Computing and Informatics, Vol. 37, 2018, 1386–1410, doi: 10.4149/cai_2018_6_1386

A COOPERATIVE LOCAL SEARCH METHOD FOR SOLVING THE TRAVELING TOURNAMENT PROBLEM

Meriem Khelifa, Dalila Boughaci

LRIA-FEI-USTHB – Computer Science Department
BP 32 El-Alia Beb-Ezzouar, Algiers 16111, Algeria
e-mail: dboughaci@usthb.dz, khalifa.merieme.lmd@gmail.com

Abstract. Constrained optimization is the process of optimizing a certain objective function subject to a set of constraints. The goal is not necessarily to find the global optimum. We try to explore the search space more efficiently in order to find a good approximate solution. The obtained solution should verify the hard constraints that are required to be satisfied. In this paper, we propose a cooperative search method that handles optimality and feasibility separately. We take the traveling tournament problem (TTP) as a case study to show the applicability of the proposed idea. TTP is the problem of scheduling a double round-robin tournament that satisfies a set of related constraints and minimizes the total distance traveled by the teams. The proposed method for TTP consists of two main steps. In the first step, we ignore the optimization criterion. We reduce the search only to feasible solutions satisfying the problem's constraints. For this purpose, we use constraints programming model to ensure the feasibility of solutions. In the second step, we propose a stochastic local search method to handle the optimization criterion and find a good approximate solution that verifies the hard constraints. The overall method is evaluated on benchmarks and compared with other well-known techniques for TTP. The computational results are promising and show the effectiveness of the proposed idea for TTP.

Keywords: Sport scheduling, traveling tournament problem (TTP), optimization, constraints, search methods, stochastic local search method (SLS)

Mathematics Subject Classification 2010: 68xxx, 68Uxx, 90-08

1 INTRODUCTION

Constrained optimization involves minimizing or maximizing a certain objective function subject to a set of constraints. The main goal is to explore efficiently the search space in order to find a good approximate solution that verifies the hard constraints that are required to be satisfied. Constrained optimization problems are often difficult to solve, due to an eventual complex interaction between the goals of optimizing the objective function while satisfying the constraints.

Several methods have been proposed for solving constrained optimization problems. These methods include: the penalty function based method, the Lagrange multiplier method that can solve optimization problems with equality constraints, the augmented Lagrange multiplier for inequality constraints that combines the classical Lagrange method with the penalty function method, the quadratic programming methods (QP) that can solve optimization problems with a quadratic objective function and linear constraints, the gradient projection method for equality constraints and the gradient projection that can be extended to solve optimization problems with linear inequality constraints [26, 20].

We propose a cooperative search method for handling in two steps: *feasibility* and *optimality*. More precisely, the proposed search method consists of two main steps. In the first step, we search for feasible configurations satisfying the problem's constraints and ignore the optimization criterion. In the second step, we explore the feasible search space to handle the *optimization* problem and find the best solution. For this purpose, we propose to use constraints programming model, in the first step, to ensure the *feasibility* criterion of solutions. In the second step, we use a metaheuristic approach to handle the *optimization* criterion and find a good approximate solution that verifies the hard constraints.

To show the applicability of the proposed search method, we take as a case study the traveling tournament problem (TTP) which is a challenging sports scheduling problem [17, 24, 19]. The objective of the TTP is to find a double-round-robin tournament schedule that minimizes the total distance traveled by the teams and satisfies the related TTP constraints [11, 17, 24].

TTP is an interesting problem in both sports scheduling and combinatorial optimization. It is known to be an NP-hard problem which makes finding quality solutions in a short amount of time difficult [27]. TTP has attracted significant interest recently since a favorable TTP schedule can generate large incomes in the budget of managing the league's sport.

Several methods have been studied for the TTP. Among them, we cite: the branch and price method [16], the iterated local search method [7] that has been applied for a special case of TTP, so-called TTPPV (the traveling tournament problem with predefined venues). In [8], an integer programming formulation is proposed to the Max-MinTTP variant of TTP, in which the problem of minimizing the longest traveled distance is addressed. A hybrid approach combining a local search heuristic with an integer programming method was designed for TTP in [12]. Further, a simulated annealing algorithm that explores feasible and infeasible schedules us-

ing several structures of neighborhoods and compound movements is studied in [2]. A variable neighborhood search based method (VNS) is proposed in [18] and recently a harmony search method is studied for the mTTP variant (mirored traveling tournamant problem) [19].

In this work, first, we study two local search methods for TTP, which are simulated annealing (SA) and variable neighborhood search (VNS). The two methods are used as a first-step for finding a feasible solution that satisfies the problem constraints. As a second step, we propose a stochastic local search algorithm (SLS) to find a good approximate solution for TTP that minimizes the total distance traveled by the teams. The overall method is implemented, tested on benchmarks and compared with other well-known techniques for TTP.

Comparing our approach and the work of Anagnostopoulos et al. [2], we give the following differences.

- In our method, the search is limited to feasible solutions while Anagnostopoulos et al. explore both feasible and infeasible schedules [2].
- In our method, we handle the TTP problem in two main steps. 1) After generating an initial double round-robin configuration, we apply a local search method based on a constraint satisfaction problem encoding. We define a cost function to select feasible solutions, i.e., those having a zero value cost. 2) We propose a stochastic local search method (SLS) to further improve the solution. SLS is limited to feasible solutions. SLS minimizes the total distance traveled by the teams. The objective function used by SLS computes this total distance. There is no need to add a penalty function since we explore feasible configurations of the search space [2].
- Contrary to our methodology, Anagnostopoulos proposed a simulated annealing method (SATTP) in one step. This SATTP starts with a random initial solution obtained by using a simple backtrack search. The cost function combines travel distances and the number of violations. In Anagnostoploulos constraint violations are penalized [2].

The rest of this paper is organized as follows: Section 2 gives background on the traveling tournament problem (TTP). Section 3 presents in detail the proposed method applied to the TTP problem. Some numerical results are given in Section 4. Finally, Section 5 concludes the work and gives some perspectives.

2 PROBLEM DEFINITION

The traveling tournament problem (TTP) is the problem of scheduling a double round-robin tournament, while satisfying a set of related constraints and minimizing the total distance traveled by the teams [13, 11, 28, 21].

The problem can be stated as follows: let us consider n teams (n even and positive), a double round robin tournament is a set of games in which every team plays every other team exactly once at home and once away. A double round robin

tournament has $2 \cdot (n-1)$ slots. The distance between team cities are given by $n \cdot n$ symmetric matrix *Dis*, such that an element Dis_{ij} of *Dis* represents the distance between the homes of the teams t_i and t_j. The teams begin in their home city and must return there after the tournament.

The TTP is the problem of finding a feasible schedule that minimizes the distance traveled by the teams, and satisfies the following constraints:

Double round robin constraint (DRRT): that means that each team plays with every other team exactly twice, once in its home and once in the home of its opponent.

AtMost constraint: each team must play no more than U and no less than L consecutive games at home or away. Specifically, in our case, L is set to 1 and U to 3.

NoRepeat constraint: A game $t_i - t_j$ can never be followed in the next round by the game $t_j - t_i$.

1. The TTP contains the number of teams (denoted n) and the distance matrix (denoted *Dis*).
2. The output are: a double round robin tournament on the n teams respecting the three constraints AtMost, NoRepeat and DRRT, and where the total distance traveled by the teams is minimized.

Table 1 gives an example of a schedule for $n = 4$ teams. The negation sign means that the team plays away.

Round / **Team**	$Round_1$	$Round_2$	$Round_3$	$Round_4$	$Round_5$	$Round_6$
t_1	3	2	4	−3	−2	−4
t_2	−4	−1	−3	4	1	3
t_3	−1	4	2	1	−4	−2
t_4	2	−3	−1	−2	3	1

Table 1. Example of double round robin tournament with $n = 4$

This schedule specifies that the team t_1 has the following schedule: it successively plays against teams t_3 at home, t_2 at home, t_4 at home, t_3 away, t_2 away and t_4 away. The travel cost of team t_1 is: $Dis_{13} + Dis_{32} + Dis_{24} + Dis_{41}$.

We note that the travel costs of a schedule S is the sum of the travel cost of every team (denoted Travel-cost(S)).

3 THE PROPOSED APPROACH APPLIED TO TTP

We propose a new method for the TTP problem. The proposed method consists of two main steps. First, we start with an initial configuration satisfying the DRRT constraint. Then, we apply a local search method to generate a feasible configuration

that verifies the three constraints: AtMost, NoRepeat, and DRRT. For this step, we study two local search methods which are: variable neighborhood search (VNS) and simulated annealing (SA) where the role is to handle the feasibility criterion of solutions. In the second step, the feasible configuration found by the local search step is sent to the stochastic local search method (SLS) for minimizing the total distance traveled by the teams. Furthermore, we design a new technique which we call the aspiration technique. This technique is used to ensure that the three hard constraints are always satisfied when applying the SLS algorithm on the feasible search space.

3.1 The Initial Configuration

The search method starts with an initial configuration verifying the DRRT constraint. We create this configuration based on graph-theory modelling [9] as follows:

We have $n/2$ games per round and $2 \cdot (n-1)$ rounds. We number the vertices of the graph from 1 to n, where n is the number of teams. We put the top n in the center and the other vertices in a circle around the top n.

- In the first day, we organize a game between (Team 1 and Team n), (Team 2 and Team $n-1$), (Team 3 and Team $n-2$), and so on, up to the game between (Team $n/2$ and Team $n/2+1$).
- In the following day, we reproduce what happened in the previous day, making a simple clockwise rotation of the coupling.

Figure 1 shows an example of the creation of the initial configuration for the case of 6 teams.

$Round_1$	(t_1, t_6)	(t_5, t_2)	(t_4, t_3)
$Round_2$	(t_5, t_4)	(t_1, t_3)	(t_6, t_2)
$Round_3$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_4$	(t_1, t_2)	(t_5, t_3)	(t_6, t_4)
$Round_5$	(t_5, t_6)	(t_1, t_4)	(t_2, t_3)

Table 2. Single round robin tournament with $n = 6$

For more details, Table 2 gives the schedule which is a single round robin tournament **(SRR)**. We note that in a single round-robin schedule, each team plays every other team once. When each team plays all others twice, this is called a double round-robin tournament (DRRT). As shown in Table 3, the double round robin tournament schedule can be obtained by adding the mirror of the SRR schedule.

3.2 Local Search Method for Feasible Schedules

After having created the initial DRRT schedule, we call the local search method in order to locate feasible configurations satisfying the three constraints: AtMost,

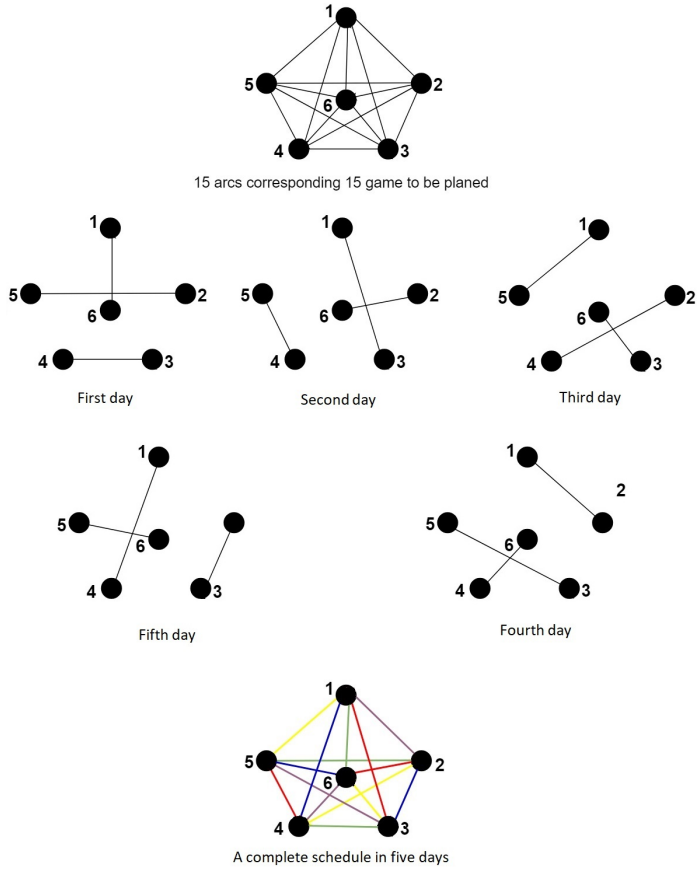

Figure 1. Creation of the initial configuration

$Round_1$	(t_1, t_6)	(t_5, t_2)	(t_4, t_3)
$Round_2$	(t_5, t_4)	(t_1, t_3)	(t_6, t_2)
$Round_3$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_4$	(t_1, t_2)	(t_5, t_3)	(t_6, t_4)
$Round_5$	(t_5, t_6)	(t_1, t_4)	(t_2, t_3)
$Round_6$	(t_6, t_1)	(t_2, t_5)	(t_3, t_4)
$Round_7$	(t_4, t_5)	(t_3, t_1)	(t_2, t_6)
$Round_8$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_9$	(t_2, t_1)	(t_3, t_5)	(t_4, t_6)
$Round_{10}$	(t_6, t_5)	(t_4, t_1)	(t_3, t_2)

Table 3. Mirror of single round robin tournament $n = 6$

NoRepeat, and DRRT. We study VNS and SA local search methods for building feasible schedules. We use a cost function to penalize configurations that violate the considered constraints. Further, we use neighborhood structures to explore the search space. The cost function, the neighborhood structures and the two methods are detailed in the following.

3.2.1 The Cost Function

The cost function consists of two terms. The first term permits to penalize configurations not satisfying the *AtMost* constraint. The second term is to penalize those not satisfying the *NoRepeat* constraint.

First, it is important to give the following useful notations in Table 4 to represent the constraints.

t_i	is the team number i where $i \in [1, n]$.
(t_i, t_j)	is the game t_i, vs. t_j in the home of t_i.
$Round_l$	is the round number l where $1 \leq l \leq \lvert Round \rvert$.
$R_{i,j}$	means that the match (t_i, t_j) is scheduled in a round $R_{i,j}$ where $1 \leq R_{i,j} \leq \lvert Round \rvert$, $\forall i, j \in \lvert T \rvert, i \neq j$. For example in Table 3, the match (t_5, t_1) is scheduled in $Round_8$ at home of t_5, $R_{5,1} = 8$, while the match (t_1, t_5) is scheduled in $Round_3$, $R_{1,5} = 3$.
S	is a DRRT schedule.

Table 4. Some useful notations and definitions

Now, we define the No-repeat constraint $f_{No\_repeat} \cdot occ\_norepeat(S, t_i, t_j)$. The function $occ\_norepeat(S, t_i, t_j)$ verifies, in a current schedule S, if the match (t_i, t_j) is followed in the next round by the match (t_j, t_i). This occurs when $R_{i,j} = R_{j,i} + 1$ or when $R_{j,i} = R_{i,j} + 1$.

$$occ\_norepeat(S, t_i, t_j) = \begin{cases} 1, & \text{if } (R_{i,j} = R_{j,i} + 1) \vee (R_{j,i} = R_{i,j} + 1), \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

For example, the schedule in Table 3 (denoted S_3), $occ\_norepeat(S_3, t_1, t_6) = 0$ because for the game (t_1, t_6), $R_{1,6} \neq R_{6,1} + 1$ and $R_{6,1} \neq R_{1,6} + 1$. The match (t_1, t_6) is played in $R_{1,6} = Round_1$ while the match (t_6, t_1) is scheduled in $R_{6,1} = Round_6$. This implies that $R_{1,6} \neq R_{6,1} + 1$.

The penalty $f_{No\_repeat}(S)$ is the total number of times that the game (t_i, t_j) is followed immediately by (t_j, t_i) in the schedule S.

$$f_{No\_repeat}(S) = \sum_{i=1}^{|T|} \sum_{j=i+1}^{|T|} occ\_norepeat(S, t_i, t_j). \quad (2)$$

Also, we define the At-Most constraint f_{At_Most}. $occ\_atmost\,(S, t_i, Round_l)$ is the number of times that the team t_i plays home or away games in three rounds successively from $Round_l$ ($Round_l, Round_l + 1, Round_l + 2, Round_l + 3$).

$$con\_atmost(S, t_i, Round_l) = \begin{cases} 1, & \text{if } occ\_atmost(S, t_i, Round_l) > 3, \\ & Round_l < |Round| - 3, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

For example, for the schedule S_3 in Table 3: $occ\_atmost\,(S_3, t_1, Round_1) = 1$, since the team t_1 is played 4 consecutive games at home in $Round_1$, $Round_2$, $Round_3$ and $Round_4$.

The penalty At-most $f_{At\_most}\,(S)$ is the total number of times the teams play more than three consecutive home games or three consecutive away games.

$$f_{At\_most}\,(S) = \sum_{i=1}^{|T|} \sum_{l=1}^{|Round|-3} con\_atmost\,(S, t_i, Round_l)\,. \tag{4}$$

The cost function is then defined by the sum of the two penalty constraints **NoRepeat** and **AtMost**:

$$\mathbf{Cost}(S) = f_{No\_repeat}(S) + f_{At\_most}(S). \tag{5}$$

The main goal of the local search method is to find a configuration of zero-cost value. This means finding a feasible configuration satisfying the three constraints: AtMost, NoRepeat, and DRRT.

3.2.2 Neighborhood Structures

We use three neighborhood structures which are detailed in the following:

- N_1: Swap Home. This move swaps the home/away roles of teams. For instance, when we take two teams t_i and t_j, the move $Swap\ Home(S, t_i, t_j)$ swaps the home/away roles of a game involving the teams t_i and t_j. If team t_i plays home against team t_j at $Round_k$ and away against team t_j at $Round_l$ then the move Swap Home (S, t_i, t_j) gives the same schedule as S, except that team t_i plays away against team t_j at $Round_k$, and home against t_j at $Round_l$. Table 5 displays an example of a move using the N_1 neighborhood structure.
- N_2: Swap Round. This move consists of swapping all games of a given pair of rounds. For example the move Swap Round $(S, Round_k, Round_l)$ swaps two given rounds ($Round_k$ and $Round_l$). Table 6 gives an example of schedule when applying the N_2 move.
- N_3 Swap Team. This move corresponds to swapping all opponents of a given pair of teams over all rounds, For example the move Swap Team (S, t_i, t_j) corresponds to swapping all opponents of teams t_i and t_j over all rounds. Table 7 shows an example of applying the N_3 move.

$Round_1$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_3, t_1)	(t_4, t_6)	(t_2, t_5)
$Round_4$	(t_6, t_1)	(t_2, t_3)	(t_4, t_5)
$Round_5$	(t_2, t_1)	(t_6, t_5)	(t_4, t_3)
$Round_6$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_7$	(t_1, t_3)	(t_6, t_4)	(t_5, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_3, t_2)	(t_5, t_4)
$Round_{10}$	(t_1, t_2)	(t_5, t_6)	(t_3, t_4)

The application of the move Swap home away: N_1 (S, t_1, t_2):

$\Downarrow$

$Round_1$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_3, t_1)	(t_4, t_6)	(t_2, t_5)
$Round_4$	(t_6, t_1)	(t_2, t_3)	(t_4, t_5)
$Round_5$	(t_1, t_2)	(t_6, t_5)	(t_4, t_3)
$Round_6$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_7$	(t_1, t_3)	(t_6, t_4)	(t_5, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_3, t_2)	(t_5, t_4)
$Round_{10}$	(t_2, t_1)	(t_5, t_6)	(t_3, t_4)

Table 5. Schedule before (up) and after (down) the application of home-away swap

The simulated annealing and variable neighborhood search (VNS) algorithms are used to search for configurations satisfying the three constraints. These methods are called for finding configurations with a cost function 5 equal to zero. Both SA and VNS use the cost function to compare, in terms of quality, two schedules S and S'. We note that the DRRT constraints are always satisfied since we started with an initial configuration satisfying the DRRT constraint.

3.2.3 SA for Feasible Configurations

SA starts with an initial DRRT schedule. Then it moves iteratively (using the N_1 move) from the current schedule to another one in the search space for finding lower cost solutions. When a new schedule with a lower cost is found, it replaces the current solution. When a new schedule of a higher cost is chosen, it replaces the current solution with some probability. This probability is decreased as the algorithm progresses (analogously to the temperature in physical annealing). The SA algorithm is sketched in Algorithm 1.

3.2.4 VNS for Feasible Configurations

VNS is a local search meta-heuristic proposed in 1997 by Mladenovic and Hansen. Various variants of VNS have been proposed since then, but the basic idea is a sys-

$Round_1$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_3, t_1)	(t_4, t_6)	(t_2, t_5)
$Round_4$	(t_6, t_1)	(t_2, t_3)	(t_4, t_5)
$Round_5$	(t_2, t_1)	(t_6, t_5)	(t_4, t_3)
$Round_6$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_7$	(t_1, t_3)	(t_6, t_4)	(t_5, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_3, t_2)	(t_5, t_4)
$Round_{10}$	(t_1, t_2)	(t_5, t_6)	(t_3, t_4)

After applying the move Swap Round: N_2 $(S, Round_3, Round_5)$:

⇓

$Round_1$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_2, t_1)	(t_6, t_5)	(t_4, t_3)
$Round_4$	(t_6, t_1)	(t_2, t_3)	(t_4, t_5)
$Round_5$	(t_3, t_1)	(t_4, t_6)	(t_2, t_5)
$Round_6$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_7$	(t_1, t_3)	(t_6, t_4)	(t_5, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_3, t_2)	(t_5, t_4)
$Round_{10}$	(t_1, t_2)	(t_5, t_6)	(t_3, t_4)

Table 6. Schedule before (up) and after (down) the application of Swap Round

tematic change of neighborhood combined with a local search [14, 22]. Unlike local search, VNS works on a set of different neighborhoods. In our study, we use three ($k = 3$) structures of neighborhood which are: N_1 (Swap Home), N_2 (Swap Round) and N_3 (Swap Team). At each iteration, we select among the three structures one to create neighbor solutions. The proposed VNS uses the deepest descent strategy (DDS) as a subroutine. More precisely, VNS starts with an initial DRRT (S) schedule and then tries to find a good solution in the whole neighborhood in an iterative manner. The DDS procedure is called for each candidate solution constructed by VNS method. As shown in Algorithm 2, DDS explores iteratively the search space of the given solution S and returns the best neighbor solution found in this space. VNS first uses the N_1 structures to create neighbor solutions. When there is no improvement, the neighborhood structure is mapped to N_2 and then to N_3 in the hope to create diverse and good neighbor solutions. As done with SA, VNS works in the same objective to find a feasible configuration. The overall process of VNS is repeated until a schedule with zero-cost is reached ($Cost(S) = 0$).

The VNS algorithm is sketched in Algorithm 3.

$Round_1$	(t_5, t_1)	(t_4, t_2)	(t_3, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_3, t_1)	(t_4, t_6)	(t_2, t_5)
$Round_4$	(t_6, t_1)	(t_2, t_3)	(t_4, t_5)
$Round_5$	(t_2, t_1)	(t_6, t_5)	(t_4, t_3)
$Round_6$	(t_1, t_5)	(t_2, t_4)	(t_6, t_3)
$Round_7$	(t_1, t_3)	(t_6, t_4)	(t_5, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_3, t_2)	(t_5, t_4)
$Round_{10}$	(t_1, t_2)	(t_5, t_6)	(t_3, t_4)

After the application of the move Swap Team: $N_3\ (S, (t_3, t_5))$:

$\Downarrow$

$Round_1$	(t_3, t_1)	(t_4, t_2)	(t_5, t_6)
$Round_2$	(t_4, t_1)	(t_3, t_5)	(t_2, t_6)
$Round_3$	(t_5, t_1)	(t_4, t_6)	(t_2, t_3)
$Round_4$	(t_6, t_1)	(t_2, t_5)	(t_4, t_3)
$Round_5$	(t_2, t_1)	(t_6, t_3)	(t_4, t_5)
$Round_6$	(t_1, t_3)	(t_2, t_4)	(t_6, t_5)
$Round_7$	(t_1, t_5)	(t_6, t_4)	(t_3, t_2)
$Round_8$	(t_1, t_4)	(t_5, t_3)	(t_6, t_2)
$Round_9$	(t_1, t_6)	(t_5, t_2)	(t_3, t_4)
$Round_{10}$	(t_1, t_2)	(t_3, t_6)	(t_5, t_4)

Table 7. Schedule before (up) and after (down) the application of Swap Team

3.3 Stochastic Local Search for TTP

As already mentioned, the two local search methods (SA and VNS) are used in the first step to handle the feasibility criterion. In the second step, we apply SLS on the feasible configuration to handle the optimization criterion and minimize the total distance traveled by the teams.

SLS [15, 3] is a local search method that combines diversification and intensification strategies to locate a good solution. The intensification phase selects the best neighbor solution while the diversification phase selects a random neighbor solution. The diversification phase is applied with a fixed probability $wp > 0$ and the intensification phase with a probability $1 - wp$. The process is repeated until a certain number of iterations called *maxiter* is reached.

To maintain that the three hard constraints are always satisfied in our SLS algorithm, we apply an aspiration technique. The latter is given in the next section.

Algorithm 1: The proposed SA for feasible configurations

Data: A DRRT schedule, $\alpha = 0.9$, the neighborhood structures N_1
Result: A feasible configuration that satisfies AtMost, NoRepeat and DRRT constraints

1 $S_0 \leftarrow$ an initial configuration verifying DRRT
2 $Temp \leftarrow$ an initial temperature
3 $S \leftarrow S_0$
4 **for** $(I = 1$ *to Maxiter*$)$ **do**
5 **if** $(Cost(S) \neq 0)$ **then**
6 $S' \leftarrow$ Choose random configuration using the neighborhood structure (N_1) on S
7 $r \leftarrow$ random number between 0 and 1
8 **if** $(r \prec e^{\frac{Cost(S)-Cost(S')}{Temp}})$ **then**
9 $S \leftarrow S'$
10 $Temp \leftarrow Temp \cdot \alpha$
11 **else**
12 Go to 13
13 Return the schedule S

Algorithm 2: DDS$(S, N_k(S))$

Data: A DRRT schedule S, the neighborhood structures $N_k(S)$
Result: an improved schedule

1 **repeat**
2 Choose $S' \in N_k(S)$ with $Cost(S') \leq Cost(S'')$, $\forall S'' \in N_k(S)$
3 **if** $(Cost(S') < Cost(S))$ **then**
4 $S \leftarrow S'$
5 **until** $Cost(S') \geq Cost(S), \forall S' \in N_k(S)$;
6 Return the schedule S

3.4 Aspiration Technique to Select the Best Neighbor

We propose a new technique which we called *aspiration technique* to filter the search space and keep only the feasible configurations. The aspiration technique permits to memorize information on moves leading to feasible neighbor configurations, starting from a current configuration. First, we explore the search space to locate feasible configurations (denoted CSC) with zero-cost according to the cost function (5) described in Section 3.2.1. Then among them, we take the best one having the minimum traveled distance.

Algorithm 3: The proposed VNS for feasible configurations

Data: A DRRT schedule, the three first Neighborhood structures N_k $(1 \leq k \leq 3)$

Result: A feasible configuration that satisfies AtMost, NoRepeat and DRRT constraints

```
1  S ← an initial configuration verifying DRRT
2  k ← 1
3  S ← local search(S)
4  for (I = 1 to Maxiter) do
5      if (Cost(S) ≠ 0) then
6          S' ← choose random solution (N_k(s))
7          S" ← Call DDS (S', N_k)
8          if (Cost((S")) < Cost((S)) then
9              S ← S"
10         else
11             if (k < 3) /* when there is no improvement, the neighborhood structure
                  is changed to the next one                                    */
12             then
13                 k ← k + 1
14             else
15                 k ← 1
16     else
17         Go to 18
18 Return the schedule S
```

For every neighborhood structure, the technique is illustrated as follows:

- For the N_1 neighborhood structure (Swap Home), we use a list of *NBgames* elements where *NBgames* is the number of the games ($NBgames = (n-1) \cdot n$). Each element of the list consists of two cells (see Figure 2). The first cell is the game (t_i, t_j) and the second is the cost of the move $N_1(S, (t_i, t_j))$ (denoted $C\_M(t_i, t_j)$). The latter is the variation of the cost value that must be applied if we swap the home/away roles of game (t_i, t_j) to create S'. If S' is the schedule obtained after applying N_1 on S in (t_i, t_j), i.e. $N_1(S, (t_i, t_j)) = S'$.

$$Cost(S') = Cost(S) + C\_M(t_i, t_j). \tag{6}$$

  With this technique, we can take the subset of swaps that gives feasible solutions with zero cost in time $O(|NBgames|)$.

- For the N_2 neighborhood structure (Swap Round), we use a matrix *Movr* of $|Round| \cdot |Round|$ elements where each element $Movr[Round_i, Round_j]$ represents the variation of the cost value that must be applied if a move exchanges two

Algorithm 4: The SLS method for TTP

Data: a TTP instance, *maxiter*, *wp*
Result: The best schedule S

```
Create an initial configuration (CS) verifying the DRRT constraint
Apply the local search method (VNS or SA) on CS to obtain a
  configuration (CSC) verifying the three constraints
S ← CSC
for (I = 1 to Maxiter) do
    if (r ≺ wp) then
        Apply the aspiration technique N1(S) to find a subset of
          neighborhoods with Cost equal to zero.
        Choose a random neighbor among them S' ∈ N1(S)
        S ← S'
    else
        Create Movr[ti, tj] using aspiration technique on N3(S)
        Select the best movement (ti, tj)
        S ← SwapTeam (S, ti, tj)
        Create a list (NBgames) using Aspiration technique, on N2(S)
        Select the best movement (ti, tj)
        S ← SwapHome (S, ti, tj)
Return the schedule S
```

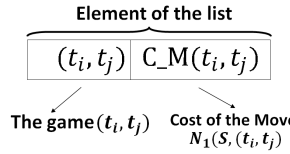

Figure 2. Structure of an element of our list

rounds: $Round_i$ and $Round_j$) to create the schedule S'. Thus the cost of $S' \in N_2(S)$ is obtained by the sum of the $Cost(S)$ and the value of the element $Movr[Round_i, Round_j]$: If S' is the schedule obtained after applying N_2 on S in $(Round_i, Round_j)$ (i.e. $N_2(S, (Round_i, Round_j)) = S'$).

$$Cost(S') = Cost(S) + Movr[Round_i, Round_j]. \tag{7}$$

We can get a subset of neighborhoods that have the cost-value equal to zero in time $\leq (O|N_2(S)|)$. We note that after every move, we update the aspiration matrix *Movr*.

- For N_3 (Swap Teams), we use a matrix *Movt* of $|T| \cdot |T|$ element where each element represents the variation of cost that must be applied if a move exchanges two plans of two teams (t_i, t_j). Thus the cost of S' (S after a move) is obtained

by the sum of the cost of S and the value of the element $Movt[t_i, t_j]$: If $S^{'}$ obtained after applying N_3 on S in (t_i, t_j), i.e. $N_3(S, (t_i, t_j)) = S^{'}$.

$$Cost(S^{'}) = Cost(S) + Movt[t_i, t_j]. \quad (8)$$

We can obtain a subset of feasible neighbor configurations with zero cost value in time equal to $O(|N_3(S)|)$. The best neighbor is obtained by browsing this subset in time $\leq O(|N_3(S)|)$. The matrix is updated after each move.

The SLS is sketched in Algorithm 4.

4 EXPERIMENTS

The source code is written in Java. The experiments are performed on a Core Duo (1.60 GHz) with 2 GB of RAM. We evaluate our method on five different sets of instances available at the website [1]. We validate our method on the most popular testbed that includes: the so-called *NLx* instances, Circular distance instance $CIRCx$, Super Instance, Galaxy and the CON instances [29, 25, 19]. The description of these instances is given as follows:

NLx instances are based on real data of the US National Baseball League, where x is an even number of teams.

CONx is the constant distance instances are characterized by a distance of one (1) between all teams.

SUPERx is based on Rugby League, a league with 14 teams from South Africa, New Zealand and Australia.

CIRCx instances: all teams are placed on a circle, with unit distances (distance of 1 between all adjacent nodes). The distance between two teams i and j with $i > j$ is then equal to the length of the shortest path between i and j which is the minimum of $i - j$ and $j - i + n$.

4.1 Parameter Tuning

The adjustment of the different parameters of the proposed algorithms is fixed by an experimental study. The set values are those for which a good compromise between the quality of the solution obtained by the algorithm and the running time of the algorithm is found. Due to the non-deterministic nature of the SLS method, for each instance, 20 runs have been considered, each of them for 10 000 iterations.

For the probability wp, a large wp ($wp > 0.6$) may cause of being trapped in local solutions, while a scriptsize wp ($wp < 0.3$) means every solution is chosen from the search space randomly, which may decrease explored more thoroughly the promising regions in search space. Therefore, we should use wp value between 0.3 and 0.6. Hence in our study the wp probability is set to 0.4.

For VNS, after twelve runs we observed that the average necessary number of iterations to find a CSC solution is about 8 000 iterations. Furthermore, in VNS the feasible schedule is found at each SLS run. The superiority of VNS is due to the good combination of the diversification and the intensification in the search space this by systematically changes the proposed neighborhood in two phases: firstly, descent to find a local optimum and finally, a perturbation phase to get out of the corresponding valley.

4.2 Numerical Results

In the following, we present the numerical results found by the proposed approach. First, we give in Table 8 (respectively in Table 9) the results found by SA (respectively VNS). The first column gives the number of teams which are instances with an even number of teams, from $n = 6$ up to $n = 36$, the column *Nbr_mov* represents the number of necessary moves to obtain a feasible configuration verifying the three constraints DRRT, AtMost, and NoRepeat. The column Time gives the CPU time in seconds to obtain the feasible configuration (the reported time is the best obtained time to find the feasible solution).

n	Nbr_mov	Time	n	Nbr_mov	Time
6	1	0.023	22	2 001	335.60
8	2	0.050	24	2 421	1 119.63
10	3	0.091	26	2 621	1 455.85
12	83	18.53	28	3 015	1 654.87
14	92	22.703	30	4 045	1 893.91
16	192	113.50	32	5 113	3 221.51
18	281	181.36	34	6 342	4 761.34
20	1 945	318.18	36	8 782	5 931.34

Table 8. The results found by SA

n	Nbr_mov	Time	n	Nbr_mov	Time
6	1	0.010	22	1 623	456.11
8	1	0.024	24	1 937	987.01
10	3	1.12	26	2 134	1 221.23
12	46	17.22	28	2 995	1 546.67
14	78	19.67	30	3 110	1 224.34
16	95	97.00	32	4 675	2 551.00
18	119	112	34	5 112	3 243.16
20	978	234.77	36	6 433	4 056.44

Table 9. The results found by VNS

We implemented two variants of our method: $\text{SLS}_{with\ SA}$ and $\text{SLS}_{with\ VNS}$. The first one is SLS combined with SA for a local search method. In the second one, we

use VNS instead of SA as a local search. Table 10 compares the time consumed by the two methods. The time is given in second.

As shown in Tables 8 and 9, both SA and VNS succeed in finding the feasible configuration (schedule satisfying DRRT, AtMost, and NoRepeat) for all the considered benchmarks (until n equals to 36 teams). The two methods VNS and SA are comparable in term of ability to find feasible configurations in comparable time. However, when we study the overall methods ($SLS_{with\ VNS}$) and ($SLS_{with\ SA}$), we can remark that VNS accelerates the search of solutions when it is integrated in SLS contrary to SA. We can see clearly that VNS is better than SA when it is integrated into SLS in term of total CPU time consuming. We draw Figure 3 to show this behavior.

Instance	**$SLS_{with\ SA}$**	**$SLS_{with\ VNS}$**
	Time	**Time**
CON4	9.89	**0.10**
CON6	24.80	**19.80**
CON8	55.47	**30.22**
CON10	104.21	**89.29**
CON12	161.04	**104.75**
CON14	344.64	**214.44**
CON16	629.99	**411.99**
CON18	1 145.00	**891.29**
CON20	1 887.12	**905.75**
CON22	2 008.05	**1 224.21**
CON24	4 507.58	**2 998.99**
NL4	98.11	**53.23**
NL6	341.89	**145.21**
NL8	967.34	**524.15**
NL10	1 064.31	**575.21**
NL12	2 681.38	**974.83**
NL14	3 671.10	**2 452.93**
NL16	7 343.24	**5 316.34**
CIRC4	123.36	**94.65**
CIRC6	370.62	**201.32**
CIRC8	612.55	**431.96**
CIRC10	1 449.50	**866.11**
Galaxy 4	160.71	**108.14**
Galaxy 6	430.11	**288.50**
SUPER4	1 941.67	**897.89**

Table 10. Time comparison between $SLS_{with\ SA}$ and $SLS_{with\ VNS}$

To better analyze the obtained results and demonstrate the performance of our approach we compare the proposed method to the best known solution and other the best-known techniques for TTP.

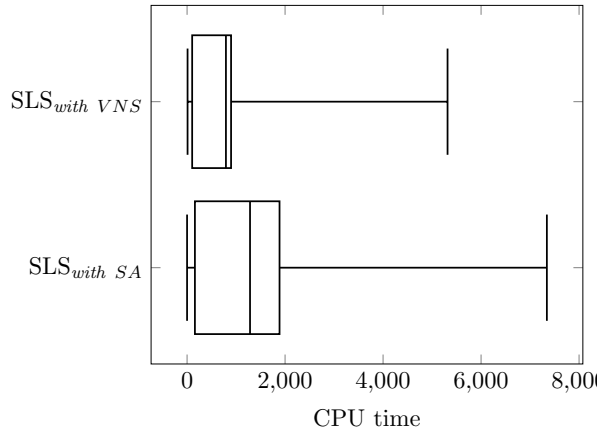

Figure 3. CPU time comparison between SLS$_{with\ SA}$ and SLS$_{with\ VNS}$

Table 11 gives the numerical results found by the overall approach. We give the CPU time (Time) in seconds (the reported time is the time needed to find the best solution), the best (Best) and the average solution (AVG) of twenty executions found by our method. We give the best known solutions (Best*) [1] for each instance and the gap between Best and Best*. The best results are in bold font. The proposed method SLS is compared with the best-known solutions Best* for TTP in order to show its performance in solving TTP.

$$Gap\,\% = \frac{\text{SLS}(Best) - Best*}{\text{SLS}(Best)} * 100. \tag{9}$$

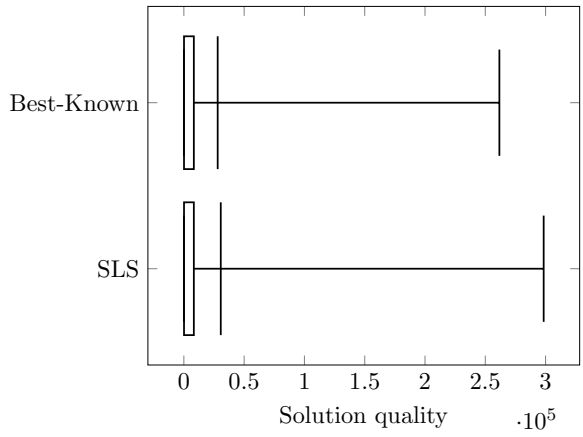

Figure 4. Comparison between SLS and Best-known

Instance	**Best***	**SLS**			**Gap %**
		Time(s)	**Best**	**Average**	
CON4	**17**	0.10	**17**	17	**0**
CON6	**43**	19.80	**43**	43	**0**
CON8	**80**	30.22	**80**	80	**0**
CON10	**124**	89.29	**124**	125	**0**
CON12	**182**	104.75	**182**	184	**0**
CON14	**252**	214.44	**252**	254	**0**
CON16	327	411.99	336	338	2.67
CON18	417	891.29	433	455	3.69
CON20	522	905.75	525	554	0.57
CON22	**628**	1 224.21	**628**	632	**0**
CON24	749	2 998.99	756	808	0.92
NL4	**8 276**	53.23	**8 276**	8 276	**0**
NL6	**23 916**	145.21	**23 916**	24 122	**0**
NL8	**39 947**	524.15	40 621	42 234	1.65
NL10	59 583	575.21	61 193	62 711	2.63
NL12	111 248	974.83	120 655	127 856	7.79
NL14	188 728	2 452.93	206 274	231 785	8.50
NL16	261 687	5 316.34	308 413	322 394	15.15
CIRC4	**20**	94.65	**20**	20	**0**
CIRC6	**64**	201.32	**64**	64	**0**
CIRC8	130	431.96	140	144	7.69
CIRC10	242	866.11	272	287	11.02
Galaxy 4	**416**	108.14	**416**	416	**0**
Galaxy 6	**1 365**	288.50	**1 365**	1 394	**0**
Galaxy8	**2 373**	1 007.01	**2 373**	2 648	**0**
Galaxy10	**3 676**	2 015.56	4 554	5 134	19.27
Galaxy12	7 034	2 897.14	7 354	8 005	4.35
Super 4	**63 405**	897.89	**63 405**	63 405	**0**
Super 6	**130 365**	1 425.11	**130 365**	130 365	**0**

Table 11. A comparison with best-known Best*

SLS succeeds in finding the optimum solutions for CON4, CON6, CON8, CON10, CON12, CON14, NL6, NL4, CIRC6, Galaxy4, Galaxy6, Galaxy8, Super4 and Super6. The gap between the best-known solutions and the results of our approach, in general, does not go above 15.15 % for NL instances and is between 1 and 3.67 % for CON instances. This demonstrates the effectiveness of the proposed approach in solving the traveling tournament problem. In addition to the numerical analysis, we draw the box plots diagrams to better visualize the distribution of cost value. The boxplot in Figure 4 shows that our method produces consistent results. The results are interesting and demonstrate the benefit of our approach for TTP.

4.3 A Comparative Study for NL Instances

Since the NLx family of instances is probably the most researched TTP benchmark-family and virtually all researches studying the TTP publish their computational results with NLx instances, in this section, we compare the proposed method to other well-known techniques for TTP on NL instances.

Table 12 compares SLS with some well-known techniques for TTP. The comparison is done with the following techniques: GA (which is a genetic algorithm with novel encoding scheme for representing a solution instance [6]), AIS_{TTP} (which is an immune-inspired algorithm based on the CLONALG framework [4]), $CTSA$ (which is a hybrid integer programming/constraint programming approach and a branch and price algorithm [23]), $AATTP$ (which is an approximation Algorithm for TTP [29]), $CPMT$ (which is a tabu search and simulated annealing [25]) and *ANT-HYP* (which is an ant based hyper-heuristic [5]).

The comparison with other techniques shows the efficiency of our method. In order to quantify this improvement we compute the performance ratio (PR) or the average gaps given as follows:

$$PR = \sum_{i=1}^{NBins} Gap_i / NBins \tag{10}$$

where the Gap_i is the gap between the best solution of our method and the best of the other techniques of the instance i. *NBins* is the total number of the considered instances.

The performance ratio between the proposed approach and $AATTP$ is equal to 8.68 % which means that our approach improves the results of $AATTP$ in average by 8.68 %. Also, our method gives better average than $CTSA$ with 0.76 % and improves the results of $CPMT$ and *ANT-HYP* method in average by 2.32 % and 3.61 %, respectively. Further, the proposed approach is able to perform better than GA results in average by 1.13 %, whereas our approach finds near solutions with deviation from AIS_{TTP} equal to -0.05 %.

4.4 A Comparative Study for CON Instances

For CON instances, the comparison is done with Tabu Search [10] since it achieves the best-known solution on almost all CON instances. Table 13 gives the results found by both SLS and Tabu search on CON instances. According to these numerical results, SLS succeeds in finding better results for all the checked instances compared to Tabu search. Furthermore, the performance ratio between our approach and Tabu search method is equal to 2.25 % which means that the proposed approach enhances the results of Tabu search in average by 2.25 %.

The superiority of our approach is explained by the good combination of the two main steps, which permits to explore efficiently the search space and locate good solutions.

Instance	**SLS**	**AATTP**	Gap	AIS_{TTP}	Gap	**CTSA**	Gap
NL4	**8 276**	—	—	—	—	—	—
NL6	**23 916**	—	—	—	—	24 467	**−2.30**
NL8	**40 621**	47 128	**−16.01**	40 156	1.14	41 754	**−2.78**
NL10	**61 193**	69 958	**−14.32**	61 351	**−0.25**	63 277	**−3.40**
NL12	**120 655**	125 086	**−3.67**	120 531	0.21	116 421	3.05
NL14	**206 274**	230 874	**−11.92**	206 434	**−0.77**	215 665	**−4.55**
NL16	**308 413**	300 744	2.48	288 674	0.04	288 674	5.40

Instance	**SLS**	**ANT − HYP**	Gap	**CPMT**	Gap	**GA**	Gap
NL4	**8 276**	—	—	—	—	—	—
NL6	**23 916**	23 916	**0**	—	—	23 916	**0**
NL8	**40 621**	40 361	0.64	41 928	**−3.21**	41 505	**−2.17**
NL10	**61 193**	65 168	**−6.49**	65 193	**−6.53**	—	—
NL12	**120 655**	123 752	**−2.56**	120 906	**−0.20**	—	—
NL14	**206 274**	225 169	**−9.16**	208 824	**−1.23**	—	—
NL16	**308 413**	321 037	**−4.09**	287 130	6.90	—	—

Table 12. A comparative study for NL instances

Instance	**SLS**	**Tabu Search**	Gap %
CON4	**17**	**17**	**0**
CON4	**43**	48	**−11.62**
CON8	**80**	81	**-1.25**
CON10	**124**	**124**	**0**
CON12	**182**	184	**−1.09**
CON14	**252**	253	**−0.09**
CON16	**336**	342	**−1.72**

Table 13. A comparative study for *CON* instances

5 CONCLUSION

We proposed a search method for constrained optimization. The proposed method handles optimality and feasibility separately. It is applied to the well-known NP-hard traveling tournament problem (TTP). TTP is concerned with finding a tournament schedule that minimizes the total distances traveled by the teams. The TTP has attracted significant interest recently since a favorable TTP schedule can generate large incomes in the budget of managing the league's sport. The proposed approach is a combination of the stochastic local search algorithm (SLS) as an optimization technique and the local search method (VNS/SA) as a search method for feasible configurations. The method is implemented, evaluated on publicly available standard benchmarks and compared with other techniques for TTP. The proposed method provides competitive results and finds solutions of high quality. It matches the best-known solutions on seventeen instances and outperforms some interesting

methods. The advantage of the novel method is that it can reduce mainly the problem complexity since we consider only feasible configurations. However, we can lose the global optima in some situation since we do not explore full search space. We plan to study the impact of exact techniques on the proposed method. It would be nice to study the effectiveness of our method in solving other constrained optimization problems.

REFERENCES

[1] Challenge Traveling Tournament Instances. `http://mat.tepper.cmu.edu/TOURN/`, 2016.

[2] Anagnostopoulos, A.—Michel, L.—Van Hentenryck, P.—Vergados, Y.: A Simulated Annealing Approach to the Traveling Tournament Problem. Journal of Scheduling, Vol. 9, 2006, No. 2, pp. 177–193, doi: 10.1007/s10951-006-7187-8.

[3] Boughaci, D.: Metaheuristic Approaches for the Winner Determination Problem in Combinatorial Auction. In: Yang, X. S. (Ed.): Artificial Intelligence, Evolutionary Computing and Metaheuristics. Springer, Berlin, Heidelberg, Studies in Computational Intelligence, Vol. 427, 2013, pp. 775–791, doi: 10.1007/978-3-642-29694-9_29.

[4] Pérez Cáceres, L.—Riff, M. C.: AISTTP: An Artificial Immune Algorithm to Solve Traveling Tournament Problems. International Journal of Computational Intelligence and Applications, Vol. 11, 2012, No. 1, Art. No. 1250008, doi: 10.1142/s1469026812500083.

[5] Chen, P.-C.—Kendall, G.—Vanden Berghe, G.: An Ant Based Hyper-Heuristic for the Travelling Tournament Problem. 2007 IEEE Symposium on Computational Intelligence in Scheduling (SCIS '07), 2007, pp. 19–26, doi: 10.1109/scis.2007.367665.

[6] Choubey, N. S.: A Novel Encoding Scheme for Traveling Tournament Problem Using Genetic Algorithm. International Journal of Computer Applications (IJCA), Special Issue on Evolutionary Computation for Optimization Techniques (ECOT 2010), Vol. 2, 2010, No. 7, pp. 79–82, doi: 10.5120/1536-139.

[7] Costa, F. N.—Urrutia, S.—Ribeiro, C. C.: An ILS Heuristic for the Traveling Tournament Problem with Predefined Venues. Annals of Operations Research, Vol. 194, 2012, No. 1, pp. 137–150, doi: 10.1007/s10479-010-0719-9.

[8] Moreira de Carvalho, M. A.—Nogueira Lorena, L. A.: New Models for the Mirrored Traveling Tournament Problem. Computers and Industrial Engineering, Vol. 63, 2012, No. 4, pp. 1089–1095, doi: 10.1016/j.cie.2012.08.002.

[9] de Werra, D.: Some Models of Graphs for Scheduling Sports Competitions. Discrete Applied Mathematics, Vol. 21, 1988, No. 1, pp. 47–65, doi: 10.1016/0166-218x(88)90033-9.

[10] Di Gaspero, L.—Schaerf, A.: A Composite-Neighborhood Tabu Search Approach to the Traveling Tournament Problem. Journal of Heuristics, Vol. 13, 2007, No. 2, pp. 189–207, doi: 10.1007/s10732-006-9007-x.

[11] Easton, K.—Nemhauser, G.—Trick, M.: The Traveling Tournament Problem Description and Benchmarks. In: Walsh, T. (Ed.): Principles and Practice of Constraint Programming – CP 2001. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2239, 2001, pp. 580–584, doi: 10.1007/3-540-45578-7_43.

[12] Goerigk, M.—Westphal, S.: A Combined Local Search and Integer Programming Approach to the Traveling Tournament Problem. Annals of Operations Research, Vol. 239, 2016, No. 1, pp. 343–354, doi: 10.1007/s10479-014-1586-6.

[13] Guedes, A. C. B.—Ribeiro, C. C.: A Heuristic for Minimizing Weighted Carry-Over Effects in Round Robin Tournaments. Journal of Scheduling, Vol. 14, 2011, No. 6, pp. 655–667, doi: 10.1007/s10951-011-0244-y.

[14] Hansen, P.—Mladenović, N.: Variable Neighborhood Search: Principles and Applications. European Journal of Operational Research, Vol. 130, 2001, No. 3, pp. 449–467, doi: 10.1016/s0377-2217(00)00100-4.

[15] Hoos, H. H.—Boutilier, C.: Solving Combinatorial Auctions Using Stochastic Local Search. Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI), 2000, pp. 22–29.

[16] Irnich, S.: A New Branch-and-Price Algorithm for the Traveling Tournament Problem. European Journal of Operational Research, Vol. 204, 2010, No. 2, pp. 218–228, doi: 10.1016/j.ejor.2009.10.024.

[17] Kendall, G.—Knust, S.—Ribeiro, C. C.—Urrutia, S.: Scheduling in Sports: An Annotated Bibliography. Computers and Operations Research, Vol. 37, 2010, No. 1, pp. 1–19, doi: 10.1016/j.cor.2009.05.013.

[18] Khelifa, M.—Boughaci, D.: A Variable Neighborhood Search Method for Solving the Traveling Tournaments Problem. Electronic Notes in Discrete Mathematics, Vol. 47, 2015, pp. 157–164, doi: 10.1016/j.endm.2014.11.021.

[19] Khelifa, M.—Boughaci, D.: Hybrid Harmony Search Combined with Variable Neighborhood Search for the Traveling Tournament Problem. In: Nguyen, N. T., Iliadis, L., Manolopoulos, Y., Trawiński, B. (Eds.): Computational Collective Intelligence (ICCCI 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9875, 2016, pp. 520–530, doi: 10.1007/978-3-319-45243-2_48.

[20] Kuester, J. L.—Mize, J. H.: Optimization Techniques with Fortran. McGraw-Hill, New York, 1973.

[21] Miyashiro, R.—Matsui, T.: Semidefinite Programming Based Approaches to the Break Minimization Problem. Computers and Operations Research, Vol. 33, 2006, No. 7, pp. 1975–1982, doi: 10.1016/j.cor.2004.09.030.

[22] Mladenović, N.—Hansen, P.: Variable Neighborhood Search. Computers and Operations Research, Vol. 24, 1997, No. 11, pp. 1097–1100, doi: 10.1016/s0305-0548(97)00031-2.

[23] Rasmussen, R. V.—Trick, M. A.: The Timetable Constrained Distance Minimization Problem. Annals of Operations Research, Vol. 171, 2009, No. 1, pp. 45–49, doi: 10.1007/s10479-008-0384-4.

[24] Ribeiro, C. C.: Sports Scheduling: Problems and Applications. International Transactions in Operational Research, Vol. 19, 2012, No. 1-2, pp. 201–226, doi: 10.1111/j.1475-3995.2011.00819.x.

[25] Rossi-Doria, O.—Sampels, M.—Birattari, M. et al.: A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. In: Burke, E., De Causmaecker, P. (Eds.): Practice and Theory of Automated Timetabling IV (PATAT 2002). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2740, 2003, pp. 329–351, doi: 10.1007/978-3-540-45157-0_22.

[26] Snyman, J. A.: Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms. Springer US, Applied Optimization, Vol. 97, 2005.

[27] Thielen, C.—Westphal, S.: Complexity of the Traveling Tournament Problem. Theoretical Computer Science, Vol. 412, 2011, No. 4-5, pp. 345–351, doi: 10.1016/j.tcs.2010.10.001.

[28] Van 't Hof, P.—Post, G.—Briskorn, D.: Constructing Fair Round Robin Tournaments with a Minimum Number of Breaks. Operations Research Letters, Vol. 38, 2010, No. 6, pp. 592–596, doi: 10.1016/j.orl.2010.08.008.

[29] Westphal, S.—Noparlik, K.: A 5.875-Approximation for the Traveling Tournament Problem. Annals of Operations Research, Vol. 218, 2014, No. 1, pp. 347–360, doi: 10.1007/s10479-012-1061-1.

Meriem Khelifa is a Ph.D. student at the University of Science and Technologies Houari Boumediene (USTHB). She received her Master degree in computer science from the University Kasdi Merbah, Ouargla, Algeria. She is an active member of the Laboratory for Research in Artificial Intelligence (LRIA). Her research interests include issues related to the meta-heuristic approaches, optimization problems, and the sports scheduling problem. She is an author of some scientific papers on sport scheduling and traveling tournament problems. Actually, she is working on developing new algorithms based on graph theory and machine learning algorithms to solve large sport scheduling instances at the HERON laboratory (Higher Education Research on Emotional Intelligence and Privacy Protection), UdeM Université de Montréal, Canada.

Dalila Boughaci is Full Professor in computer science at the University of Science and Technology USTHB (Algeria). She got her Ph.D. from the University of Aix-Marseille (France) in 2008 and her "Habilitation" Post-Doctoral diploma from USTHB University in 2009. She earned her Bachelor of Engineering degree in computer science from Algiers University, M.Sc. degree in computer science and her second Ph.D. in programming systems from the University of Sciences and Technology, Beb-Ezzouar, Algiers in 1997, 2001 and 2008, respectively. Her current research interests are in the areas of data mining, deep learning, evolutionary computation, artificial intelligence, meta-heuristics, multi-agent systems, network security, credit scoring and e-commerce. She has published several papers on these research topics in journals and conferences and directed several Ph.D., M.Sc. and B.Sc. students' projects. She has taught parallel computing, machine learning, web service, object oriented programming, algorithmic, software engineering, databases, Java, agents and programming languages at Algiers, and served on several program committees. She is a member of the LRIA Artificial Intelligence Laboratory at the University of Algiers. She is the head of the research team: Optimization, Reasoning and Application of the LRIA Laboratory.