European Journal of Operational Research 000 (2015) 1–8

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Branch-and-bound with decomposition-based lower bounds for the Traveling Umpire Problem

Túlio A. M. Toffolo[a,b,*], Tony Wauters[a], Sam Van Malderen[a], Greet Vanden Berghe[a]

[a] KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC, Belgium
[b] Federal University of Ouro Preto, Department of Computing, Brazil

ARTICLE INFO

Article history:
Received 9 February 2015
Accepted 5 October 2015
Available online xxx

Keywords:
Traveling Umpire Problem
Decomposition strategies
Branch-and-bound

ABSTRACT

The Traveling Umpire Problem (TUP) is an optimization problem in which umpires have to be assigned to games in a double round robin tournament. The objective is to obtain a solution with minimum total travel distance over all umpires, while respecting hard constraints on assignments and sequences. Up till now, no general nor dedicated algorithm was able to solve all instances with 12 and 14 teams. We present a novel branch-and-bound approach to the TUP, in which a decomposition scheme coupled with an efficient propagation technique produces the lower bounds. The algorithm is able to generate optimal solutions for all the 12- and 14-team instances as well as for 11 of the 16-team instances. In addition to the new optimal solutions, some new best solutions are presented and other instances have been proven infeasible.

1. Introduction

The Traveling Umpire Problem (TUP) is a sports timetabling problem giving attention to the schedule of the umpires (referees). The goal is to assign the umpires to the matches of a tournament, whose schedule is given beforehand.

A double round robin tournament is considered, with $2n$ teams playing twice against each other – once in their home venue and once away. This results in a competition with $4n-2$ rounds, each consisting of n matches. Such tournament requires assigning n umpires to the games, with the objective to minimize their total travel distance. In order to obtain a fair schedule, hard constraints (a)–(e) are imposed:

(a) every match in the tournament is officiated by exactly one umpire;
(b) every umpire must work in every round;
(c) every umpire must visit the home venue of every team at least once;
(d) no umpire is in the same venue more than once in any q_1 consecutive rounds;
(e) no umpire officiates games of the same team more than once in any q_2 consecutive rounds. This constraint is similar to the previous one, but also takes the 'away team' into consideration.

The values q_1 and q_2 range respectively from 1 to n and 1 to $\lfloor \frac{n}{2} \rfloor$.[1]

Since the introduction of the TUP by Trick and Yildiz (2007), considering the Major League Baseball tournament, many exact and heuristic approaches have been developed. The initial work was extended (Trick & Yildiz, 2011) by a Benders cuts guided large neighborhood search. These papers also provided both Integer Programming (IP) and Constraint Programming (CP) formulations for the problem. A greedy matching heuristic and a simulated annealing approach using a two-exchange neighborhood were described by Trick, Yildiz, and Yunes (2012). Trick and Yildiz (2012) presented a Genetic Algorithm (GA) with a locally optimized crossover procedure. A stronger IP formulation and a relax-and-fix heuristic were proposed by de Oliveira, de Souza, and Yunes (2014), who improved both lower and upper bounds. Wauters, Van Malderen, and Vanden Berghe (2014) improved solutions and lower bounds by an enhanced iterative deepening search with leaf node improvements (IDLIs), an iterated local search (ILS) and a new decomposition based lower bound methodology. Further improvements for some instances were found by Toffolo, Van Malderen, Wauters, and Vanden Berghe (2014), who proposed a branch-and-price algorithm with a fast branch-and-bound for solving the pricing problems. Two branching strategies were investigated and many bounds were improved. Xue, Luo, and Lim (2015) presented two exact approaches to the TUP: a branch-and-bound algorithm relying on a Lagrangian relaxation for obtaining

* Corresponding author at: KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC, Belgium. Tel.: +32 9 265 87 04, +32476054944.
E-mail address: tulio.toffolo@kuleuven.be, tulio@toffolo.com.br (T.A.M. Toffolo).

[1] Trick and Yildiz (2007) originally presented the parameters d_1 and d_2 such that $q_1 = n - d_1$ and $q_2 = \lfloor \frac{n}{2} \rfloor - d_2$, with $0 \le d_1 < n$ and $0 \le d_2 < \lfloor \frac{n}{2} \rfloor$.

http://dx.doi.org/10.1016/j.ejor.2015.10.004

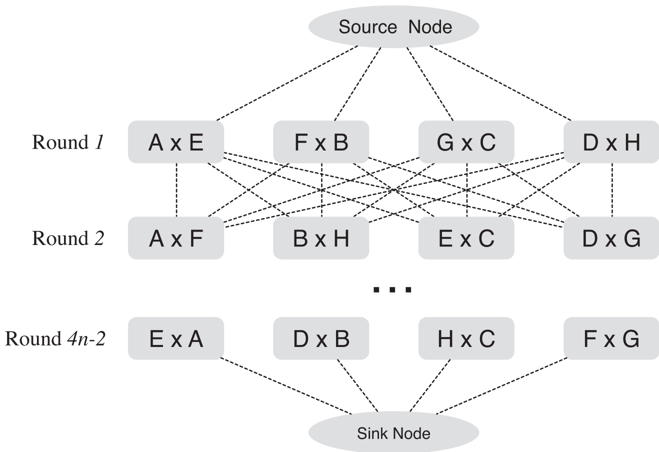

Fig. 1. Graph $G = (V, E)$ representing an 8-team TUP instance.

lower bounds and a branch-and-price-and-cut algorithm. The latter approach enabled solving two 14-team instances within the runtime limit of 48 hours. Several lower bounds were also improved.

In this work, we present a new branch-and-bound approach to the TUP. We introduce a simple decomposition scheme that, coupled with a propagation technique, results in very strong lower bounds. This enables increasing the size of all instances solved to optimality from 12 to 14 teams. In addition, many 16-team instances are solved.

The following section presents a formulation for the TUP based on the formulations introduced by Trick and Yildiz (2007) and de Oliveira et al. (2014). Section 3 details the proposed branch-and-bound technique while Section 4 discusses the lower bound strategies considered in the algorithm. Section 5 presents computational experiments considering both lower and upper bounds and, finally, Section 6 summarizes the conclusions.

2. Integer programming formulation for the TUP

We present a flow formulation for the TUP based on the formulations presented by Trick and Yildiz (2007) and de Oliveira et al. (2014). A graph $G = (V, E)$ is given, in which each node represents a game and directed edges connect the nodes (games) of round r to the nodes of round $r + 1$. This graph G also contains:

- a *source node*, f, and directed edges connecting f to the nodes representing games of the first round;
- a *sink node*, l, and directed edges connecting the nodes representing games of the last round to l.

Fig. 1 presents an example of this graph for an 8-team instance. The formulation considers the following input data:

d_e: distance of directed edge e;
I : set of teams $\{1, \ldots, 2n\}$;
H_i: set of nodes where team i plays at home;
R : set of rounds $\{1, \ldots, 4n-2\}$;
Q'_{ir}: set of nodes (games) of team i playing at home in rounds $R \cap \{r, \ldots, r+q_1-1\}$;
Q''_{ir}: set of nodes (games) of team i (home or away) in rounds $R \cap \{r, \ldots, r+q_2-1\}$;
U: set of umpires $\{1, \ldots, n\}$.

And the following variables:

$$x_{eu} = \begin{cases} 1 & \text{if edge } e \text{ is selected for umpire } u \\ 0 & \text{otherwise} \end{cases}$$

Finally, let $\delta(I)$ and $\omega(I)$ denote the set of edges that respectively enter and exit the nodes in I. The formulation of the problem is given by Eqs. (1)–(7).

$$\text{minimize} \quad \sum_{e\in E}\sum_{u\in U} d_e x_{eu} \tag{1}$$

$$\text{subject to} \quad \sum_{e\in\delta(j)}\sum_{u\in U} x_{eu} = 1 \quad \forall j \in V\backslash\{\text{source, sink}\} \tag{2}$$

$$\sum_{e\in\delta(j)} x_{eu} - \sum_{e\in\omega(j)} x_{eu} = \begin{cases} -1 & \text{if } j \text{ is the source} \\ +1 & \text{if } j \text{ is the sink} \\ 0 & \forall j \in V\backslash\{\text{source, sink}\}, \end{cases} \quad \forall u \in U \tag{3}$$

$$\sum_{e\in\delta(H_i)} x_{eu} \geq 1 \quad \forall i \in I; \forall u \in U \tag{4}$$

$$\sum_{e\in\delta(Q'_{ir})} x_{eu} \leq 1 \quad \forall i \in I; \forall r \in R; \forall u \in U \tag{5}$$

$$\sum_{e\in\delta(Q''_{ir})} x_{eu} \leq 1 \quad \forall i \in I; \forall r \in R; \forall u \in U \tag{6}$$

$$x_{eu} \in \{0, 1\} \quad \forall e \in E; \forall u \in U \tag{7}$$

The objective, given by Eq. (1), is to minimize the total distance traveled by the umpires. Constraints (2) ascertain that each game is officiated by exactly one umpire. Constraints (3) are flow preservation constraints, and together with the graph structure ensure that every umpire officiates exactly one game per round. If an umpire is at the location of a team in round r, the umpire must leave from the same location to go to the next location in round $r + 1$. This is also guaranteed by the flow preservation constraints. Constraints (4) state that every umpire must visit every location at least once during the season. Constraints (5) and (6) specify that every umpire must wait $q_1 - 1$ days to revisit the same home location and $q_2 - 1$ days to revisit the same team, respectively. Finally, constraints (7) specify that the variables considered are binary.

3. Branch-and-bound

Building on the branch-and-bound procedure established by Land and Doig (1960), we introduce a specialized decomposition-based algorithm to the TUP. This algorithm considers the same graph $G = (V, E)$ presented for the integer programming formulation in Section 2. Starting from the first round, the branch-and-bound algorithm assigns games to umpires, one at a time and round after round, until the sink node is reached. An assignment of a game to an umpire in a round is feasible if (*i*) the umpire did not visit the same location in the previous $q_1 - 1$ rounds and (*ii*) the umpire did not officiate any of the teams during the previous $q_2 - 1$ rounds. Whenever multiple games can be assigned to one umpire in one round, the algorithm greedily chooses the assignment incurring the smallest increase in travel distance. In case of ties, the games are sorted lexicographically.

If no valid assignment can be found for an umpire in a certain round, the procedure backtracks to the previous umpire and chooses the next game in the ordered list of games in the round. If the umpire considered is the first one of the round, then the algorithm returns to the previous round. This procedure continues until the sink node is reached for all umpires. If the resulting solution does not violate constraint (c), it is feasible and its total distance serves as an upper bound. This upper bound is, together with the calculated lower bounds, used to prune the parts of the search tree where no optimal solution can reside.