Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

Two exact algorithms for the traveling umpire problem

Li Xue[a,c], Zhixing Luo[b,*], Andrew Lim[b,c]

[a] School of Management, Xi'an Jiaotong University, No. 28, Xianning West Road, Xi'an, Shaanxi 710049, PR China
[b] International Center of Management Science and Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, PR China
[c] Department of Management Sciences, City University of Hong Kong, Tat Chee Ave, Kowloon Tong, Hong Kong

ARTICLE INFO

Article history:
Received 21 March 2014
Accepted 22 December 2014
Available online 30 December 2014

Keywords:
Umpire scheduling
Lagrangian relaxation
Column generation
OR in sports
Mathematical model

ABSTRACT

In this paper, we study the traveling umpire problem (TUP), a difficult combinatorial optimization problem that is formulated based on the key issues of Major League Baseball. We introduce an arc-flow model and a set partition model to formulate the problem. Based on these two models, we propose a branch-and-bound algorithm and a branch-and-price-and-cut algorithm. The branch-and-bound algorithm relies on lower bounds provided by a Lagrangian relaxation of the arc-flow model, while the branch-and-price-and-cut algorithm exploits lower bounds from the linear programming relaxation of the set partition model strengthened by a family of valid inequalities. In the branch-and-price-and-cut algorithm, we design an efficient label-setting algorithm to solve the pricing problem, and a branching strategy that combines three different branching rules. The two algorithms are tested on a set of well-known benchmark instances. The two exact algorithms are both able to rapidly solve instances with 10 teams or less, while the branch-and-price-and-cut algorithm can solve two instances with 14 teams. This is the first time that instances with 14 teams have been solved to optimality.

1. Introduction

The traveling umpire problem (TUP) is a relatively new combinatorial optimization problem that originates from Major League Baseball. Given a double round-robin tournament of $2n$ teams, in which each team plays against all of the other teams twice at home and away in a season of $4n-2$ time slots, the problem requires to determine an assignment of n umpires to the games in each time slot. Each game must have one umpire, and each umpire must be assigned to one game in each time slot. The objective is to minimize the total travel distance of the umpires. From another point of view, the TUP requires the design of a set of paths, starting from the first time slot and ending at the last time slot, that allow the umpires to visit every game exactly once throughout the season. To capture the most important features of umpire scheduling in the game, the TUP considers two additional requirements. The first, referred to as the frequency requirement, forbids any umpire to visit the home venue of any team more than once in any $n-d_1$ consecutive time slots and to see the same team more than once in any $\lfloor \frac{n}{2} \rfloor - d_2$ consecutive time slots, where d_1 and d_2 are two integer parameters that satisfy $0 \le d_1 \le n-1$ and $0 \le d_2 \le \lfloor \frac{n}{2} \rfloor - 1$, respectively. The second requirement, referred to as the visit requirement, enforces each umpire to visit the home venue of every team at least once throughout the season.

As the TUP is a relatively new problem, few research papers have addressed it to date. The problem was first introduced by Trick and Yildiz (2007), and was then further addressed by Trick and Yildiz (2011). Trick and Yildiz (2011) formulated the problem as an integer programming (IP) model and a constraint programming model, and designed a Bender's cuts guided large neighborhood search heuristic to deal with the problem. To test the models and the proposed heuristic, Trick and Yildiz (2011) generated a set of benchmark instances where the number of teams ranged from 4 to 32. Their computational results suggested that the TUP was very difficult to solve to optimality; the commercial IP solver ILOG CPLEX could only exactly solve instances with 10 teams at most. Moreover, finding a feasible solution for medium-sized instances was very difficult; both CPLEX and the large neighborhood search heuristic failed to find any feasible solutions in many medium-sized instances. The background and applications of the TUP, and the set of benchmark instances used by Trick and Yildiz (2011), were further explained by Trick, Yildiz, and Yunes (2012). These benchmark instances are available at http://mat.tepper.cmu.edu/TUP/. Trick et al. (2012) also proposed a simple greedy heuristic to generate initial solutions and a simulated annealing to further improve these initial solutions. Trick and Yildiz (2012) proposed a genetic algorithm to solve the TUP. In this genetic algorithm, when two parent solutions were crossed over to generate offsprings, these offsprings were partially optimized to improve their

* Corresponding author.
E-mail addresses: xltry1416@gmail.com, 771553487@qq.com (L. Xue), luozx.hkphd@gmail.com, luozhx@cityu.edu.hk (Z. Luo), lim.andrew@cityu.edu.hk (A. Lim).

http://dx.doi.org/10.1016/j.ejor.2014.12.038
0377-2217/© 2014 Elsevier B.V. All rights reserved.

qualities. This crossover operator is referred to as a *locally optimized crossover*. Experiments on the benchmark instances showed that this genetic algorithm could find better feasible solutions, or feasible solutions for some medium-sized instances, for which no feasible solution had been found before. de Oliveira, de Souza, and Yunes (2014) introduced a stronger IP model, based on which they implemented a relax-and-fix primal heuristic that iteratively solved relaxations of the IP model and progressively fixed variables until a feasible was found. Recently, Wauters, Van Malderen, and Vanden Berghe (2014) introduced a lower bound approach based on problem decomposition and a local search heuristic, which substantially improved the best-known lower bounds for many instances and find several new best feasible solutions, respectively. Toffolo, Van Malderen, Wauters, and Vanden Berghe (2014) proposed a branch-and-price algorithm, where a specialized branch-and-bound algorithm with multiple pruning techniques was applied to solve the pricing problem. It successfully improved several best-known lower bounds and upper bounds in the literature. The first complexity analysis on the TUP was conducted by de Oliveira, de Souza, and Yunes (2015).

In this paper, we propose two new IP models for the TUP: an arc-flow model and a set partition model. These two models use different methods to enforce the frequency requirement and the visit requirement. The arc-flow model uses an exponential number of constraints to eliminate solutions that violate either of the two requirements, while the set partition model defines a decision variable with respect to a path that satisfies both of the two requirements, leading to an exponential number of variables. The set partition model is further strengthened by the subset row (SR) inequalities, which were first applied in the vehicle routing problem (Jepsen, Petersen, Spoorendonk, & Pisinger, 2008). Although these two models cannot be solved directly by existing IP solvers (e.g. CPLEX and Lingo), a Lagrangian relaxation of the arc-flow model and the linear programming (LP) relaxation of the set partition model yield strong lower bounds. These lower bounds can be exploited to design effective exact algorithms. Therefore, we propose a branch-and-bound algorithm based on a Lagrangian relaxation and a branch-and-price-and-cut algorithm based on the LP relaxation. The two algorithms were extensively tested on the benchmark instances proposed by Trick and Yildiz (2011). Both the branch-and-bound algorithm and the branch-and-price-and-cut algorithm can rapidly achieve optimal solutions on instances with up to 10 teams. Moreover, the branch-and-price-and-cut algorithm can solve 2 instances with 14 teams to optimality in 48 hours due to the stronger lower bounds from the LP relaxation. This is the first time instances with 14 teams have been solved to optimality.

The remainder of this paper is organized as follows. First, we present the arc-flow model, the set partition model and the SR inequalities in Section 2. We describe the branch-and-bound algorithm in Section 3 and the branch-and-price-and-cut algorithm in Section 4. Section 5 is devoted to the computational results, and Section 6 concludes the paper with some closing remarks.

2. Mathematical models

Before presenting the two IP models, we introduce some necessary notations and definitions to simplify our presentation. Let $T = \{1, \dots, 2n\}$, $U = \{1, \dots, n\}$ and $S = \{1, \dots, 4n-2\}$ denote the set of teams, the set of umpires and the set of time slots in the season, respectively. Let G_t denote the set of games taking place in time slot t. Each game in G_t must have an umpire from U, while each umpire from U must be assigned to one game in G_t. Let $d_{i,j}$ $(i, j \in T)$ denote the travel distance between the home venues of team i and team j. We use a tuple (i, j) to represent the arc between two games, i and j, which take place in adjacent time slots. Let $G = \bigcup_{t\in S} G_t$ denote the set of games in the season. For each game $g \in G$, let g^- denote the home team and g^+ denote the guest team. Let $p = (g_1, \dots, g_m)^t$ denote a path that starts from game g_1 in G_t and ends at game g_m in G_{t+m-1}. Let G_p and E_p denote the set of games and the set of arcs in path p, respectively.

Definition 1. A path $p = (g_1, \dots, g_m)^t$ is complete if $t = 1$ and $m = 4n-2$.

Definition 2. A path $p = (g_1, \dots, g_m)^t$ is infeasible if it satisfies at least one of the following conditions:

1. there exist two games g_i and g_j $(g_i, g_j \in G_p)$ such that $i - j + 1 \le n - d_1$ and $g_i^- = g_j^-$;
2. there exist two games g_i and g_j $(g_i, g_j \in G_p)$ such that $i - j + 1 \le \lfloor \frac{n}{2} \rfloor - d_2$ and at least one of the following four equations is satisfied: $g_i^- = g_j^+, g_i^- = g_j^-, g_i^+ = g_j^-$ and $g_i^+ = g_j^+$;
3. p is complete and $\bigcup_{g\in G_p} g^- \neq T$.

The first two conditions correspond to the frequency requirement, while the third condition corresponds to the visit requirement. A path that satisfies any one of the above three conditions must violate the frequency or the visit requirement and therefore cannot exist in any feasible solutions.

2.1. Arc-flow model

Let P denote the set of infeasible paths. The binary decision variable $x_{i,j}$ $(i \in G_t, j \in G_{t+1}, t = 1, \dots, 4n-3)$ is set to one if an umpire travels from game i to game j, and zero otherwise. The arc-flow model can be formulated as follows:

$$\min \sum_{t=1,\dots,4n-3} \sum_{i\in G_t} \sum_{j\in G_{t+1}} d_{i^-,j^-} x_{i,j} \tag{1}$$

$$\text{s.t.} \sum_{j\in G_2} x_{i,j} = 1, \quad \forall\, i \in G_1, \tag{2}$$

$$\sum_{j\in G_{t+1}} x_{i,j} = \sum_{j\in G_{t-1}} x_{j,i} = 1, \quad \forall\, i \in G_t,\ t = 2, \dots, 4n-3, \tag{3}$$

$$\sum_{j\in G_{4n-3}} x_{j,i} = 1, \quad \forall\, i \in G_{4n-2}, \tag{4}$$

$$\sum_{(i,j)\in E_p} x_{i,j} \le |E_p| - 1, \quad \forall\, p \in P, \tag{5}$$

$$x_{i,j} \in \{0, 1\}, \quad \forall\, i \in G_t, j \in G_{t+1},\ t = 1, \dots, 4n-3. \tag{6}$$

Objective (1) minimizes the total travel distance of the umpires. Constraints (2) and (4) ensure that all of the games in the first and the last time slot have one umpire. Constraints (3) are the flow conservation constraints for the games from time slot 2 to time slot $4n-3$, and also ensure that these games have one umpire. Constraints (5) are the infeasible path elimination constraints, which ensure that all the paths for the umpires are feasible .

The infeasible path elimination constraints (5) are very weak in general, especially when the length of an infeasible path becomes large. We use a technique similar to that used by Kallehauge, Boland, and Madsen (2007) to strengthen the infeasible path elimination constraints. For an infeasible path $p = (g_1, \dots, g_m)^t$, let $\bar{E}_p = \{(g_s, i) \mid i \neq g_{s+1}, (g_1, \dots, g_s, i)^t$ be feasible, and $s = 1, \dots, m-1\}$ denote the set of arcs that can destroy the infeasible path p. Then the infeasible path constraints (5) can be re-formulated as follows:

$$\sum_{(i,j)\in \bar{E}_p} x_{i,j} \ge 1, \quad \forall\, p \in P. \tag{7}$$

The idea behind (7) is that once at least one arc in $\bar{E}_p$ is selected, the infeasible path p is destroyed. In the rest of the paper, we replace constraints (5) with constraints (7) as the infeasible path elimination constraints.