm. We test all four solution methods on benchmark problems and compare the performance. Furthermore, we present a new heuristic solution method called the *circular traveling salesman approach* (CTSA) for solving the *traveling tournament problem*. The solution method is able to obtain high quality solutions almost instantaneously, and by applying the TCDMP, we show how the solutions can be further improved.

Keywords Timetabling; Integer Programming; Constraint Programming; Sports scheduling.

1 Introduction

Sports scheduling has proved to be a research area containing a large number of highly applicable problems which are very hard to solve. Furthermore, good solutions may lead to great savings in travel costs due to reduced travel distances or may increase the revenue earned from TV stations since special requirements can be satisfied. From an operations research perspective this makes sports scheduling constitute an ideal research area.

Two problems have dominated the field of sports scheduling: the problem of minimizing the total travel distance and the problem of minimizing the total number of *breaks*. A break is two consecutive home games or two consecutive away games and the number of breaks is normally minimized when teams return home after each away game. In this way teams get alternating patterns of home and away games and the supporting fans are able to watch their team regularly throughout the season. However, in some leagues, the teams move directly from one away game to the next without returning home due to the large distances. In this kind of tournament, the travel costs constitute a substantial part of the overall costs and a schedule minimizing the total travel distance is preferred. The *traveling tournament problem* (TTP) [5] captures the fundamental difficulties when scheduling tournaments with break and distance aspects and many solution methods have been developed for solving this problem, see for instance [1, 3, 6, 20, 24].

In the literature concerning break minimization, a decomposition approach separating the problem into two subproblems is often used [4, 8–10, 16, 17, 21]. One of the subproblems finds a home-away assignment, while the other problem determines when the teams meet. The order of these subproblems varies but Trick [22] suggests that the most constrained subproblem should be solved first. When the timetable for the schedule

is solved first, without assignment of home/away games, the second phase is then the *break minimization problem* of finding a home-away assignment minimizing the total number of breaks. The problem has received much attention and effective solution methods have been obtained. Régin [18] solves the problem using constraint programming, Trick [22] uses integer programming and later Elf, Jünger, and Rinaldi [7] and Miyashiro and Matsui [15] formulate the problem as a maximum cut problem and present corresponding solution methods.

In this paper we consider a generalization to the break minimization problem when distances are considered instead of breaks. The problem is denoted the *timetable constrained distance minimization problem* (TCDMP) and it applies for example when leagues face many requests from TV networks. In order to increase the revenue earned from TV networks, the sports leagues must be able to accommodate the requests from the TV networks. Typically, these requests concern high-quality games since the TV networks want to schedule these games in such a way that the number of viewers is maximized. When many of these requests are present, it may be necessary to determine all the opponents before finding a home-away assignment and, in this case, a good solution for the TCDMP becomes important.

Approaches for the break minimization cannot be applied directly to handle the distance minimization problem. For instance, in maximum cut approaches, a graph is created the goal is to find a bipartition of the nodes so as to maximize the number of edges across the cut. An edge across the cut corresponds to a non-break, so maximizing the number of non-breaks is equivalent to minimizing the number of breaks. Once distances are added, however, the objective information is now no longer just embedded in edges across a cut. Such edges correspond to travel between a home game and an away game, or vice versa, but travel corresponding to consecutive away games are not represented in the cut. So it does not seem that a straightforward modification of maximum cut approaches can be applied to TCDMP.

To solve the TCDMP we present and compare four solution methods. The problem is formulated as an IP model and a CP model which are solved using CPlex [12] and Solver [13], respectively. Furthermore, we present a hybrid IP/CP solution method utilizing the strengths of both techniques by using CP for solving feasibility problems and IP for solving an optimization problem. Finally, the problem is solved by using a branch and price algorithm. It would be appealing to try to formulate this problem as a weighted maximum cut problem, mimicking the work of Elf et al. [7] and Miyashiro and Matsui [15], but it is not possible to accurately model the distance traveled through just the cut values.

In addition to the TCDMP we also present a new heuristic solution method for the TTP. This method called the *circular traveling salesman approach* (CTSA) takes advantage of already obtained solutions for one instance class of the TTP to obtain solutions for the general TTP problem. By doing so, CTSA is able to obtain solutions almost instantaneously since it only needs to solve a traveling salesman problem with up to 16 nodes. The solutions obtained are close to the known upper bounds and, besides being used as final schedules, they can be used as initial solutions for some of the metaheuristic solution methods for the TTP.

In the following section, we give a short introduction to sports scheduling, define the TTP and the TCDMP and present the IP and the CP formulations for the TCDMP. In Section 3, we outline the hybrid IP/CP approach and Section 4 explains the branch and price algorithm. Afterwards, Section 5 presents the computational results of the 4 solution methods for the TCDMP before Section 6 outlines the CTSA and presents the solutions for some of the benchmark problems. Finally, we give some concluding remarks in Section 7.

2 Problem Formulation

In this section we will give a formal definition of both the TTP and the TCDMP and present integer and constraint programming formulations for the TCDMP. Before we define the problems, let us give a brief introduction to some of the sports scheduling terminology.

We consider *double round robin tournaments* with an even number of teams n (an odd number of teams can be handled by adding a dummy team, representing a bye for a team). A double round robin tournament consists of two games between all pairs of teams leading to a total of $n(n-1)$ games. We assume that these games are distributed evenly over $2(n-1)$ *time slots* such that all teams play exactly one game in each slot.

All teams have a home venue and play one game against all other teams at this venue. These games are called *home games* while games played at another venue are called *away games*. If a team plays two consecutive home games or away games, we say that the team has a *break* in the last of the two slots.

In Figure 2.1, we give an example of a schedule for a double round robin tournament. The columns correspond to slots while the rows correspond to teams and the entry (i, s) gives the opponent of team i in slot s. An @ means that the team plays away against the opponent stated right after the @ and we have highlighted breaks using boldface.

Slot	1	2	3	4	5	6	7	8	9	10
Team 1:	@6	3	@5	2	@4	6	@3	5	@2	4
Team 2:	@5	6	**4**	@1	3	**5**	@6	**@4**	1	@3
Team 3:	4	@1	6	**5**	@2	**@4**	1	@6	**@5**	2
Team 4:	@3	5	@2	**@6**	1	**3**	@5	2	**6**	@1
Team 5:	2	@4	1	@3	6	@2	4	@1	3	@6
Team 6:	1	@2	**@3**	4	@5	**@1**	2	**3**	@4	5

Fig. 2.1. Schedule for a double round robin tournament with 6 teams.

A schedule for a round robin tournament can be separated into two components, a *timetable* and a *pattern set*. The timetable gives the opponent of each team in each slot without considering venues. The pattern set consists of a *pattern* for each team and these patterns are vectors with an entry for each slot, saying whether the corresponding team plays home or away. The combination of a timetable and a pattern set gives a schedule for the tournament. Figure 2.2 shows the timetable and the pattern set for the schedule shown in Figure 2.1. In the pattern set, 1 represents a home game while 0 represents an away game.

Slot	1	2	3	4	5	6	7	8	9	10
Team 1:	0	1	0	1	0	1	0	1	0	1
Team 2:	0	1	**1**	0	1	**1**	0	**0**	1	0
Team 3:	1	0	1	**1**	0	**0**	1	0	**0**	1
Team 4:	0	1	0	**0**	1	**1**	0	1	**1**	0
Team 5:	1	0	1	0	1	0	1	0	1	0
Team 6:	1	0	**0**	1	0	**0**	1	**1**	0	1

(a)

Slot	1	2	3	4	5	6	7	8	9	10
Team 1:	6	3	5	2	4	6	3	5	2	4
Team 2:	5	6	4	1	3	5	6	4	1	3
Team 3:	4	1	6	5	2	4	1	6	5	2
Team 4:	3	5	2	6	1	3	5	2	6	1
Team 5:	2	4	1	3	6	2	4	1	3	6
Team 6:	1	2	3	4	5	1	2	3	4	5

(b)

Fig. 2.2. (a) Pattern set, (b) Timetable.

A *repeater* in a schedule are two consecutive games between the same opponents.

Using this terminology we are now able to define the TTP and the TCDMP.

Definition 1 (TTP [5]). *Given n teams, a distance matrix specifying the distance between the venues and an upper bound UB on the number of consecutive home and consecutive away games, find a feasible schedule without repeaters which minimizes the total distance traveled by all teams. It is assumed every team begins and ends at its home venue.*

Multiple instance classes of the TTP exist and among them is the circular distance TTP. In this version all teams are placed on a circle with a distance of exactly 1 to both neighbors and it is assumed that the teams travel on the circle when they visit each other. This problem will be used in the heuristic solution method for the TTP.

Definition 2 (TCDMP). *Given a timetable for a double round robin tournament with n teams, a distance matrix specifying the distances between the venues and an upper bound UB on the number of consecutive home and consecutive away games, find a feasible pattern set which minimizes the total distance traveled by all teams.*

We have modelled the TCDMP using both IP and CP and the models are presented in the following two subsections. In the rest of the paper, we let n denote the number of teams, while T denotes the set of teams. The set of slots is denoted S, and we let $S^0 = S \cup \{0\}$. The distance matrix is represented by D, and entrance $D_{i_1 i_2}$ contains the distance between the venue of team i_1 and the venue of team i_2. TT denotes the timetable and entrance TT_{is} gives the opponent of team i in slot s. Notice that $D_{TT_{is}TT_{is+1}}$ is the travel distance of team i between slots s and $s+1$ if team i plays away in both slots.

2.1 Integer Programming Formulation

To formulate the problem as an IP model, we use a binary variable h_{is} for each $i \in T$ and each $s \in S$. h_{is} equals 1 if team i plays home in slot s and it equals 0 if it plays away. To calculate the total travel distance, we use an integer variable, d_{is} for each $i \in T$ and each $s \in S^0$, which is equal to the distance team i travels between slot s and slot $s+1$. We use the dummy slots 0 and $2n-1$ to make sure that all teams start and end at home. This gives the following IP model.

$$\min \sum_{i \in T} \sum_{s \in S^0} d_{is} \tag{2.1}$$

$$\text{s.t.} \quad d_{is} \geq (1 - h_{is} - h_{is+1}) D_{TT_{is}TT_{is+1}} \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.2}$$

$$d_{is} \geq (h_{is} - h_{is+1}) D_{iTT_{is+1}} \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.3}$$

$$d_{is} \geq (-h_{is} + h_{is+1}) D_{TT_{is}i} \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.4}$$

$$h_{i0} = 1 \qquad \forall i \in T \tag{2.5}$$

$$h_{i2n-1} = 1 \qquad \forall i \in T \tag{2.6}$$

$$\sum_{s \in S} h_{is} = n - 1 \qquad \forall i \in T \tag{2.7}$$

$$h_{i_1 s} + h_{i_2 s} = 1 \qquad \forall i_1, i_2 \in T,\ i_1 < i_2,\ \forall s \in S,\ TT_{i_1 s} = i_2 \tag{2.8}$$

$$h_{is_1} + h_{is_2} = 1 \qquad \forall i \in T,\ \forall s_1, s_2 \in S,\ s_1 < s_2,\ TT_{is_1} = TT_{is_2} \tag{2.9}$$

$$\sum_{s=\hat{s}}^{\hat{s}+UB} h_{is} \leq UB \qquad \forall i \in T,\ \forall \hat{s} \in \{1,\ldots,2(n-1)-UB\} \tag{2.10}$$

$$\sum_{s=\hat{s}}^{\hat{s}+UB} h_{is} \geq 1 \qquad \forall i \in T,\ \forall \hat{s} \in \{1,\ldots,2(n-1)-UB\} \tag{2.11}$$

$$h_{is} \in \{0,1\} \qquad \forall i \in T,\ \forall s \in \{0,\ldots,2n-1\} \tag{2.12}$$

$$d_{is} \in \mathbb{Z}_+ \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.13}$$

Constraints (2.2) - (2.4) give lower bounds on the distance team i travels between slots s and $s+1$ when i plays away in the two slots, when it plays home and away and when it plays away and home, respectively. Constraints (2.5) and (2.6) ensure that all teams start and end at home and constraints (2.7) make sure that all teams have exactly $n-1$ home games. Constraints (2.8) require that, when teams i_1 and i_2 meet in slot s, one of the teams must play home and the other must play away, while constraints (2.9) make sure that team i plays one home game and one away game in two slots with the same opponent. Finally, the constraints (2.10) and (2.11) give upper bounds on the number of consecutive home games and the number of consecutive away games.

2.2 Constraint Programming Formulation

When formulating the problem as a CP model, we use variables similar to the variables used in the IP model, but the CP model allows us to reformulate the constraints. In particular, we are able to formulate the constraints (2.7), (2.10) and (2.11) as a single constraint called *sequence* and we can use logical expressions to determine the travel distance. This gives the following CP model.

$$\min \sum_{i \in T} \sum_{s \in S^0} d_{is} \tag{2.14}$$

$$\text{s.t.}\ \ (h_{is}=1) \wedge (h_{is+1}=1) \Rightarrow (d_{is}=0) \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.15}$$

$$(h_{is}=0) \wedge (h_{is+1}=1) \Rightarrow (d_{is}=D_{TT_{is}i}) \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.16}$$

$$(h_{is}=1) \wedge (h_{is+1}=0) \Rightarrow (d_{is}=D_{iTT_{is+1}}) \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.17}$$

$$(h_{is}=0) \wedge (h_{is+1}=0) \Rightarrow (d_{is}=D_{TT_{is}TT_{is+1}}) \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.18}$$

$$sequence(1, UB, UB+1, [h_{i1},\ldots,h_{i2n-2}], [1], [n-1]) \qquad \forall i \in T \tag{2.19}$$

$$h_{is_1} \neq h_{is_2} \qquad \forall i \in T,\ \forall s_1, s_2 \in S,\ s_1 < s_2,\ TT_{is_1} = TT_{is_2} \tag{2.20}$$

$$h_{is} \neq h_{TT_{is}s} \qquad \forall i \in T,\ \forall s \in S \tag{2.21}$$

$$h_{i0} = 1 \qquad \forall i \in T \tag{2.22}$$

$$h_{i2n-1} = 1 \qquad \forall i \in T \tag{2.23}$$

$$h_{is} \in \{0,1\} \qquad \forall i \in T,\ \forall s \in \{0,\ldots,2n-1\} \tag{2.24}$$

$$d_{is} \in \mathbb{Z}_+ \qquad \forall i \in T,\ \forall s \in S^0 \tag{2.25}$$

The constraints (2.15) - (2.18) determine the travel distance of team i between slots s and $s+1$, depending on whether team i plays home or away in the two slots. The *sequence constraints* (2.19) are global constraints with the syntax $sequence(min, max, width, varVec, valueVec, cardVec)$ where min, max and $width$ are

numbers, $varVec$ is a vector of variables and $valueVec$ and $cardVec$ are vectors with the same index set. The constraints are satisfied if each entrance in $cardVec$ equals the number of variables in $varVec$, which are equal to the corresponding entrance in $valueVec$ and for each value in $valueVec$ at least min and at most max variables are equal to this value in any subsequence of length $width$. In our case the constraint says that team i must play exactly $n-1$ home games and in $UB+1$ consecutive slots it cannot play less than one home game or more than UB home games. Constraints (2.20) state that team i must play one home game and one away game in two slots where it meets the same opponent and constraints (2.21) require that the opponent of team i plays home (away) if team i plays away (home). Constraints (2.22) and (2.23) make sure that all teams start and end home.

The sequence constraint is a key aspect of this formulation. Filtering for the sequence constraint is currently an active research area. In our work, we rely on the filtering algorithm in ILOG Solver [13] based on the work in [19]. van Hoeve et al. [25] survey the strength of alternative sequence filtering approaches.

In the following sections, we present a hybrid IP/CP approach and a branch and price algorithm for solving the TCDMP. A Benders decomposition approach similar to the method presented in [17] has also been implemented but the reduction in time used to solve the master problem cannot offset the additional iterations which are required compared to the hybrid IP/CP approach.

3 Hybrid IP/CP Approach

The first of the specialized solution methods is a hybrid IP/CP approach which decomposes the problem into two phases. Phase 1 generates all feasible patterns for each team in the tournament and Phase 2 finds the optimal pattern set by assigning each team to one of the patterns found in Phase 1.

CP is in general very effective at solving feasibility problems and it has the advantage of being able to find all solutions to a specific problem instead of a single solution. These characteristics make a CP model ideal for finding feasible patterns in Phase 1. On the other hand, IP is typically stronger than CP when it comes to optimization problems, since the linear relaxation can be used to prune suboptimal solutions. Therefore IP is used in Phase 2 to choose the optimal patterns from the patterns generated in Phase 1. The details of the two phases are explained below.

3.1 Phase 1

In order to generate all feasible patterns, we use a CP model for each team and find all feasible solutions to each of the models. Each pattern must contain exactly $n-1$ home games and satisfy the upper bound on the number of consecutive home games and consecutive away games. Furthermore, a pattern for a specific team i must satisfy that, for all pairs of slots s_1 and s_2 where $TT_{is_1} = TT_{is_2}$, the pattern has both a home game and an away game.

To formulate a CP model for finding all feasible patterns of team i, we use a binary variable h_s for each $s \in S$. As in the earlier sections, $h_s = 1$ implies a home game in slot s and $h_s = 0$ implies an away game.

The CP model for team i looks as follows.

$$\text{solve:}$$

$$sequence(1, UB, UB+1, [h_1, \ldots, h_{2(n-1)}], [1], [n-1]) \tag{3.1}$$

$$h_{s_1} \neq h_{s_2} \qquad \forall s_1, s_2 \in S,\ s_1 < s_2,\ TT_{is_1} = TT_{is_2} \tag{3.2}$$

$$h_s \in \{0, 1\} \qquad \forall s \in S \tag{3.3}$$

Constraint (3.1) corresponds to constraint (2.19) in the CP formulation and makes sure that the number of home games is correct and the upper bound on consecutive home games and consecutive away games is satisfied. Constraints (3.2) correspond to the constraints (2.20) and make sure that the patterns are feasible with respect to the timetable.

For each team $i \in T$, we let P_i denote the set of feasible patterns and for each $j \in P_i$, we let h_{js} represent the entry h_s of pattern j. We also calculate the distance team i must travel, if it uses pattern j, and denote it d_{ij}. The distance can be calculated since the timetable gives us the opponent of each slot and the pattern tells if team i plays home or away.

3.2 Phase 2

In Phase 2, we must find an optimal allocation of each team i to a pattern $j \in P_i$, such that the total travel distance is minimized and the pattern set is feasible with respect to the timetable.

To formulate an IP model for this problem, we use a binary variable x_{ij} for each $i \in T$ and each $j \in P_i$. The variable is 1 if team i is assigned to pattern j and 0 otherwise. We also use the home-away parameter h_{js} for each pattern j and each slot s and the distance parameter d_{ij} for each team i and each pattern $j \in P_i$.

$$\min \sum_{i \in T} \sum_{j \in P_i} d_{ij} x_{ij} \tag{3.4}$$

$$\text{s.t.} \sum_{j \in P_i} x_{ij} = 1 \qquad \forall i \in T \tag{3.5}$$

$$\sum_{i \in \{i_1, i_2\}} \sum_{j \in P_i} h_{js} x_{ij} = 1 \qquad \forall i_1, i_2 \in T,\ i_1 < i_2,\ \forall s \in S,\ TT_{i_1 s} = i_2 \tag{3.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in T,\ \forall j \in P_i \tag{3.7}$$

Constraints (3.5) are the assignment constraints saying that all teams must be assigned to a feasible pattern and constraints (3.6) make sure that when 2 teams meet, one of the teams play home and the other plays away. These two constraints are enough to ensure a feasible pattern set with respect to the timetable since we know from Phase 1 that all the teams play one home game and one away game in two slots where they meet the same opponent.

4 Branch and Price

In addition to the hybrid IP/CP approach, we developed a branch and price algorithm (see Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance [2] for a detailed description of branch and price) to solve the TCDMP. This method has successfully been applied to the TTP and it is to date the best exact solution method for the TTP.

The branch and price algorithm assigns teams to patterns by solving a linear programming (LP) problem which is restricted to only contain a subset of the feasible patterns instead of all the feasible patterns. This problem is known as the master problem.

The solution of the master problem may be fractional since we use an LP problem and it might not be optimal since we consider only a subset of the patterns. The optimality issue is handled by solving a pricing problem which finds patterns to the master problem with negative reduced costs. These patterns are added to the master problem and it is re-solved. If no patterns with negative reduced costs exist and the solution is fractional, the algorithm uses branch and bound to obtain an integer solution.

There are a number of pieces that go into a branch and price approach. These include:

1. the approach for finding an initial solution,
2. a method for solving the master problem,
3. a method for solving the subproblem,
4. a branching rule consistent with the subproblem solution method, and
5. an order in which to branch on the nodes.

Since our approach is a relatively standard branch and price approach, we will summarize our choices.

In order to find an initial feasible pattern set, we use a CP model. The model corresponds to the CP model presented in Section 2.2 but, in this model, we ignore the travel distance, since we are only looking for a feasible solution. We have tested the effect of generating an additional number of good patterns initially. In this way fewer patterns have to be generated during the search and less time is spent on solving the pricing problem. However, the overall computation time increased when this approach were used since the computation time of the master problem increased and additional branching took place before the optimal solution were found.

The master problem of the branch and price algorithm is almost identical to the linear relaxation of the problem solved in Phase 2 of the hybrid IP/CP approach. The only differences are that the problem is restricted since P_i does not contain all the feasible patterns of team i, and that a number of branching constraints are added. The number of branching constraints corresponds to the level of the current node in the branch and bound tree. The master problem is solved as a linear program.

When the master problem has been solved we use a pricing problem for finding patterns with negative reduced costs. To solve the pricing problem, we use an IP model since we want to minimize the reduced cost. Alternatively, a CP model could be used to generated patterns with negative reduced costs but we have obtained the best results when the reduced cost is minimized. The pricing problem is solved for each team and in case the reduced cost is less than zero, we add the pattern to the master problem. When the pricing problem has been solved for all teams, we re-solve the master problem if patterns have been added. Otherwise we need to branch.

Instead of branching on one of the fractional values from the master problem, we use higher order branching. We choose a team $\hat{i}$ and a slot $\hat{s}$ and create two new nodes by letting team $\hat{i}$ play home in slot $\hat{s}$ in one of the nodes and away in the other. We have two alternative methods for finding the team and slot. In the first, we find a team and slot where the current solution is fractional, but close to 1; in the second, we find a team and slot where it is fractional and close to 0.5.

The final piece is to define the node selection approach in exploring the branch and bound tree. Again, we have two approaches.

The first strategy is the well-known *depth first strategy*. This strategy chooses one of the child nodes of the current strategy if any exists and otherwise it backtracks. The strategy corresponds to a *last in, first out* (LIFO) strategy since it always chooses the last node which has been added.

The second strategy is a *best lower bound strategy* which attempts to find a node that is likely to include a good solution. If there is a lower bound at each node, then nodes with smaller lower bounds may contain better solutions for our minimization. In our case, we use the objective value of the parent node as lower bound and therefore the strategy chooses the node with the highest parent value. Ties are broken arbitrarily.

With these pieces, we have a branch and price approach where linear programming is used to solve the master problem, constraint programming is used to generate initial columns, and integer programming is used to generate new patterns relative to the dual prices of the master problem. There are two choices for branching and two node selection approaches, for a total of four variants.

5 Computational Results for the TCDMP

In order to explore the computational complexity of the TCDMP and to compare the proposed solution methods, we have tested all 4 methods on 60 instances ranging from 6 to 16 teams.

At the homepage http://mat.gsia.cmu.edu/TOURN/, Michael Trick's benchmark problems for the TTP can be found. These problems have been studied intensively and a number of solutions are presented at the web page. By letting $UB = 3$, using the presented distance matrices and permuting the slots of the presented TTP solutions, we have generated instances of the TCDMP.

For each even number of teams from 6 to 16 we have generated 10 instances by permuting slots of the following solutions. For 6 teams we have used the solution of Easton May 7, 1999; for 8 teams, the solution of Easton January 27, 2000; for 10 teams, the solution of Langford, June 13, 2005 and for 12, 14 and 16 teams, the solution of Zhang Xingwen August 28, 2002. All the tests have been performed on an Intel Xeon 2.67 GHz processor with 6 GB RAM. The IP and CP models have been solved by using OPL Studio [11] with the callable libraries CPLEX and Solver. The hybrid IP/CP approach and the branch and price algorithm have been implemented in OPL script.

The computational results are presented in Table 5.1. For each number of teams and each solution method, the table shows the number of instances solved and the minimum, average and maximum time used on the solved instances. We have used a time limit of 1800 seconds and instances which have not been solved within this time limit are not included in the average. Since we have 2 node selection strategies and two branching strategies for the branch and price algorithm, we present four versions of this solution method: BP-df-1, BP-df-2, BP-bv-1 and BP-bv-2. The terms df and bv refer to depth first and best value node selection, respectively, while 1 and 2 refer to the first and the second branching strategy.

Table 5.1 shows that, even though CP is better than IP at solving instances with 6 teams, it is not able to solve any of the instances with 8 teams. IP is doing a little better and is able to solve all the instances with less than 10 teams and a single instance with 10 teams. The fact that IP is able to handle larger instances than CP is no surprise, since IP models often excel compared to CP models when optimization problems are considered.

Both the hybrid IP/CP approach and the branch and price algorithm clearly outperform the IP and CP models. We see that the hybrid IP/CP approach shows the best results and, in addition to being the fastest on average, it is also the most stable of the solution methods. The only drawback of this method is a

Table 5.1. Computational Results for the TCDMP.

n	Solution method	Number solved	Time (s) Min.	Avg.	Max.
6	IP	10	0.42	22.51	217.18
6	CP	10	8.03	17.00	24.94
6	IP/CP	10	0.06	0.07	0.08
6	BP-df-1	10	0.83	1.59	3.34
6	BP-bv-1	10	0.82	1.27	2.29
6	BP-df-2	10	0.84	1.74	3.43
6	BP-bv-2	10	0.83	1.25	2.19
8	IP	10	60.72	397.35	954.56
8	CP	0	-	-	-
8	IP/CP	10	0.41	0.44	0.47
8	BP-df-1	10	2.87	11.74	21.65
8	BP-bv-1	10	2.88	6.99	11.90
8	BP-df-2	10	2.83	10.05	22.94
8	BP-bv-2	10	2.82	6.15	8.19
10	IP	1	1273.74	1273.74	1273.74
10	CP	0	-	-	-
10	IP/CP	10	1.79	2.02	2.34
10	BP-df-1	10	7.23	27.01	157.56
10	BP-bv-1	10	7.05	16.55	46.50
10	BP-df-2	10	6.98	24.66	64.49
10	BP-bv-2	10	7.03	15.51	44.34
12	IP	0	-	-	-
12	CP	0	-	-	-
12	IP/CP	10	10.16	12.19	18.19
12	BP-df-1	10	24.29	281.60	1386.09
12	BP-bv-1	10	23.77	130.31	434.97
12	BP-df-2	10	23.80	243.23	1038.98
12	BP-bv-2	10	23.56	151.74	650.79
14	IP	0	-	-	-
14	CP	0	-	-	-
14	IP/CP	10	35.52	38.07	42.83
14	BP-df-1	10	49.62	95.00	248.32
14	BP-bv-1	10	48.61	91.71	240.47
14	BP-df-2	10	48.69	165.40	750.18
14	BP-bv-2	10	48.74	72.78	164.82
16	IP	0	-	-	-
16	CP	0	-	-	-
16	IP/CP	10	153.80	197.16	260.47
16	BP-df-1	9	122.47	866.13	1770.94
16	BP-bv-1	9	119.37	827.90	1645.91
16	BP-df-2	8	121.42	662.81	1377.21
16	BP-bv-2	9	119.38	631.30	1382.13

rather large memory consumption since all patterns are generated initially. For instances with 16 teams, it generates up to 85000 patterns and uses approximately 200 MB of memory.

The computation times of the branch and price algorithm are highly dependent on the size of the branching tree and we see that there is an order of magnitude in difference between the minimum and maximum time. This means that the algorithm is only competitive with the hybrid IP/CP approach when an integer solution is found relatively fast in the branching tree.

The slowness of the branch and price approach is somewhat surprising. The vast majority of the patterns generated by the hybrid IP/CP approach are never used, so it seems advantageous to generate them as needed, as the branch and price approach does. For some instances, this indeed is a good thing: the fastest instances do solve faster than the hybrid approach. But for many instances, it appears that the columns being generated are the not the right ones. This appears to lead to lots of branching that is wasting time without moving towards optimality. Still, the branch and price approach may be practical in cases where the number of patterns generated is too large for the hybrid approach. So, for the 18 team instances, while none of the approaches were able to solve these instances within the given time limit, the hybrid IP/CP approach is quite impractical due to the 246,000 patterns generated. Branch and Price is able to at least generate feasible solutions even if optimal solutions cannot be found.

6 The Traveling Tournament Problem

In this section we present a new heuristic solution method for solving the TTP. This method, called the *circular traveling salesman approach* (CTSA), obtains solutions by combining solutions for the traveling salesman problem with solutions for the circular distance TTP. Furthermore, we show how the CTSA can be extended by solving the TCDMP afterwards in which case we will refer to it as the *extended circular traveling salesman approach* (ECTSA).

One of the main problems when solving the TTP is the fact that the travel distance is heavily affected by both the timetable and the pattern set. This makes it very hard to find a good decomposition of the problem and, without a decomposition approach, the problem is too hard to solve using exact solution methods even for instances with only eight teams. This has led to a number of highly specialized metaheuristic solution methods capable of finding very good solutions for all the instance classes of the TTP outlined at [23]. However, such algorithms are often cumbersome to implement and it may take a lot of fine tuning to obtain the best schedules.

The strength of the CTSA presented here is the fact that it is very simple to implement, solutions are found almost instantaneously and the solutions are not far from the best solutions currently found for the benchmark instances of the TTP. The basic idea is to use a two step approach. Step 1 approximates an instance of the TTP by an instance of the circular distance TTP, while Step 2 uses a solution for the circular distance TTP to obtain a solution for the TTP. The approximation is performed by finding the optimal traveling salesman tour through all the venues and then distributing the teams evenly on a circle with circumference n according to the order of the traveling salesman tour. This forms an instance of the circular distance TTP and by using a solution for this instance we have a schedule for the original TTP. This process is illustrated in figure 6.1.

The approach can be outlined as follows.

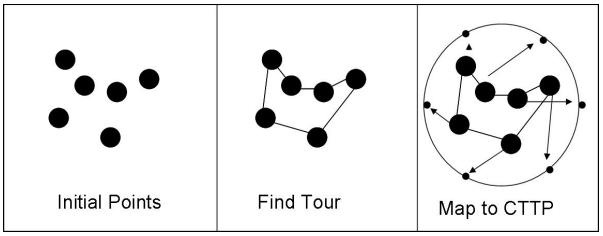

Fig. 6.1. Using the Circular TTP (CTTP) to approximate the TTP

The circular traveling salesman approach.

Step 1 Solve the traveling salesman problem for the teams in the tournament to give an ordering of the teams.

Step 2 Solve the circular distance TTP with teams ordered according to the solution from Step 1 and use the resulting schedule to calculate the travel distance when the real distance matrix is used.

The TSP from Step 1 is solved using a standard solution method. The problem is solved without the subtour elimination constraints and, in case the solution contains subtours, cuts are added and the problem is solved again. This process continues until a feasible solution has been obtained.

In Step 2, we do not actually solve the circular distance TTP since the currently best solutions for the problem, presented at [23], can be used. Note that there is only one circular distance TTP instance for each size of problem, so this "pre-processing" can be done once for every size n.

In this step, the team ordering 1,2,. . . ,n and the ordering n,1,2,. . . ,n-1 leads to the same objective value of the circular distance TTP - but they result in different schedules and hence different travel distances for the original TTP. This means that we are able to obtain $2n$ solutions from the TSP solution by rotating the original ordering and reversing the ordering. For example, the ordering 1,2,3, also leads to the orderings: 2,3,1; 3,1,2 and the reverse orderings 3,2,1; 2,1,3 and 1,3,2.

The CTSA has been tested on the benchmark instances: NL6, NL8,. . . , NL16 which can be found at [23] together with corresponding solutions. In Step 2 we need solutions for the circular distance TTP with 6 to 16 teams. For the instance with 10 teams, we use the solution by Langford found at [23] and for all other instances, we use the solutions obtained by Lim, Rodrigues, and Zhang [14]. In order to examine the effect of the $2n$ orderings, we have solved the problem for each ordering. The results are displayed in Table 6.1 which gives the best known upper and lower bound on the optimal travel distance, together with the maximum, minimum and average travel distance found by the CTSA for the $2n$ orderings. The solution method has been implemented in OPL script and it took less than 0.5 second to solve the TSP in all instances.

Keeping the simpleness of the CTSA in mind, the solutions are surprisingly close to the current upper bound. In addition, to be used as final schedules, the solutions can also be used as initial starting points in some of the metaheuristic approaches for the TTP. Lim et al. [14] use a beam search algorithm for obtaining initial solutions in their simulated annealing algorithm. But although the solutions obtained by the CTSA can be found almost instantaneously, the quality is comparable to the beam search. For comparison, Table 6.2 displays the computational results of the beam search algorithm reported in [14] found on a 2.53 GHz Pentium 4 PC with 512 MB of RAM.

Table 6.1. Computational results of the CTSA

			Travel distance		
Instance	LB	UB	Minimum	Maximum	Average
NL6	23916	23916	24467	26472	25559.8
NL8	39479	39721	41754	44028	42822.5
NL10	57500	59436	63844	67943	66106.7
NL12	107483	111248	116598	124975	121339.8
NL14	182797	189759	216659	230463	223925.1
NL16	248852	267194	288674	307925	295858.3

Table 6.2. Computational results of the beam search algorithm reported in [14].

	Travel distance			Time (s)		
Instance	Minimum	Maximum	Average	Minimum	Maximum	Average
NL6	24579	25146	24802.8	576	669	628.5
NL8	41265	42334	41852.6	3211	3443	3345.0
NL10	63337	65856	64544.2	5801	6299	6179.6
NL12	118047	123770	120528.8	8009	8730	8594.0
NL14	202916	208797	205929.3	10332	10947	10806.4
NL16	283795	295864	289235.2	12855	13889	13631.2

In order to improve the solutions of the CTSA, it is possible to add a Step 3 in which the pattern set is improved if possible. This is done by solving the TCDMP, given the original distance matrix and the timetable found in Step 2, to obtain the optimal pattern set given the timetable. The results for this extended version ECTSA are reported in Table 6.3 together with the computation times. Notice that the computation times are mainly due to the time needed to solve the TCDMP.

Table 6.3. Computational results of the ECTSA.

	Travel distance			Time (s)		
Instance	Minimum	Maximum	Average	Minimum	Maximum	Average
NL6	24467	26255	25401.5	0.05	0.19	0.07
NL8	41754	44028	42822.5	0.36	0.39	0.37
NL10	63277	66080	64635.6	2.47	2.74	2.55
NL12	116421	124588	120838.1	11.39	20.66	12.60
NL14	215665	230463	223586.9	48.71	58.07	50.43
NL16	288674	307925	295628.5	239.90	294.30	246.44

Since the pattern set and the timetable are already optimized in the solution of the circular distance, only marginal reductions can be obtained by solving the TCDMP. However, in case the obtained solutions will be used as final schedules, it might be worth spending the additional computation time required by the ECTSA. Although the ECTSA uses much more computation time than the CTSA it is still substantially faster than the beam search.

7 Conclusion

In this paper we have defined the TCDMP applicable for double round robin tournaments in which the timetable is determined before the pattern set and the total travel distance must be minimized. The problem is formulated as an IP model and a CP model and the computation times used to solve these models are compared with the time needed to solve two specialized algorithms. The first of the two algorithms is a hybrid IP/CP approach and the computational results show that this is the best of the four solution methods. The second algorithm is a branch and price algorithm and although the computational results are not as good as the hybrid approach it does provide solutions to almost all the instances.

Furthermore, we present the heuristic solution method CTSA for the TTP capable of finding high quality solutions in very short time. The speed of this approach makes it very well suited for obtaining initial solutions which can be used in the metaheuristic solution methods for the TTP. An interesting direction for future research would be to test the effect of using this kind of initial solutions.

We have evaluated the effect of solving the TCDMP after having applied the CTSA and in some cases the extended approach is able to improve the solutions. However, solving the TCDMP takes a relatively long time compared to the time needed in the CTSA and the improvements may not justify this increase in computation time. Nevertheless, the CTSA presents one way of obtaining a good timetable for which the TCDMP can be applied. Finding new ways of obtaining such timetables and perhaps better timetables would be a very interesting challenge.

References

[1] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. In *Proceedings CPAIOR'03, Montreal*, 2003.

[2] C. Barnhart, E.L. Johnson, G.L Nemhauser, M.W.P Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for huge integer programs. *Operations Research*, 46:316–329, 1998.

[3] T. Benoist, F. Laburthe, and B. Rottembourg. Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problem. In *Proceedings CPAIOR'01, Wye College (Imperial College), Ashford, Kent UK*, 2001.

[4] F. Della Croce and D. Oliveri. Scheduling the italian football league: an ilp-based approach. *Computers and Operations Research*, 33(7):1963–1974, 2006.

[5] K. Easton, G. Nemhauser, and M. Trick. The traveling tournament problem: Description and benchmarks. In *Proceedings CP'01*, volume 2239 of *Lecture Notes in Computer Science*, pages 580–585. Springer, 2001.

[6] K. Easton, G. Nemhauser, and M. Trick. Solving the traveling tournament problem: a combined integer programming and constraint programming approach. In E. Burke and P. De Causmaecker, editors, *PATAT 2002*, volume 2740 of *Lecture Notes in Computer Science*, pages 100–109. Springer, 2003.

[7] M. Elf, M. Jünger, and G. Rinaldi. Minimizing breaks by maximizing cuts. *Operations Research Letters*, 31:343–349, 2003.

[8] M. Henz. Scheduling a major college basketball conference - revisited. *Operations Research*, 49:163–168, 2001.

[9] M. Henz. Constraint-based round robin tournament planning. In D. De Schreye, editor, *Proceedings of the International Conference on Logic Programming, Las Cruces, New Mexico*, pages 545–557. MIT Press, 1999.

[10] M. Henz, T. Müller, and S. Thiel. Global constraints for round robin tournament scheduling. *European Journal of Operational Research*, 153:92–101, 2004.

[11] *ILOG OPL Studio 3.7, Language Manual*. ILOG, 2003.

[12] Inc. ILOG. *CPLEX 9.0 User Manual*. ILOG, 2003.

[13] Inc. ILOG. *ILOG Solver 5.0 User's Manual and Rference Manual*. ILOG, 2000.

[14] A. Lim, B. Rodrigues, and X. Zhang. A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research (In press)*.

[15] R. Miyashiro and T. Matsui. Semidefinite programming based approaches to the break minimization problem. *Computers and Operations Research*, 33(7):1975–1982, 2006.

[16] G.L. Nemhauser and M.A. Trick. Scheduling a major college basketball conference. *Operations Research*, 46(1):1–8, 1998.

[17] R.V. Rasmussen and M.A. Trick. A Benders approach for the constrained minimum break problem. *European Journal of Operational Research (In Press)*.

[18] J.-C. Régin. Minimization of the number of breaks in sports scheduling problems using constraint programming. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 57:115–130, 2001.

[19] J.-C. Regin and J.-F. Puget. A filtering algorithm for global sequencing constraints. In *Proceedings CP'97, volume 1330 of LNCS*. Springer, 1997.

[20] C.C. Ribeiro and S. Urrutia. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research (In press)*.

[21] A. Schaerf. Scheduling sport tournaments using constraint logic programming. *Constraints*, 4:43–65, 1999.

[22] M. Trick. A schedule-then-break approach to sports timetabling. In E. Burke and W. Erben, editors, *PATAT 2000*, volume 2079 of *Lecture Notes in Computer Science*, pages 242–252. Springer, 2001.

[23] M.A. Trick. Michael trick's guide to sports scheduling. URL http://mat.tepper.cmu.edu/sports/Instances/.

[24] P. Van Hentenryck and Y. Vergados. Traveling tournament scheduling: A systematic evaluation of simulated annealing. In J.C. Beck and B.M. Smith, editors, *Lecture Notes in Computer Science*, volume 3990, pages 167–181. Springer-Verlag Berlin Heidelberg 2006, 2006.

[25] W.-J. van Hoeve, G. Pesant, L.-M. Rousseau, and A. Sabharwal. Revisiting the sequence constraint. In *Proceedings CP'06, volume 4204 of LNCS*. Springer, 2006.