Analysis of a Constructive Matheuristic for the Traveling Umpire Problem

Reshma C. Chandrasekharan Túlio A. M. Toffolo Tony Wauters

Abstract

The Traveling Umpire Problem (TUP) is a combinatorial optimization problem concerning the assignment of umpires to the games of a fixed double round-robin tournament. The TUP draws inspiration from the real world multi-objective Major League Baseball (MLB) scheduling problem, but is, however, restricted to the single objective of minimizing total travel distance of the umpires. Several hard constraints are employed to enforce fairness when assigning umpires, making it a challenging optimization problem. The present work concerns a constructive matheuristic approach which focuses primarily on large benchmark instances. A decomposition-based approach is employed which sequentially solves Integer Programming (IP) formulations of the subproblems to arrive at a feasible solution for the entire problem. This constructive matheuristic efficiently generates feasible solutions and improves the best known solutions of large benchmark instances of 26, 28, 30 and 32 teams well within the benchmark time limit. In addition, the algorithm is capable of producing feasible solutions to various small and medium benchmark instances competitive with those produced by other heuristic algorithms. The paper also outlines experiments conducted to evaluate various algorithmic design parameters such as subproblem size, overlap and objective functions.

Keywords: TUP, decomposition, sports scheduling

1 Introduction

The Traveling Umpire Problem (TUP) is a sports scheduling problem introduced by Trick and Yildiz (2007) for modeling umpire scheduling in the American Major League Baseball (MLB). Much like any other optimization problem concerning game-scheduling, the practical importance of studying the TUP lies in the huge financial investments associated with satisfying the multiple objectives and logistic factors when scheduling games. A typical MLB tournament consists of 30 teams playing 2430 games over a period of 180 days. A total of 15 umpires officiate these games and each umpire requires traveling from one city to the other officiating about 130 games. Therefore it is highly desirable to optimize the travel distance in order to minimize the travel time and the costs involved. While umpire scheduling is a multi-objective problem, the TUP only concerns the primary objective of minimizing the total travel distance of the umpires. The TUP continues to represent a challenging optimization problem due to various hard constraints involved.

Problem definition

The MLB is scheduled as a double round-robin tournament of n umpires and $2n$ teams. Each team is associated with a home venue and any given pair of teams A and B play against each other exactly twice, once at team A's home venue and again at B's home venue. At each round every team plays exactly once, resulting in a total of $4n - 2$ tournament rounds. Given the schedule of such a round-robin tournament of $2n$ teams, the goal of the TUP is to assign n umpires to the tournament's games such that the total travel distance of the umpires is minimized. In addition, various constraints are enforced to ensure fairness concerning the assignment of umpires to consecutive teams or their venues as follows:

(a) Every game in the tournament is officiated by exactly one umpire

(b) Each umpire officiates exactly one game per round

(c) Each umpire must visit the home location of each team at least once

(d) An umpire cannot visit the same venue more than once in any $q1$ consecutive rounds

(e) An umpire cannot officiate the games of the same team more than once in any $q2$ consecutive rounds.

Parameters $q1$ and $q2$ are called tightness parameters and they range from 1 to n and 1 to $\lfloor \frac{n}{2} \rfloor$ respectively. Exceptions to this bounds are the instances (12,7,2), (14,8,3) and (14,8,2) presented as tuple $(2n, q_1, q_2)$.

Existing literature and current research

Trick and Yildiz (2007) utilized IP and CP formulations on TUP instances of 8 to 32 teams in their foundational research concerning the problem. These exact methods solved small instances of 8 to 10 teams to optimality. In the same paper, Trick and Yildiz present a constructive Greedy Matching Heuristic (GMH) which assigns umpires round by round to produce feasible solutions. The GMH produced optimal solutions for most small sized instances of 8 to 10 teams and feasible solutions for several medium sized instances of 12 to 16 teams. Since then various heuristic algorithms have been developed producing near-optimal solutions on various benchmark instances of 8 to 32 teams.

Trick et al. (2012) proposed a Simulated Annealing (SA) algorithm which improved the solutions produced by GMH. This method produced a feasible solution for the benchmark instance of 30 teams for the first time. In the same year, Trick and Yildiz (2012) introduced a Genetic Algorithm (GA) which produced optimal solutions for all instances of 8 to 10 teams and improved existing solutions for various instances of 12 to 30 teams. De Oliveira et al. (2014) proposed a Relax and Fix Heuristic (RFH) which solves a relaxed IP formulation of the full schedule where a window of variables are fixed progressively after each round until the last round is fixed. RFH improved best known solutions for various medium-sized (14-24 teams) instances, improved existing lower bounds and provided lower bounds for instances of more than 16 teams for the first time. Two heuristic algorithms, Enhanced iterative Deepening search with Leaf node Improvements (IDLI) and Iterated Local Search (ILS) algorithms were developed by Wauters et al. (2014) which improved the best solutions for small and medium-sized instances and utilized a new decomposition method to improve lower bounds for all benchmark instances. In addition, ILS was also able to produce high quality solutions for the large instances of 26 to 32 teams.

Aside these heuristic algorithms, there have been few exact algorithms which have proven useful in establishing the optimality or infeasibility of various instances. Toffolo et al. (2014) presents a branch and price algorithm which improved the solutions of five instances of 16-teams and proved the infeasibility of some others for the first time. Xue et al. (2015) introduced two other exact algorithms which were capable of solving two 14 teams-instances to optimality for the first time. Recently, Toffolo et al. (2016b) presented a branch and bound algorithm coupled with a decomposition method that produces strong lower bounds. This powerful algorithm is capable of producing optimal solutions for most instances of 8 to 16 teams with a runtime of few minutes.

Though TUP is considered a computationally hard optimization problem, the problem's complexity remains unknown for sure. de Oliveira et al. (2015) proved that the TUP is NP complete for certain parameter combinations. While various algorithms are capable of solving most small and medium sized instances to optimality, no efficient algorithm has been able to produce optimal solutions to the large benchmark instances comprising of 26, 28, 30 and 32 teams. The literature suggests that decomposition-based methods coupled with hybrid heuristic approaches produce better solutions for these instances. Various decomposition-based methods have been implemented where subproblems were solved by exact or heuristic methods. This paper employs a decomposition scheme based on constructive matheuristics (CMH) that improves the current best known solutions for the large instances within the benchmark time limit. This work draws motivation from and builds upon the decomposition strategies for the TUP presented in Toffolo (2017).

CMH and matheuristics more generally represent algorithmic hybrids of metaheuristic and mathematical programming algorithms. The exact and heuristic components may operate either hierarchically or in parallel exchanging information appropriately. Some of the first attempts in this direction are covered in the survey by Puchinger and Raidl (2005) while Doerner and Schmid (2010) and Archetti and Speranza (2014) surveys matheuristic methods for vehicle routing problems. Aside vehicle routing problems, matheuristic algorithms have been successfully applied in other problems such as the nurse rostering problem (Della Croce and

Salassa, 2014), permutation flow shop scheduling (Della Croce et al., 2011) and multidimensional knapsack problem (Hanafi et al., 2010). CMH in particular has been applied in problems such as shift minimization task scheduling problem (Smet et al., 2014), resource constrained project scheduling (Toffolo et al., 2016a) and nurse rostering problems (Santos, 2016).

CMH follows a constructive strategy where rounds of the game are grouped together to form subproblems which are solved using exact techniques. The optimal solutions of these subproblems are subsequently utilized to construct a solution for the entire problem. The construction strategy and challenges associated are explained in detail in the following section and section 3. Like other constructive heuristic techniques, CMH also requires tuning its parameters such that the solution so constructed using the optimal solutions from the subproblems results in a feasible solution for the entire problem. Constructive techniques in the literature for TUP adapt different techniques to navigate constructive heuristic to produce feasible solutions such as penalty functions and backtracking strategies. Algorithm design parameters of CMH and their effect on the navigation of CMH is also covered in section 3. The SA algorithm presented in Trick et al. (2012) was the first method capable of producing a feasible solution to a large instance was tested for 5 hours and since then, the benchmark time limit for a large TUP instance has been considered to be 5 hours. The computational experiments testing these parameters and the corresponding analysis is discussed in section 4 while conclusions are summarized in section 5.

2 The IP formulation

This paper utilizes the flow formulation of the TUP proposed by Toffolo et al. (2016b). Games are represented as nodes in a graph $G = (V, E)$ and directed edges connect games of round r to those of round $r + 1$. Nodes contain information regarding the game, home team and away team. The distance of a directed edge is the distance between the home venues of the two games. An example is illustrated in Figure 1. Before being utilized by the CMH, this graph is first processed by removing infeasible edges. Edges between nodes which contain same teams are deleted as they contradict constraint (iv) and (v) for $q_1 > 1$ and $q_2 > 1$. Moreover, umpires are fixed to the games of the first round to break the inherent symmetry (Yildiz, 2008) concerning the assignment of umpires.

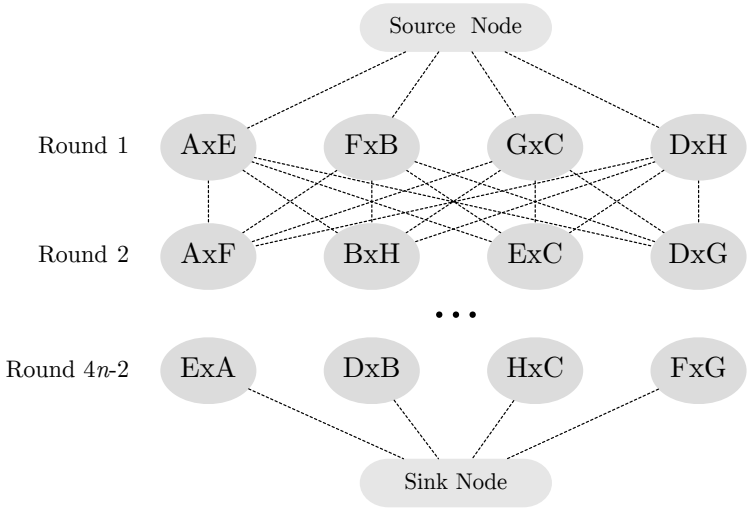

Fig. 1. $G = (V, E)$

The formulation requires the following input:

d_e : distance of directed edge e
I : set of teams $\{1, ..., 2n\}$
H_i : set of nodes where team i plays at home
R : set of rounds $\{1, ..., 4n-2\}$
$Q_{ir}^{'}$: set of nodes of team i playing at home in rounds $R \cap \{r, ..., r+q_1-1\}$
$Q_{ir}^{''}$: set of nodes of team i playing at home in rounds $R \cap \{r, ..., r+q_2-1\}$
U : set of umpires $\{1, ..., n\}$

The set of edges that enter node I is given by $\delta(I)$ and the set of nodes that exit node I is denoted by $\omega(I)$. The decision variables are:

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

The problem's formulation is given below:

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

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

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

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

$$\sum_{e \in \delta(Q_{ir}^{'})} X_{eu} \leq 1 \quad \forall i \in I, \forall r \in R, \forall u \in U \tag{5}$$

$$\sum_{e \in \delta(Q_{ir}^{''})} X_{eu} \leq 1 \quad \forall i \in I, \forall r \in R, \forall u \in U \tag{6}$$

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

Eq. (1) denotes the objective function which minimizes the total travel distance of the umpires. Constraint (2) restricts the assignment of multiple umpires to a single game while Constraint (3) denotes the flow preservation constraints concerning the proposed flow formulation. These constraints enforce that an umpire officiates a game in round r and that the umpire proceeds to the round $r+1$. Constraint (4) ensures that every umpire visits each home location at least once. Eq. (5) and Eq. (6) represents the tightness constraints while Eq. (7) states that the decision variables assumes binary values.

3 Constructive Matheuristic

The constructive matheuristic (CMH) follows a decomposition-based approach by which the entire problem is divided into subproblems called blocks. The goal of CMH is to solve these subproblems to optimality and utilize these optimal blocks to construct a solution for the entire problem. This requires fine tuning the CMH design parameters such that solutions of previously solved blocks do not prevent the feasibility of the blocks yet to be solved and that merging subproblem-solutions remains feasible in terms of the entire problem. In addition, the problem may comprise of constraints which may only be evaluated on the full problem. Thus, the CMH strategy must also ensure the feasibility of constraints if any which may not be evaluated locally on individual subproblems.

In this particular problem, consecutive rounds are grouped to form subproblems or blocks. Each block is identified by the first round of the block, denoted by b. The CMH solves an IP formulation of the problem restricted to each block. However, only a relaxed IP formulation obtained by removing constraint (4) is solved on the blocks as constraint (4) may only be evaluated on a complete schedule. The CMH starts execution from the first block containing the source node, solves each block to optimality and continues solving until either the final block is reached or when a block proves infeasible. As CMH proceeds to the next block, variables in the previous block are fixed. The number of unvisited home locations for each umpire must return to zero in the final round to ensure that the solution produced by CMH is feasible. Design parameters such as block size (η), weight (w), overlap (θ) and the number of relaxed future rows (ρ) are utilized to navigate CMH to produce feasible solutions for the TUP. This is accomplished by the block objective function, $Z(b, f, w)$ defined in the following section.

Block objective function ($Z(b, f, w)$)

The primary aim of the block objective function $Z(b, f, w)$ is to minimize the total distance traveled by umpires while traveling within the rounds of the concerned block. $Z(b, f, w)$ aims at navigating the CMH to produce solutions which do not violate constraint (4) as well. This is accomplished by employing an incentive term in F. Whenever an umpire is assigned to a venue it has not visited before, an incentive is assigned to the block objective subject to a weight in terms of the block.
The block objective function is defined as follows:

$$Z(b, f, w) = \sum_{u \in U} \sum_{e \in E_b} (X_{ue} d_e - f(b) w X_{ue} y_{ue})$$

where,

$$E_b : \text{Edges in block } b \tag{8}$$

$$y_{ue} = \begin{cases} 1 & \text{if edge } e \text{ starts at a node which has not been visited by umpire } u \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\tag{10}$$

After solving block b, the CMH counts the number of home venues each umpire is left to visit and stores their sum in the variable k_b. The goal is to navigate CMH to produce solutions for which the value of k_b at the last block, in other words, the violations to constraint (4) is zero. Function $f(b)$ is problem specific and must be selected such that k_b goes to zero in the last round. In this problem, $f(b)$ is defined to be an increasing function of the blocks so that the incentive increases as CMH proceeds to the last block and that the incentive's rate of increase is just enough to let any violations to constraint (4) disappear when CMH finishes executing the final block. Not only that $f(b)$ significantly affects the trend in the values attained by k_b as CMH proceeds through the blocks, the choice of $f(b)$ also affect the actual values attained by the travel distance of the umpires within the block. Various preliminary experiments were conducted and the conclusions are summarized in Section 4. For this problem, $f(b) = \sqrt{b}$.

Weight (w)

Once function $f(b)$ is selected for the problem, fine-tuning of the incentive term is performed by introducing an additional design parameter weight (w). The value of w determines the exact weight of the incentive term in the objective function. Once multiple design parameters are employed, the value of weight (w) helps in fine-tuning CMH to produce feasible solutions for the TUP such that the objective function continues to focus on minimizing the distance traveled by umpires. The CMH also enables the flexibility of having different weights for different blocks based on the design parameters explained throughout the following sections.

Block size (η)

Block size η corresponds to the subproblem-size. While a smaller block size offers less flexibility while assigning umpires to blocks, a larger block size may help prevent future blocks being infeasible and may

also improve the final solution-quality. However, a large value of η also implies that CMH must solve more difficult subproblems, resulting in longer runtimes. Therefore choosing an appropriate value of η is central to producing high quality solutions within the benchmark time limits.

Overlap (θ)

Due to algorithm's constructive nature, solution of each block significantly affects the feasibility of future blocks and in turn the feasibility of solution for the entire problem. Overlap is possible between consecutive blocks and overlap between blocks allow the CMH to accommodate the quality and feasibility of the next block while solving the current block. Thus the objective function is not anymore a function only of the current block (b), but also of the rounds of the next block overlapped with those of the current block. Parameter overlap (θ) denotes the number of last rows of a block which overlap with the rows of the next block. Increasing the value of θ increases the number of subproblems CMH solves thereby resulting in a larger computation time. Overlap between subproblems is also expected to result in better continuity in properties and effect of design parameters between successive blocks. Thus implementing overlap between rows is expected to improve the predictability of CMH's behavior.

Relaxed future rows (ρ)

In order to solve larger block size without incurring increased computational times, few succeeding rows are attached to blocks where within these rows, the integrality constraints of the variables is relaxed. Relaxed future rows refers to rows so added and the number of such relaxed future rows added to the blocks is denoted by ρ. The influence of the value of ρ on the solution-quality and runtime are tested. The relaxed future rows is expected to produce better solutions without increasing the runtime as much as when the bock size is increased from η to $\eta + \rho$. The impact of adding relaxed rows is compared and contrasted with the effect of increasing block size.

Figure 2 illustrates propagation of CMH blocks on a given schedule. The rectangular window represents the current block and nodes with dashed borders correspond to relaxed future rounds. For this example, $\eta = 4$, $\theta = 1$ and $\rho = 2$.

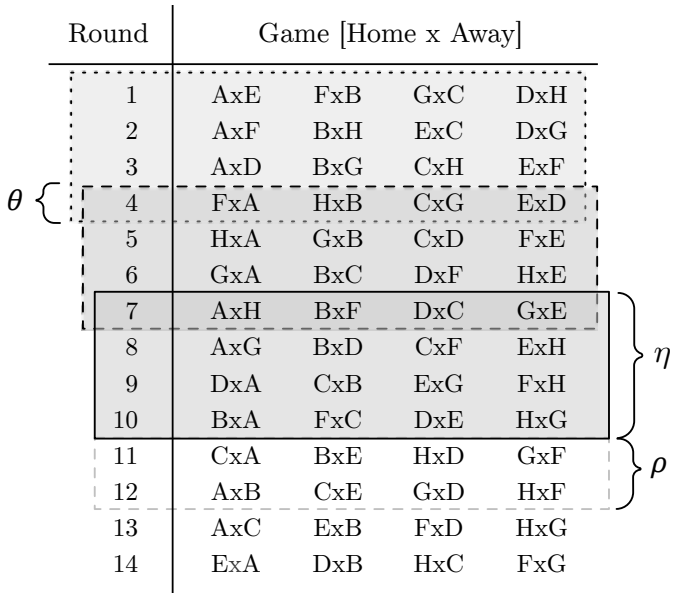

Fig. 2. CMH blocks for TUP

4 Computational Experiments

Extensive experiments were performed and the CMH algorithm was tested on the TUP benchmark instances of 8-32 teams for a range of parameter combinations. All these instances and their existing best known solutions are available at http://benchmark.gent.cs.kuleuven.be/tup. Experiments to further study the effect of design parameters such as block objective function, weight, block size, overlap and relaxed future rows were conducted focusing on the large benchmark instances of 26 to 32 teams for $q_1 = q_2 = 5$. The goal is to produce high quality solutions within the benchmark time limit of 5 hours. The CMH algorithm is coded in Python and it uses Gurobi Version 7.0.1 to solve the blocks. Experiments were performed on four threads of an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz computer running Ubuntu 16.04.2 LTS.

4.1 Design parameters

This section summarizes experiments conducted to test the effect of the design parameters block objective function, weight, block size, overlap and relaxed future rounds.

Block objective function ($Z(b, f, w)$)

Preliminary experiments were conducted to arrive at an appropriate objective function for the block and its component $f(b)$ in particular. $f(b)$ is chosen to be an increasing function of the blocks such that there is a higher incentive to prevent violations to constraint (iii) when CMH executes last blocks. Various objective functions were tested and those candidates utilized in this section to summarize the conclusions efficiently are listed below:

$$Z_{dist} = \sum_{u \in U} \sum_{e \in E_b} X_{ue} d_e$$

$$Z_{inc} = -\sum_{u \in U} \sum_{e \in E_b} w X_{ue} y_{ue}$$

$$Z_{lin} = \sum_{u \in U} \sum_{e \in E_b} (X_{ue} d_e - bw X_{ue} y_{ue})$$

$$Z_{sqrt} = \sum_{u \in U} \sum_{e \in E_b} (X_{ue} d_e - \sqrt{b} w X_{ue} y_{ue})$$

$$Z_{cbrt} = \sum_{u \in U} \sum_{e \in E_b} (X_{ue} d_e - \sqrt[3]{b} w X_{ue} y_{ue})$$

Figures 3, 4 and 5 present graphs those explain the conclusions using a sample instance of 30 teams where the parameters $\eta = 5, w = 50, \theta = 4$ and $\rho = 0$. Similar conclusions were obtained for other instances as well. Figure 5 show how different objective functions influence the total objective and incentive terms and Figures 3 and 4 demonstrate how these factors eventually manifest in terms of the total distance traveled by the umpires and violations to constraint (4), denoted by k.

Figure 3 plots k_b, the number of home locations yet to be visited by umpires, against the blocks represented by its first round b. The graph demonstrates the effect of objective function in determining the trend in violations to constraint (4). From the values of k_b attained by different graphs at the last round, otherwise called k, it is clear that an incentive term is necessary in the objective function in order to obtain feasible solutions. These graphs not only imply that an incentive term is necessary, but also that the value of the incentive term at each block influences the rate at which k_b decrease. For Z_{cbrt}, $k \neq 0$ while for Z_{sqrt}, k goes to 0. Z_{inc} corresponds to the case when the objective function only minimizes the violations to constraint (4) and does not have a term for minimizing the distance. This graph serves as a good estimate to study how fast k_b may vanish.

The total distance traveled by umpires on each block is plotted in Figure 4. It is evident from these graphs that lower incentive values leads to shorter travel distance. This observation is supported by the the total travel distance obtained by CMH for the full schedule using different objective functions $Z_{dist}, Z_{cbrt}, Z_{sqrt}, Z_{lin}$ and Z_{inc} which amount to be 435580, 439473, 446004, 467713 and 805008 miles respectively. This observation can be explained utilizing Figure 5.

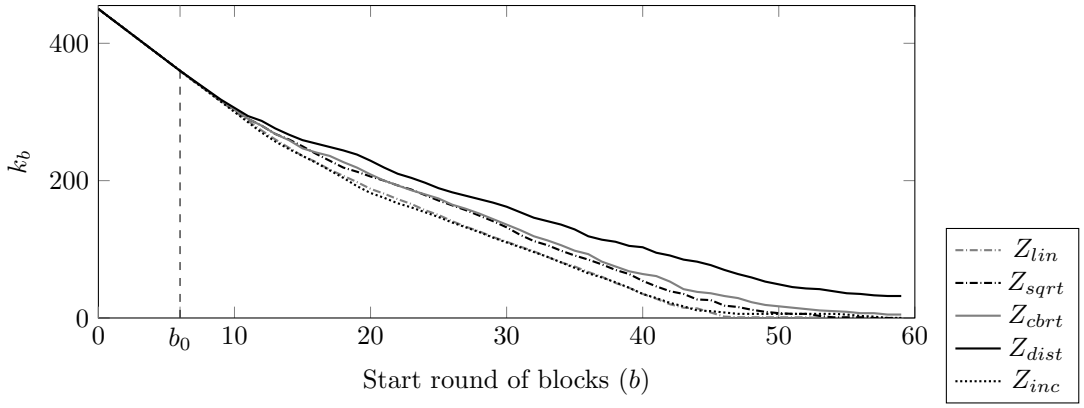

Fig. 3. Trend in violations for different block objective functions

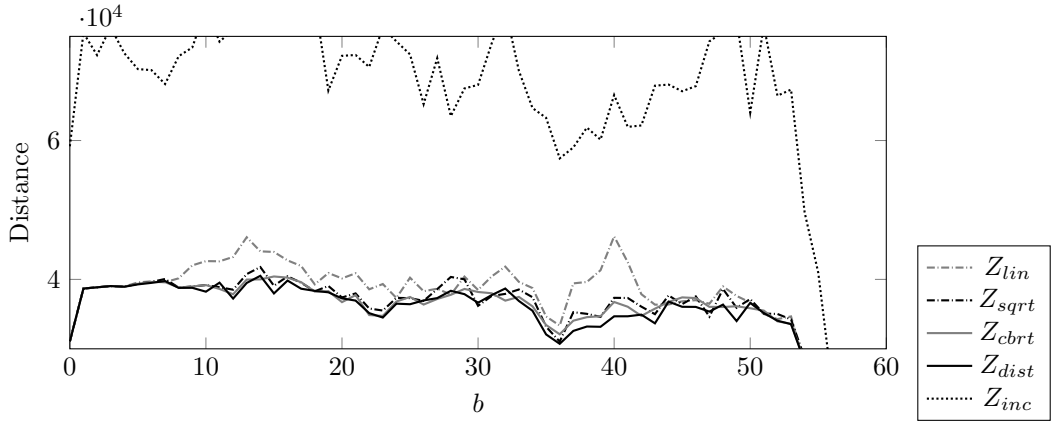

Fig. 4. Total travel distance of the umpires obtained for various objective functions

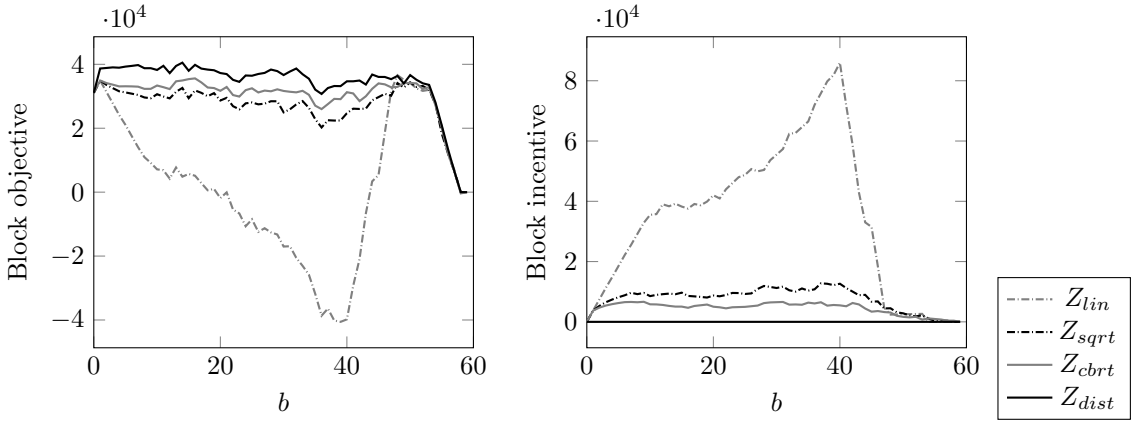

Fig. 5. Trend in block objective and block incentive values for different block objective functions

Figure 5 compares the objective and incentive values attained by different objective functions at each block. These graphs indicate that the value of objective is lower for a choice of $f(b)$ that leads to a larger incentive. But the trend in the incentive values imply that the lowering of objective does not occur due to the effective minimization of distance, but from the large contribution of the incentive term. As Figure 3 indicates, a higher value of incentive makes the violations fall faster, but however, note that CMH only requires that the violations go to zero when it executes the last block. Therefore, an exceedingly high value of incentive prevents CMH from minimizing the actual travel distance of the umpires. So, the goal of the preliminary experiments is to choose an appropriate objective function such that the choice of $f(b)$ leads to

an incentive contribution which is small enough to let CMH minimize the total travel distance of the umpires effectively and just big enough to bring the violations to zero after the last block is solved.

Different functions were tested to arrive at the best choice of $f(b)$. Out of the five functions included in the discussion Z_{sqrt} produces the feasible solution with smallest total travel distance by the umpires. Z_{cbrt} leads to even smaller total distance traveled by the umpires, but leads to $k = 5$. Various functions such as e^b and $log(b)$ which leads to larger incentives than that contributed by Z_{sqrt} or lower than that contributed by Z_{cbrt} are not included in the discussion as they either do not lead to feasible solutions or are not as efficient as Z_{sqrt} in optimizing the distance. Choices of $f(b)$ such as b^x, $\frac{1}{2} < x < \frac{1}{3}$ and other linear combinations of $b^{\frac{1}{2}}$ and $b^{\frac{1}{3}}$ which leads to incentives larger than that contributed by Z_{cbrt} and smaller than that contributed by Z_{sqrt} were also tested. Some of them gave better solutions than those produced by Z_{sqrt} for some parameter combinations. However, for simplicity the objective function was chosen to be Z_{sqrt} which produced high quality solutions consistently over a wide range of parameter combinations. For the rest of the paper, objective function refers to Z_{sqrt}.

***Weight** (w)*

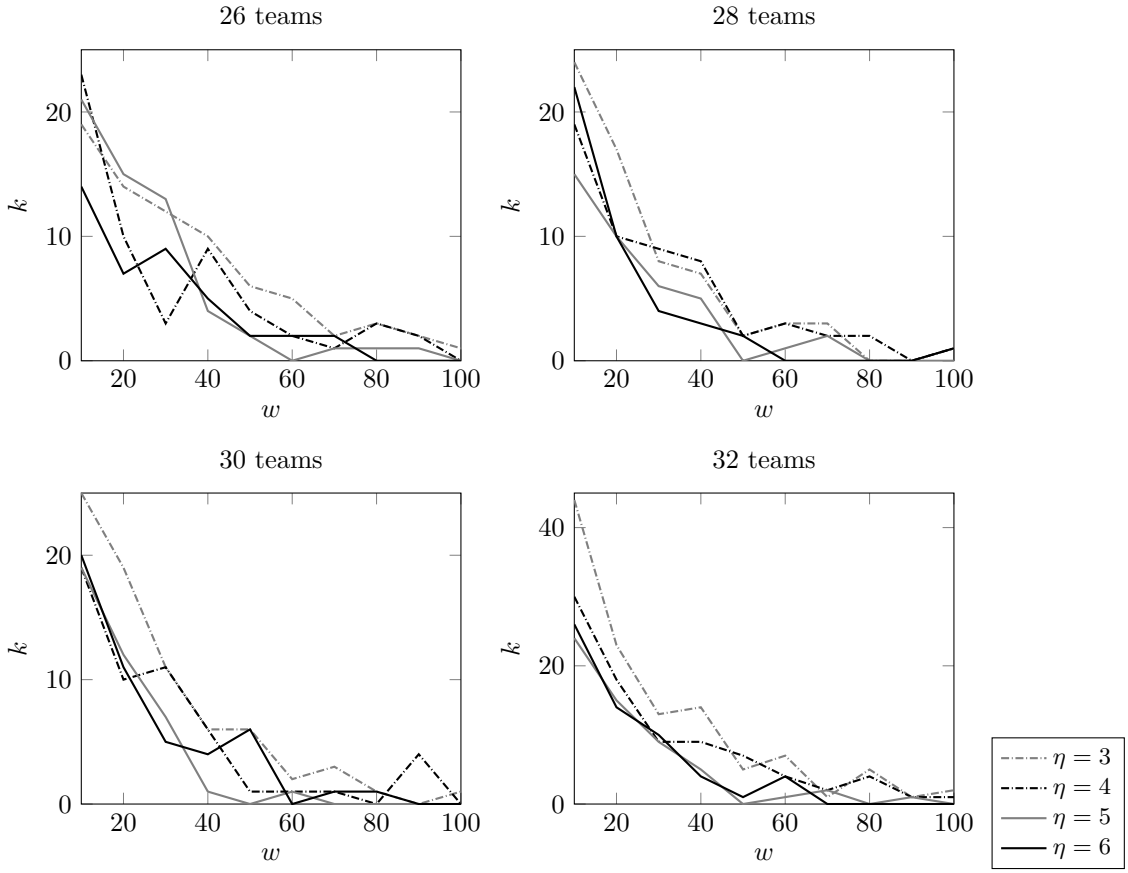

Fig. 6. Trend in violations (k) with weight (w)

Figure 6 plots violations (k) against weight (w) for various block sizes (η) on the four large instances for $\theta = 0$ and $\rho = 0$. The parameter w which is introduced to fine tune the incentive term is allowed to take values in $\{10, 20, \ldots, 100\}$. As expected, the figure indicates that increasing w decreases the violations to constraint (4), denoted by k. But higher values of w shifts the focus of optimization to the minimization of violations and leads to higher values of total distance traveled by umpires. This is illustrated by Figure 7 which plots the value of the total travel distance of umpires against w for the large instances, again for $\theta = 0$ and $\rho = 0$. Therefore for given values of ρ, θ and η the smallest w for which $k = 0$ is expected to produce the best feasible solution.

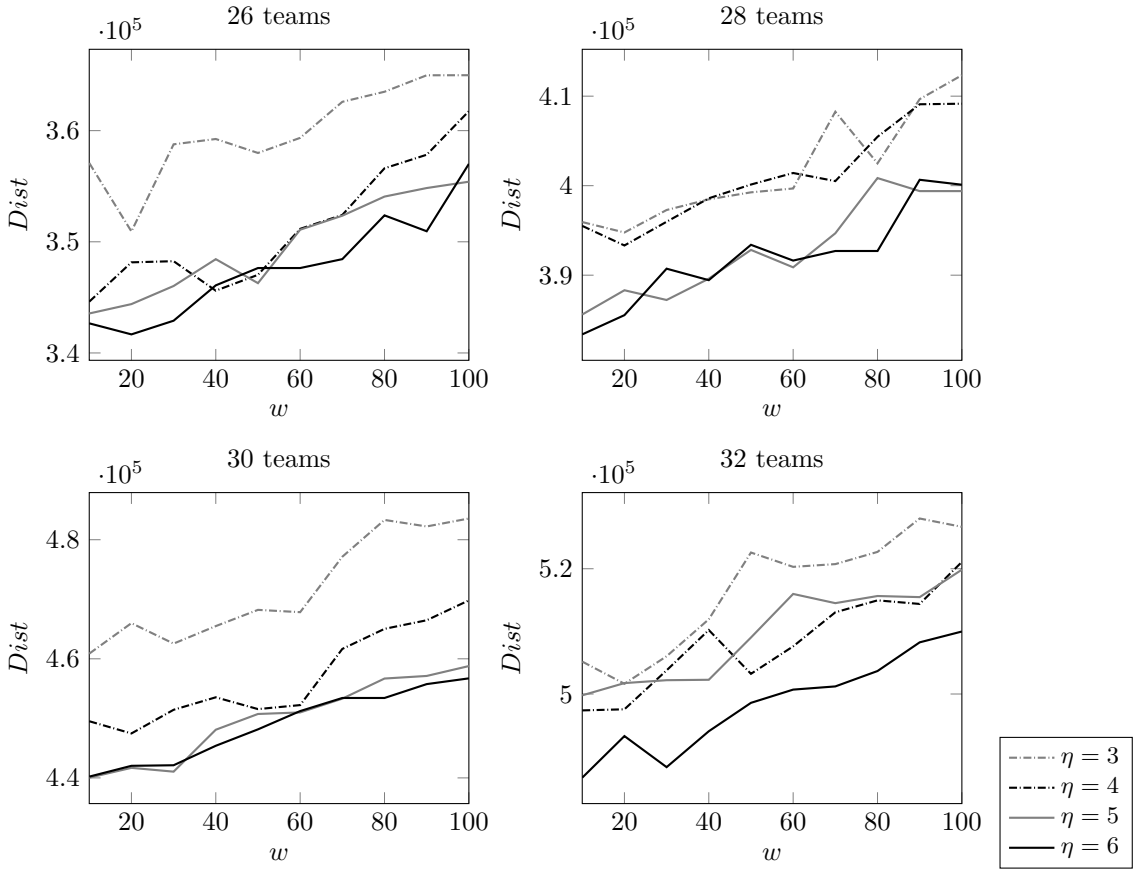

Fig. 7. Total travel distance for different block size

Block size (η)

Graphs presented in figure 7 illustrates also that larger block sizes lead to better solutions. Figure 8 plots the solve time for block sizes for the large instances and these graphs demonstrate that larger block sizes require longer solve time. Therefore, the the parameter η is allowed to take values in in $\{3, 4, 5, 6\}$ so that the the run time lies within the bounds of benchmark time limits. Experiments for $\eta = 6$ is not conducted for 32-teams as runtime exceeds the benchmark time limit.

Overlap (θ)

Experiments were conducted to test the effect of overlap on the quality of the solutions. The contrast in terms of the final solution quality with zero and maximum overlap are illustrated in Figure 9. Distances with and without overlap are plotted against weight(w) and the graphs indicate that in general, overlap leads to considerable improvement in the final solution quality. Figure 9 also plots the trend in violations with respect to increasing values of w. The figure illustrates that if $k = 0$ for a given value of weight w', the value of k continues to be 0 for any value of weight $w > w'$. This is a property that emerges when the blocks are overlapped for high values of η and was absent in experiments without overlap (see Figure 6 for comparison). This property enables predicting the feasibility of the solution produced by CMH for a given value of w when blocks are overlapped.

Relaxed future rows (ρ)

Adding relaxed future rows is generally seen to improve the solution quality for almost all instances. In addition, it is observed that on adding ρ' future rows to blocks of size η', CMH generates solutions which are comparable to those produced for block size $\eta' + \rho'$ without significantly increasing the solve time. This

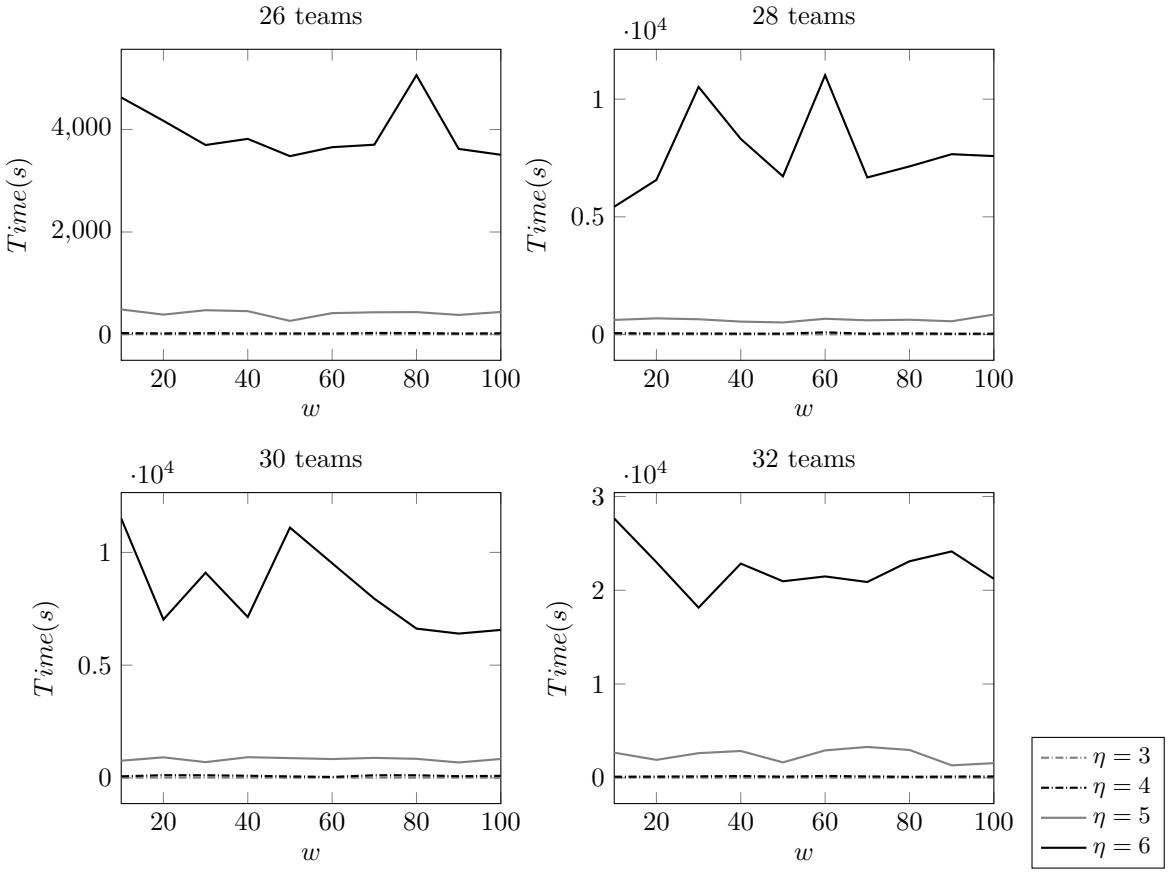

Fig. 8. Solve time for different block sizes

is illustrated by an example in Figure 10. Figure 10(a) compares the total travel distance for the 30-teams instance without overlap for block size 5 and 6 against that generated when a relaxed future row is added to block size 5. The total run time of the corresponding cases are compared in Figure 10(b). This property has proven useful especially for improving the solution quality of the 32-teams instance. While CMH fails to generate feasible solutions for $\eta = 6$ within benchmark time limits, relaxed rows were added to smaller block sizes to produce better results in shorter solve time.

The best feasible solution obtained for a parameter combination of ρ, η and θ for w in $\{10, \ldots, 100\}$ is summarized in Table 1. Only those solutions which are produced within the benchmark time limits are presented.

4.2 Improving the CMH strategy

Information from the trend in violations when the functions Z_{dist} and Z_{inc} are utilized by CMH is employed to improve the CMH strategy. Figure 3 demonstrates that till the block starting with round b_0, the graphs of Z_{dist} and Z_{inc} remains overlapped, suggesting that for these blocks, the incentive term does not significantly impact the trend in the violations. This information is utilized to remove the incentive term in these blocks to improve the solutions produced by CMH. For all blocks starting at a row smaller than b_0, w is set to 0. Experiments conducted suggest that the effect is the most significant when the blocks are maximum overlapped. This strategy improves the solutions of three out of the four large instances.

Table 2 lists the best feasible solutions produced by CMH for each large instance within the benchmark time limits. These solutions are compared with the solutions produced by ILS (Wauters et al., 2014) and with the current best known solutions. The corresponding parameter configurations and the percentage gap between the current best known lower bounds are also presented. $*$ indicates that the solution was obtained

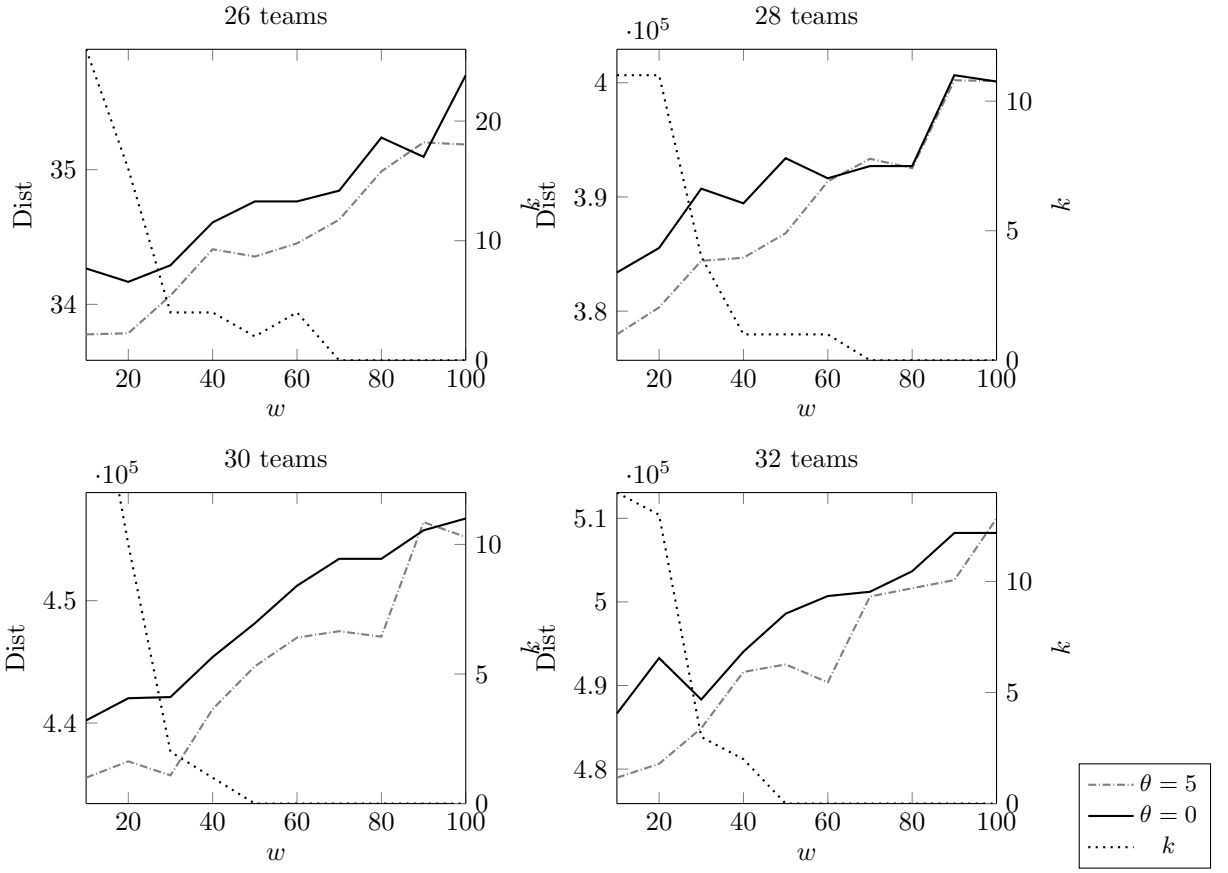

Fig. 9. Effect of overlap in the total travel distance and violations

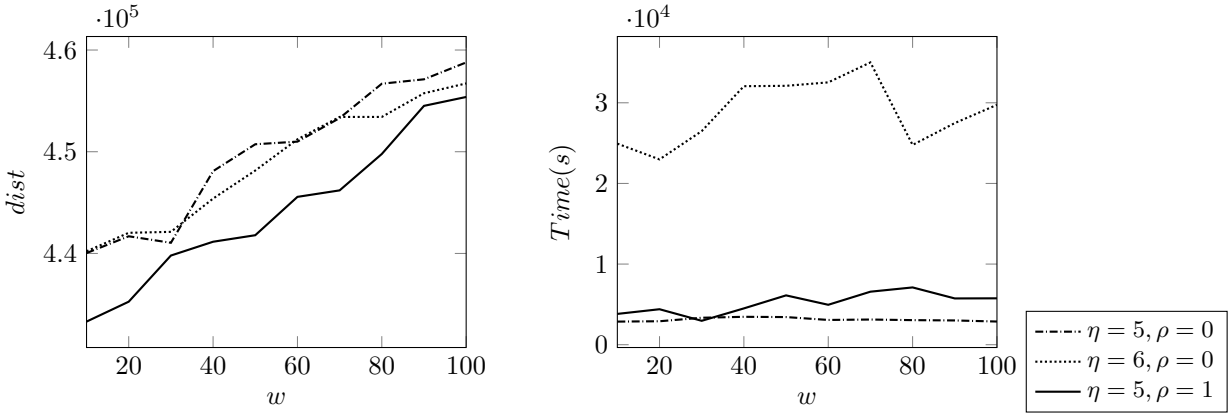

Fig. 10. Effect of adding relaxed rows in the total travel distance and solve time

using the improved strategy.

4.3 Parameter tuning

Since the CMH algorithm has various parameters, the package irace 2.3 was utilized to train the parameters. The best parameter configuration obtained by irace is given by $(\eta, w, \theta, \rho) = (6, 60, 3, 2)$. The solutions corresponding to this configuration for all large instances are presented in Table 3. The percentage gap (% Gap) between these solutions found by irace and those produced by CMH also presented.

Table 1: Summary of results obtained by CMH for large instances

			26		28		30		32	
ρ	η	θ	Dist.	Time(s)	Dist.	Time(s)	Dist.	Time(s)	Dist.	Time(s)
0	3	2	365,060	12.74	400,684	18.30	453,811	23.61	511,680	41.05
		1	367,235	8.19	402,251	6.83	460,291	10.75	514,968	18.09
		0			402,495	6.11	482,228	10.78		
	4	3	350,962	92.65	405,101	210.81	451,806	217.41	500,794	414.17
		2			400,628	74.62	453,860	107.65	509,265	179.25
		1	356,227	25.34	392,226	81.16	453,974	78.85	509,224	154.99
		0			409,095	33.92	465,031	115.71		
	5	4	346,687	1,282.51	386,528	2,112.46	443,632	3,460.04	494,806	5,886.88
		3	350,245	619.74	390,317	1,109.28	443,772	1,550.46	499,305	3,067.33
		2	347,884	476.87	392,679	816.53	455,969	1,247.06	495,338	2,052.81
		1	352,405	542.40	397,170	688.72	452,347	936.36	506,774	1,608.71
		0	351,087	417.90	392,822	512.96	450,734	877.72	509,021	1,644.47
	6	5	346,285	13,431.05	392,520	21,894.63	444,635	32,104.04		
		4	345,340	7,264.73	390,933	14,173.14	441,931	13,335.99		
		3	350,468	6,887.34	390,887	8,891.55	444,471	12,728.84		
		2	350,992	3,238.62	389,837	9,121.47	444,974	8,227.36		
		1	349,659	2,365.69	394,584	10,246.09	445,562	7,423.22		
		0	350,948	3,620.97	391,631	11,017.39	451,221	9,522.13		
1	2	1	365,060	16.40	400,684	22.74	463,096	29.78	519,169	40.82
		0	369,112	9.62	407,126	10.52	466,944	16.57	514,433	22.20

Table 2: Best results obtained by CMH for the large instances

$2n$	(η, w, θ, ρ)	CMH	Time(s)	ILS	Best known	Gap (%)
26	$(4, 50, 3, 2)$*	344,110	3,405.52	354,134	351,932	5.96
28	$(4, 50, 3, 2)$*	388,129	6,258.46	398,101	390,635	6.76
30	$(4, 40, 0, 2)$	440,696	2,660.46	451,917	443,739	6
32	$(4, 40, 3, 2)$*	490,380	17,168.5	502,890	491,075	7.38

Table 3: irace-results for the large instances

$2n$	irace	CMHbest	% Gap
26	350,468	345,340	1.46
28	390,919	389,077	0.47
30	448,055	440,696	1.64
32	497,030	491,091	1.19

4.4 Performance of CMH on small and medium sized instances

Though the CMH was aimed at generating feasible solutions to the large instances, it is worth mentioning that the proposed algorithm have been able to produce feasible solutions comparable to those produced by other heuristic algorithms and are close to optimality in almost all instances of 8 to 22 teams. Table 4 reports best results and the solve time of the experiments on these small and medium sized instances. The existing best known solution and percentage gap of the solutions produced by CMH with respect to that of the existing best known solutions are also reported. Note that the CMH has also been able to improve solutions for two of the medium sized instances 16-8-3 and 18-7-4 as indicated by the negative values of gap. The optimal solutions and the solutions those improve the existing best known solutions are highlighted in the table.

Table 4: Result for small and medium sized instances

$2n$	q_1	q_2	CMH	Time(s)	Best known	% Gap
12	7	2	86,889	4.4	86,889	0
12	5	3	93,679	2.28	93,679	0
12	4	3	90,148	0.62	89,826	0.36
14	8	3	181,232	12.79	172,177	5.26
14	8	2	149,017	7.31	147,824	0.81
14	7	3	164,440	94.36	164,440	0
14	7	2	146,980	18.55	146,656	0.22
14	6	3	159,507	312.18	158,875	0.4
14	6	2	145,280	6.32	145,124	0.11
14	5	3	155,346	75.38	154,962	0.25
14	5	2	143,548	6.98	143,357	0.13
14A	8	3	170,026	34.28	166,184	2.31
14A	8	2	144,165	27.47	143,043	0.78
14A	7	3	160,830	137.76	158,760	1.3
14A	7	2	141,673	19.55	140,562	0.79
14A	6	3	154,402	13.05	152,981	0.93
14A	6	2	139,775	2.82	138,927	0.61
14A	5	3	149,751	59.26	149,331	0.28
14A	5	2	137,853	5.7	137,853	0
14B	8	3	168,107	6.14	165,026	1.87
14B	8	2	141,628	72.47	141,312	0.22
14B	7	3	157,884	152.31	157,884	0
14B	7	2	139,208	2.28	138,998	0.15
14B	6	3	154,275	28.65	152,740	1
14B	6	2	138,297	18.61	138,241	0.04
14B	5	3	151,015	30.14	149,455	1.04
14B	5	2	136,245	18	136,069	0.13
14C	8	3	178,310	26.04	161,262	10.57
14C	8	2	141,998	14.53	141,015	0.7
14C	7	3	159,812	85.21	154,913	3.16
14C	7	2	138,928	1.64	138,832	0.07
14C	6	3	153,541	9.41	150,858	1.78
14C	6	2	136,913	2.13	136,394	0.38
14C	5	3	148,977	24.81	148,349	0.42
14C	5	2	135,302	2.39	134,916	0.29
16	8	2	166,939	161.84	161,999	3.05
16	7	3	165,765	324.85	165,765	0
16	7	2	152,495	186.83	150,433	1.37
16A	8	2	174,044	184.36	171,882	1.26
16A	7	3	180,641	311.65	178,511	1.19
16A	7	2	165,254	202.52	163,709	0.94
16B	7	3	184,636	6,003.46	180,204	2.46
16B	7	2	169,075	1,324.8	167,190	1.13
16C	8	2	186,437	4.32	179,939	3.61
16C	7	3	183,429	3,523.82	180,483	1.63
16C	7	2	169,466	145.68	166,479	1.79
18	8	3	202,662	1,607.16	248,302	−18.38
18	7	4	215,129	212.71	217,502	−1.09

5 Conclusion

This paper introduces a constructive matheuristic algorithm (CMH) for solving the Traveling Umpire Problem which concerns the assignment of umpires to the games of a tournament subject to multiple hard constraints. The proposed algorithm focuses on finding solutions for the large benchmark instances. Various algorithmic design parameters such as objective function, subproblem size, subproblem overlap and relaxed subproblems were considered and extensive experiments were conducted. The CMH algorithm improves existing solutions for these large instances.

Experiments conducted to test subproblems size demonstrate that larger subproblems result in better solutions. With regard to the effect of overlapping subproblems, experiments indicate that it not only increases the possibility of a feasible solution, but also adds predictability in CMH's behavior. Adding relaxed future rows enable larger subproblems to be solved in a shorter time, improving the quality of the solutions within the benchmark time limit. Therefore, design parameters not only affect the feasibility but also the solution quality and algorithm runtime.

The proposed algorithm also improved the solutions of two medium sized instances of 18 teams and produced feasible solutions to almost all other instances in short run times. Moreover, the constructive matheuristic solutions improve solutions of other heuristic algorithms in almost all instances and this indicates the possibility of matheuristics constituting the state of the art of various other optimization problems.

Parameter tuning experiments were conducted to arrive at the best combination of design parameters for each instance. An interesting future research topic would be to further study the inter-operation of design parameters and design a quick start algorithm which outputs the best configuration for a given instance. Furthermore, the algorithm can utilize other formulations of the same problem to investigate possible improvements. A more general ambition constitutes the development of a constructive matheuristic framework which may be applied to a wider range of optimization problems.

6 Acknowledgements

References

Archetti, C. and M. G. Speranza
2014. A survey on matheuristics for routing problems. *Computers and Operations Research.*

De Oliveira, L., C. C. De Souza, and T. Yunes
2014. Improved bounds for the traveling umpire problem: A stronger formulation and a relax-and-fix heuristic. *European Journal of Operational Research.*

de Oliveira, L., C. C. de Souza, and T. Yunes
2015. On the complexity of the traveling umpire problem. *Theoretical Computer Science.*

Della Croce, F., A. Grosso, and F. Salassa
2011. A matheuristic approach for the total completion time two-machines permutation flow shop problem. *Evolutionary computation in combinatorial optimization, Lecture notes in computer Science, Berlin, Heidelberg: Springer*, 6622.

Della Croce, F. and F. Salassa
2014. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research.*

Doerner, K. F. and V. Schmid
2010. Survey: Matheuristics for rich vehicle routing problems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).*

Hanafi, S., J. Lazi??, N. Mladenovi??, C. Wilbaut, and I. Cr??vits
2010. New hybrid matheuristics for solving the multidimensional knapsack problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).*

Puchinger, J. and R. Raidl
2005. Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization : A Survey and Classification. *Artificial Intelligence and Knowledge Engineering Applications a Bioinspired Approach.*

Santos, H. G., T. T. A. M. G. R. A. M. . R. S.
2016. Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1):225–251.

Smet, P., T. Wauters, M. Mihaylov, and G. Vanden Berghe
2014. The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega (United Kingdom).*

Toffolo, T. A. M.
2017. Decomposition-based algorithms for optimization problems. *PhD thesis, KU Leuven, Belgium.*

Toffolo, T. A. M., H. G. Santos, M. A. M. Carvalho, and J. A. Soares
2016a. An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *J. Scheduling*, 19:295–307.

Toffolo, T. A. M., T. Wauters, S. Van Malderen, and G. Vanden Berghe
2014. Branch-and-price and improved bounds to the traveling umpire problem. *In Proceedings of the 10th international conference on practice and theory of automated timetabling, (PATAT, August 2014), York, UK.*

Toffolo, T. A. M., T. Wauters, S. Van Malderen, and G. Vanden Berghe
2016b. Branch-and-bound with decomposition-based lower bounds for the Traveling Umpire Problem. *European Journal of Operational Research.*

Trick, M. A. and H. Yildiz
2007. Bender's cuts guided large neighborhood search for the traveling umpire problem. *P. V. Hentenryck & L. Wolsey (Eds.), Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture notes in computer science, Berlin Heidelberg: Springer*, 4510.

Trick, M. A. and H. Yildiz
2012. Locally optimized crossover for the traveling umpire problem. *European Journal of Operational Research.*

Trick, M. A., H. Yildiz, and T. Yunes
2012. Scheduling major league baseball umpires and the traveling umpire problem. *Interfaces.*

Wauters, T., S. Van Malderen, and G. Vanden Berghe
2014. Decomposition and local search based methods for the traveling umpire problem. *European Journal of Operational Research.*

Xue, L., Z. Luo, and A. Lim
2015. Two exact algorithms for the traveling umpire problem. *European Journal of Operational Research.*

Yildiz, H.
2008. Methodologies and applications of scheduling, routing and related problems. *PhD thesis, Carnegie Mellon University.*