Proceedings of 2017 4th International Conference on
Control, Decision and Information Technologies (CoDIT'17) / April 5-7, 2017, Barcelona, Spain

An enhanced genetic algorithm with a new crossover operator for the traveling tournament problem

Meriem Khelifa
LRIA-FEI- USTHB
Computer Science Department
Email: khalifa.merieme.lmd@gmail.com

Dalila Boughaci
LRIA-FEI- USTHB
Computer Science Department
Email: dboughaci@usthb.dz

Esma Aïmeur
Department of Computer Science and
Operations Research HERON Laboratory
Email: aimeur@IRO.UMontreal.CA

***Abstract*—This paper proposes an enhanced genetic algorithm (E-GA) with a new crossover operator for the well-known NP-hard traveling tournament problem (TTP). TTP is the problem of scheduling a feasible double round robin tournament that minimizes the total distances traveled by the teams. The proposed E-GA for TTP uses a new crossover operator based on sharing the best partial path teams among the schedules. Further, E-GA uses a variable neighborhood search as a subroutine to improve the intensification mechanism in GA. The proposed method is evaluated on publicly available standard benchmarks and compared with other techniques for TTP. The computational experiment shows that the proposed method could build interesting results comparable to other state-of-the-art approaches.**

I. Introduction

Sport scheduling is a research area containing a large number of challenging problems [1], [2] which are very hard to solve. In this paper, we are interested in the traveling tournament problem (TTP) which is a core problem in sports scheduling. TTP is the problem of creating a feasible double round robin schedule that minimizes the overall distances traveled by all teams [2]. TTP is widely believed to be NP-hard [3] which makes difficult finding quality solutions due to strong feasibility issues together with a complex optimization.
Various works in different contexts tackled the problem of traveling tournament scheduling. Among we highlight the following ones: the Branch-and-price-based approaches developed in [4].In [5], the authors designed an algorithm which approximates the optimal solution by a factor of 5.875. An enhanced harmony search combined with a variable neighborhood search (V-HS) for mTTP is presented in [6]. An hybrid algorithm based on integer programming and local search heuristic is proposed to resolve TTP in [7].

In this paper, we propose an enhanced genetic algorithm (E-GA) for TTP. The enhancement is done by using a new fitness crossover operator based on sharing the best partial path teams among the schedules. Further, E-GA calls the variable neighborhood search method (VNS), used at each iteration to improve the solutions created by GA. It is important to note that E-GA can build the remaining rounds of any incomplete schedule, which allows the sports federation to create any schedules with fixing certain rounds. We note that it is not easy to create the remaining rounds of an incomplete schedule without breaking the DRRT constraint.

The rest of this paper is organized as follows: in Section 2, we give the problem description. Section 3 presents in detail the proposed approach for TTP. Section 4 presents some numerical results. Finally, Section 5 concludes in gives some future works.

II. Problem description

The traveling tournament troblem (TTP) is a well-known sport scheduling combinatorial optimization problem which involves n teams $T = (t_1, \ldots, t_n)$ (n is even). TTP is the problem of scheduling a double round-robin tournament (DRRT) that is a set of games in which every team plays every other team exactly once at home and once away and all teams must play only one match every round. Accordingly DRRT has $n(n-1)$ games and $n/2$ games are played in every round. Thus exactly $2(n-1)$ slots (rounds) are required to schedule a Double round robin tournament. The distance between team cities are given by $(n \times n)$ 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 aim of TTP is to find a double round-robin tournament that minimizes the distance traveled by all teams, and satisfies the following related constraints:

1) Each team plays only one game on one day. There is no gap, i.e. every day one game is played.
2) Each pair of teams plays twice, once at each other home.
3) No team should play more than 3 consecutive home or away games.
4) A team cannot play against the same opposition in two consecutive games.

- **The TTP inputs** are: the number n of teams and the Dis distance matrix.
- **The output** will be a Double Round Robin Tournament on the n teams respected the three constraints: AtMost, NoRepeat and DRRT, and the total distance traveled by all the teams is minimized.

Table I gives an example of a TTP schedule for n equals to 6.

This schedule specifies that the team t_1 has the following schedule: it successively plays against teams t_6 at home, t_3 at home, t_5 away, t_2 at home, t_4 at home t_6 away t_3 away, t_2

978-1-5090-6465-6/17/$31.00 ©2017 IEEE

TABLE I. DOUBLE ROUND ROBIN TOURNAMENT (DRRT)(n=6)

R_1	(t_1, t_6)	(t_5, t_2)	(t_4, t_3)
R_2	(t_5, t_4)	(t_1, t_3)	(t_6, t_2)
R_3	(t_5, t_1)	(t_2, t_4)	(t_6, t_3)
R_4	(t_1, t_2)	(t_5, t_3)	(t_6, t_4)
R_5	(t_5, t_6)	(t_1, t_4)	(t_2, t_3)
R_6	(t_6, t_1)	(t_2, t_5)	(t_3, t_4)
R_7	(t_4, t_5)	(t_3, t_1)	(t_2, t_6)
R_8	(t_1, t_5)	(t_4, t_2)	(t_3, t_6)
R_9	(t_2, t_1)	(t_3, t_5)	(t_4, t_6)
R_{10}	(t_6, t_5)	(t_4, t_1)	(t_3, t_2)

away and t_5 at home, t_2 away and t_4 away. The travel cost of team t_1 is :

$$L-path_t(S,t_1) = d(t_1,t_5)+d(t_5,t_1)+d(t_1,t_6)+d(t_6,t_3)+ d(t_3,t_2)+d(t_2,t_1)+d(t_1,t_2)+d(t_2,t_4)+d(t_4,t_1).$$

$$Cost\_travel(schedule) = \sum_{t=1}^{t=n} L-path_t(S,t).$$

III. ENHANCED GENETIC ALGORITHM FOR TTP (E-GA)

Genetic algorithm (GA) is an evolutionary bio-inspired optimization method [8] which works by mimicking the evolutionary principles and chromosomal processing in natural genetics. GA starts with a set of solutions (represented by chromosomes) called population with associated fitness values. Thereafter, the population of solutions is modified to a new population by applying three operators similar to natural genetic operators: the selection, the crossover and the mutation. The selection is the operator that permits to select candidate individuals for the crossing step. The crossover is applied to the parents to obtain a new generation of individuals (offspring), and the mutation is applied in order to introduce diversification into the population. GA works iteratively by successively applying these three operators in each generation till a termination criterion is satisfied.

In this paper, we propose an enhanced genetic algorithm method for TTP. The proposed enhancement consists of the new crossover operator for TTP and the variable neighborhood search (VNS) used at each iteration to improve the quality of solutions.

A. The used neighborhood structures

We used three structures of neighborhood given as follows:
Swap Round(S, R_i, R_j): consists of swapping all games of a given pair of rounds.
Swap Home (S, t_i, t_j): Swap the home/away roles of a game involving the teams t_i and t_j.
Swap Team (S, t_i, t_j): corresponds to swapping all opponents of a given pair(t_i, t_j) of teams over all rounds.

B. Initialization of the population

In our model, each individual represents a schedule. The fitness value of each individual is the total traveling distances. The population corresponds to a set of individuals that are generated by using both intensification and diversification strategies. More precisely, we start with a feasible configuration (S) created by using The well-known polygon method [9]. Then the population is created according to the following processes:

- Generate all the neighbors schedules Set_Neigh from S by using(swap home, swap team,swap rounds);
- Choose the best neighbor $Neigh(S)$ of Set_Neigh with:

Min (Cost_travel(Neigh(S))$-\alpha$(distance_haming(Neigh(S),S)))

$distance\_haming(Neigh(S), S)$ is the Hamming distance between S and it's neighbor $Neigh(S)$ which represent the difference between schedule S and its neighbor. Indeed it's the number of positions at which the corresponding teams are different in both schedules.

- Join the best neighbor to the population. Then repeat the previous processes on each new added schedule to fill the population.

C. Reproduction mechanism

Before explain the reproduction mechanism, we give in following some important notions used in our study.

1) Best Partial Path of the Teams (BPPT): We define *BPPT* as the partial path teams. It is a set of rounds that represents a partial itinerary of all teams. In Fig 1(Parent1), we show an example of a partial path with a length equals to 3 (R_2, R_3, R_4) that represents(with using NL benchmark):

- The itinerary of t1: $t_1(t_5, t_1, t_2)$ with a distance of $(distr = d(t_5,t_1)+d(t_1,t_2)) = 1350$,
- The itinerary of t2: $t_2(t_3, t_2, t_2)$, $(distr = 80)$,
- The itinerary of t3: $t_3(t_3, t_4, t_6)$, $(distr = 780)$,
- The itinerary of t4: $t_4(t_6, t_4, t_4)$, $(distr = 408)$,
- The itinerary of t5: $t_5(t_5, t_2, t_4)$, $(distr = 1427)$,
- The itinerary of t6: $t_6(t_6, t_1, t_6)$, $(distr = 1042)$.

The travel cost of the partial path $\{R_2, R_6, 3_4\} = 1350 + 80 + 780 + 408 + 1427 + 1042 = 3737$. The set of partial path (subsets) that we can create from the set of rounds $\{R_0, R_1 ... R_{2(n-1)}\}$($n$ is the number of teams) is equal to $A_{|R|,lng} = |R|!/\left(|R| - lng\right)!$ (lng is the length of the partial path) and *BPPT* is the best partial path with minimum travel cost.

The reproduction process is controlled by the three GA operators which are: the selection, the crossover and the mutation. At every iteration, the selection of the two parents is done according to a simple ranking mechanism [10] which gives higher chances to schedules with low-cost travel to be used for reproduction, then the crossover operator is applied to create new children from selected parents. The crossover operator which we have proposed is based on trajectory crossover with finding the best partial path teams (BPPT). Then it transfers the set of rounds of BPPT (from parent1) to child into position corresponding to those in the parent1 .The

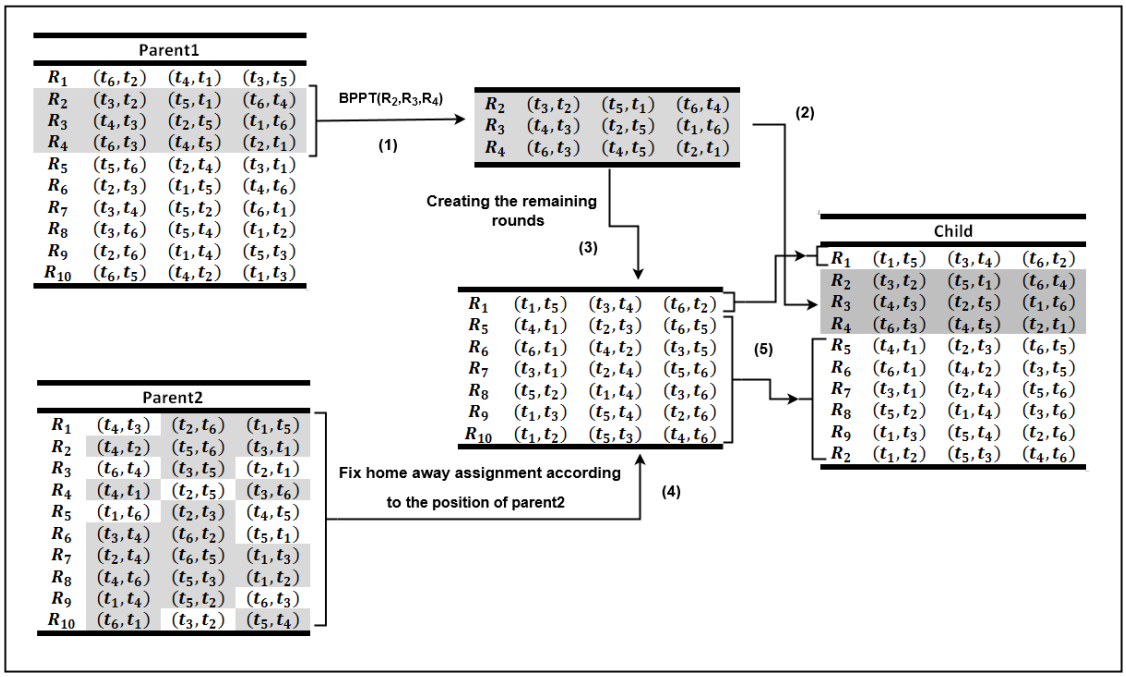

Fig. 1. Example of the crossover process($BPPT = \{R_2, R_3, R_4\}$)

TABLE II. SOME USEFUL DEFINITIONS

ACR	Definition
$L - path_t(S,t)$	Represents the distance traveled by the team t in the schedule$S$ (incomplete or complete schedule)
An incomplete schedule	An incomplete schedule is a schedule with $\|R\| \leq 2(n-1)$

process of creating the remaining rounds $|R| - |BPPT|$) is outlined in **Algorithm**2 with regards to issues of home/away patterns and the position of the games in parent2 (see Fig 1).

2) The proposed crossover: The proposed crossover has the following major steps:

1) Calculate BPPT of the parent1;
2) Transfer the set of rounds of BPPT (from parent1) to child into position corresponding to those in the parent1;
3) Create the remaining rounds of the incomplete schedule BPPT ($\{R_1, R_5, R_6, R_7, R_8, R_9, R_{10}\}$) (section 3), with regards to issues of home/away patterns and the position of the $GameNonBPPT$ in parent2 if it is possible (4).
4) Copy the remaining rounds to the child (see Fig 1).

3) Creation of the remaining rounds: This section presents the procedure of creating the remaining round of the incomplete schedule S (BPPT). Table II represent some useful definitions within the algorithms. In Fig 1 BPPT is an incomplete S schedule with three rounds (R_2, R_3, R_4), $L - path_{t_1}(S, t_1) = d(t_5, t_1) + d(t_1, t_2)$ represent the distance traveled by the team t in the incomplete schedule S (BPPT).

Algorithm 1 : weight assignment to the edges of the graph.

Require: The graph $\vec{G}\ (H, E)$ and the incomplete schedule S (BPPT).
Ensure: A weighted graph $\vec{G}\ (H, E)$

```
for i = 1 to n do
  for j = 1 to n do
    {Check if the edge occurred in any previous rounds in schedule S }
    if e_{t_i,t_j} ∈ S then
      {If the edge occurred the weight of the edge become ∞}
      w(e_{t_i,t_j}) ← ∞
    else
      w e_{t_i,t_j} ← L − path_t(S, t_i) + d(t_i, t_j)
    end if
  end for
end for
return A weighted graph G(H, E).
```

The process of creating the remaining rounds of the incomplete schedule BPPT can be detailed as follows: we create the graph of teams $T = (t_1, \ldots, t_{2n})$. Home location of team t_i is denoted as h_i. The set $H = \{h_i\}_{i=1}^{2n}$ forms a complete directed graph $\vec{G}\ (H, E)$ where H are the nodes and E are edges. The edge $\vec{e_{i,j}}$ corresponds to the match between team i and team j. The direction of the edge indicates that the game takes place in the home city of team i. Fig 2 shows an example of a complete directed graph $\vec{G}\ (H, E), H = \{t_1, t_2, t_3, t_4, t_5, t_6\}$. The weight of the edges : $\forall i, j \in (1...n) : w(e_{t_i,t_j}) = \infty$ if the game between team t_i and t_j occurred in our incomplete schedule (BPPT) otherwise $w(e_{t_i,t_j}) = L - path_t(S, t_i) + d(t_i, t_j)$. The weight assignment

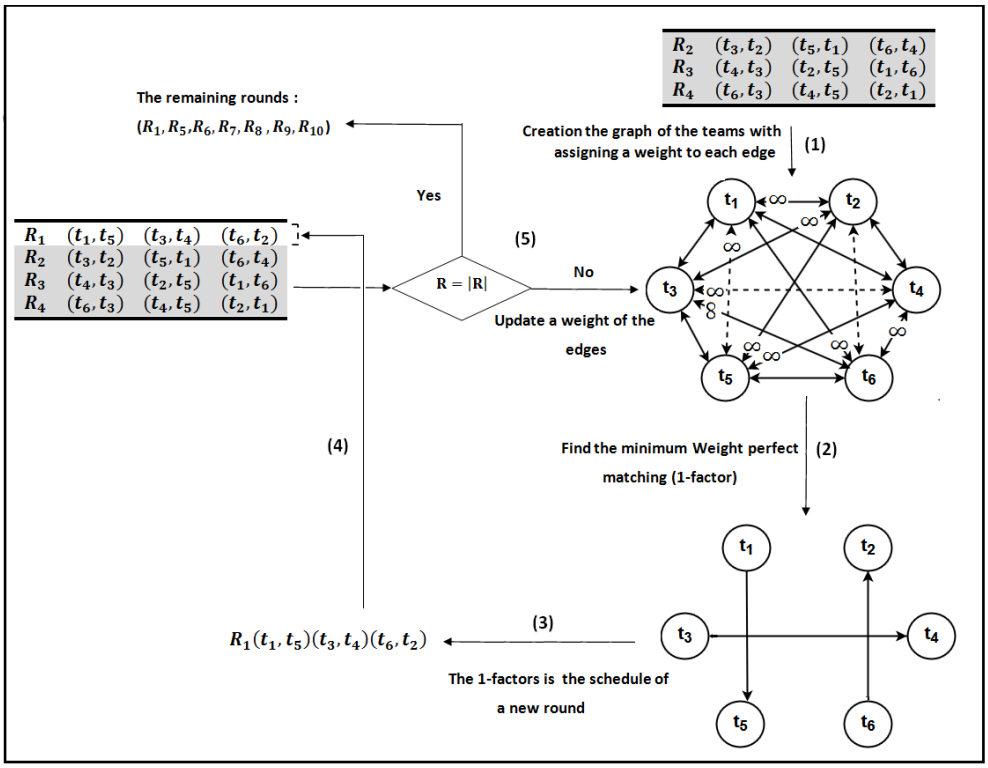

Fig. 2. An example of the creation of the remaining rounds of the incomplete schedule BPPT

is sketched in Algorithm 1.
A-factor of a graph $G(H,E)$ is a sub-graph $G^{'}(H,E^{'})$, with $E^{'} \in E$, all edges of $E^{'}$ are independent(they have no common end vertex) and all the nods have their degrees equal to one. Since in every round each team plays only one game on one round, $n/2$ games are played in every round. Accordingly each round corresponds to a 1-factor in $\overrightarrow{G}(H,E)$.

Algorithm 2 :Creating the remaining rounds

Require: The incomplete schedule S (BPPT).
Ensure: A complete schedule with the rest rounds
1: Create the graph of the teams $\overrightarrow{G}(H,E)$
2: Assigning a weight to the edge of the graph($\overrightarrow{G}(H,E),S$)
3: **repeat**
4: {to ensure that the 1-factor has no infinity-weight edges}
5: Find the minimum Weight perfect matching (1-factor) in the graph ($\overrightarrow{G}(H,E)$)
6: The 1-factors will be the schedule of a new rounds R in the incomplete schedule
7: Join the new round in the schedule S to get S.
8: Update the weight of the edge
9: **until** $R = |R|$
10: **return** A complete schedule S.

The Fig 2 show an example of 1-factor that represents the schedule of the round $R_1(t_1,t_5)(t_3,t_4)(t_6,t_2)$. The remaining rounds are created by finding the minimum weight perfect matching or 1-factor in the graph of the teams graph as shown in Algorithm 2. Fig 2 shows an example of the process of creating the remaining rounds of the incomplete schedule BPPT depicted in Fig1. In order to ensure a good diversity level among the population and encourage poor individual to be improved, children are mutated after being created by crossover operator. The mutation operator used in our case is the proposed Swap Rounds for$(\{R\} - \{BPPT\})$.

D. Variable neighborhood search improvement step

The overall E-GA for TTP is sketched in Algorithm 3.

Algorithm 3 : The E-GA algorithm for TTP.

1: Create an initial population P with $size\_pop$ individuals
2: **for** $iter = 1$ to $Max - iter$ **do**
3: Select two individuals from the population by selection operator
4: Apply the crossover operator on the two selected parents
5: Apply the Mutation operator on the two children
6: Apply VNS on the resulting solution
7: Replace the worst individuals of P with new offspring if it improves P.
8: Keep the best $P - elistm$ solutions to the new generation.
9: **end for**
10: **return** the best schedule S found.

To improve the quality of solutions, we apply VNS [11] as a local search meta-heuristic in each iteration. VNS is a systematic change of neighborhood combined within a local

search. We considered the three structures which are: Swap teams, Swap home and Swap round.

IV. Experimental results

The source codes are written in Java. The experiments were performed on a Intel(R) Core(TM)i5 4210UCPU @ (1.70 GHz) with 8 GB of RAM memory.

A. The considered benchmarks

We evaluated the proposed E-GA on three different data sets of instances available at the website [12]. We considered the most popular instances which are: the so-called NLx instances, the Constant distance instances, named CON_x (the x stands for the dimension of the instance or the number of the teams) and $CIRCx$: NLx instances based on real data of the US National Baseball League. NL_x instances are probably the most well-researched TTP benchmark- family and virtually all researchers studying the TTP publish their computational results with NLx instances. The constant distance instances CON_x are characterized by a distance of 1 between all teams. $CIRCx$ instances: all teams placed on a circle, with unit distances(distance of 1between 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 and it equals the minimum of i-j and j-i + n.

B. Parameters tuning

The adjustment of parameters of E-GA is fixed by an experimental study. The fixed 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.

The E-GA parameters are: the number of individuals in the population which is set to $n \cdot (n \cdot ln(n))$ where n is the number of teams, the mutation rate is set to 0.16 (i.e., on average, only 1 out of 6 schedules is mutated), the crossover rate is set to 1 (i.e., at each iteration the crossover operator is applied to all schedules ($size\_pop$)). The maximum number of iterations grows with the number of teams: $Max - iter = 1000 * n$.

We studied also the impact of BPPT length on the solution quality. We note that, the numerical results after 100 generations demonstrate that when $|BPPT| \cong 2n/3$, E-GA achieves the best mean results. We note also that the length of $|BPPT| \geq (n-1)+n/2$ may increase the risk of getting the same configuration.

C. The numerical results

Table III gives the results found by E-GA on the different considered instances of NL, CON and $CIRC$. The first column gives the name of instance, the second gives the lower bound for the considered instance, the third, the fourth and the fifth give the results found by E-GA method (respectively the minimum maximum and the average) of 3 runs, the column sixth and seventh and eighth give the running time (Minimum, maximum, Average). The last column give the gap between the best solution of our method and the Lower-bound [12].

$$GAP_{(E-GA,LB)} = \frac{Cost-travel_{LB} - Cost-travel_{E-GA}}{Cost-travel_{E-GA}} * 100$$

The proposed method succeeds in finding the optimal solution for $NL4$ and $NL6$, CON_4, CON_6, CON_8 , $CIRC4$ and $CIRC6$. For the rest of instances the gap is small and between 3.03 and 14,72%. However the gap between optimal solutions and our results in our strategies do not go down in general below 14,72%.

We observe that our results are close to Lower bound. In order to quantify this, we compute the performance ratio (PR) given by the formula $PR = \sum_{NBins}^{i=1} GAP_i/NBins$, where GAP_i is the gap between best solutions of our method of the instance i and its lower bound (using the Gap operator defined in formula IV-C), NBins is the number of instances for each benchmark.

The performance ratio for NL benchmark is equal to PR(NL)=3,01%. This result indicates that the proposed approach finds near optimal solutions with deviation from optimality equal to 2,93%, PR(CON)=0% and PR(CIRC)=4,43%. The general deviation from optimality value is 2,48%. The superiority of our method is due to the good combination of genetic algorithm operators with local search method. Moreover, the new crossover operator in GA increases the diversity of solutions with escaping from local optima. Further, the local search methods VNS explores more thoroughly the promising regions in search space.

TABLE IV. A comparative study for NL

Instance	**Our results**	AIS_{TTP} [13]	$CTSA$ [14]	AATTP[15]
NL4	**8276**	_	_	
NL6	**23916**	_	24579	
NL8	**39776**	39721	41265	47128
NL10	**61895**	62103	63337	69958

Instance	**Our results**	GA [16]	ACO [17]	SA [18]
NL4	**8276**	8276	8276	8276
NL6	**23916**	25908	23916	23916
NL8	**39776**	43112	39721	3921
NL10	**61895**	_	60399	59583

D. Further comparison with other techniques

In order to demonstrate the performance of our method in solving TTP, we conducted a further comparison with the following techniques for the NL class: AIS_{TTP} [13] (an immune-inspired algorithm based on the CLONALG framework); $CTSA$ [14] (a hybrid integer programming/constraint programming approach and a branch and price algorithm); GA [16] (Genetic Algorithm with Novel encoding scheme for representing a solution instance); $AATTP$ [15] (A 5.875-approximation Algorithm for TTP); ACO An ant colony optimization approach [17]; SA simulated annealing [18].

We notice that in the first family of instances NL (for NL4..Nl10). E-GA improves the results obtained by AIS_{TTP} with an average of 1.33% (using the gap operator defined in formula IV-C), Our approach is able to perform better than $CTSA$ results with an average equal to 2,85%, E-GA also gives better average than $AATTP$ with 13,51%, and improves the results of GA with an average of 3,94%, whereas our approach finds near solutions with deviation from ACO and SA equal to -0,63% and -1,28 respectively.

TABLE III. RESULTS FOUND BY (E-GA) ON THE NL, CON AND CIRC FAMILIES

Instance	LB	**Distance**			Time(secs)			Gap%
		Min	Aver	Max	Min	Aver	Max	
NL4	8276	8276	8276	8276	4,00	12,03	20,03	**0**%
NL6	22969	23916	24073	24101	25,22	45,80	80,06	**0**%
NL8	38760	39972	40619	41424	355,64	508,05	733,01	-3,03%
NL10	56506	61895	63142	69958	899,21	2010, 22	4023,05	-8,70%
CON4	17	17	17	17	2,01	8,08	15,34	**0**%
CON6	43	43	43	43	14,05	22,63	48,55	**0**%
CON8	80	80	80	81	238 ,85	603,42	1009,87	**0**%
CON10	124	124	125	130	784,2	932, 70	2847,50	**0** %
CIRC4	20	20	20	20	3,44	10,34	18,29	**0**%
CIRC6	64	64	64	64	28,12	60,20	104,05	**0**%
CIRC8	128	146	148	154	342 ,01	709,34	1088,54	-3,03%
CIRC10	220	280	288	294	911,26	1030, 40	5034,60	-14,72 %

V. CONCLUSION

In this paper we proposed an enhanced genetic algorithm (E-GA) for solving the well-known league scheduling problem NP-hard TTP sports. The proposed method (E-GA) starts with an initial population generated by using polygon method with intensification and diversification strategies. Then, it applies GA process with new crossover operator. This increased the diversity of solutions and permitted to explore efficiently the search space. Further, in order to improve the intensification mechanism of E-GA, we used VNS as an improvement step in GA. The overall method is evaluated on some datasets and compared with the optimal solutions. The results showed that E-GA is able to find optimal solutions for some instances and the general deviation from optimality equal to 2,48%. E-GA was compared also to other approaches. The results prove that our E-GA outperforms some interesting methods for TTP. This is due to the good combination of GA operators with VNS local search. It is important to note that our contribution is not just to achieve a good algorithm to solve TTP but also to build the remaining rounds of any incomplete schedule, which allows creating any schedules with fixing certain rounds. Indeed, our new crossover operator does not depend on a genetic algorithm approach. It can be used as an operator by other techniques or other heuristics. We plan to study the impact of our proposed crossover in other evolutionary methods.

REFERENCES

[1] A. C. B. Guedes and C. C. Ribeiro, "A heuristic for minimizing weighted carry-over effects in round robin tournaments," *Journal of Scheduling*, vol. 14, no. 6, pp. 655–667, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10951-011-0244-y

[2] K. Easton, G. Nemhauser, and M. Trick, *The Traveling Tournament Problem Description and Benchmarks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 580–584.

[3] C. Thielen and S. Westphal, "Complexity of the traveling tournament problem," *Theoretical Computer Science*, vol. 412, no. 4, pp. 345 – 351, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397510005451

[4] S. Irnich, "A new branch-and-price algorithm for the traveling tournament problem," *European Journal of Operational Research*, vol. 204, no. 2, pp. 218 – 228, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221709007929

[5] S. Westphal and K. Noparlik, "A 5.875-approximation for the traveling tournament problem," *Annals of Operations Research*, vol. 218, no. 1, pp. 347–360, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10479-012-1061-1

[6] M. Khelifa and D. Boughaci, "Hybrid harmony search combined with variable neighborhood search for the traveling tournament problem," in *International Conference on Computational Collective Intelligence*. Springer, 2016, pp. 520–530.

[7] M. Goerigk and S. Westphal, "A combined local search and integer programming approach to the traveling tournament problem," *Annals of Operations Research*, vol. 239, no. 1, pp. 343–354, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10479-014-1586-6

[8] J. R. Sampson, "Adaptation in natural and artificial systems (john h. holland)," *SIAM Review*, vol. 18, no. 3, pp. 529–530, 1976. [Online]. Available: http://dx.doi.org/10.1137/1018105

[9] D. de Werra, "Some models of graphs for scheduling sports competitions," *Discrete Applied Mathematics*, vol. 21, no. 1, pp. 47 – 65, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0166218X88900339

[10] D. Whitley, "The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best," in *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann, 1989, pp. 116–121.

[11] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, pp. 449–467, 2001.

[12] (1999) Challenge traveling tournament problems. [Online]. Available: http://mat.tepper.cmu.edu/TOURN/

[13] L. P. CCERES and M. C. RIFF, "Aisttp: An artificial immune algorithm to solve traveling tournament problems," *International Journal of Computational Intelligence and Applications*, vol. 11, no. 01, p. 1250008, 2012. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S1469026812500083

[14] R. V. Rasmussen and M. A. Trick, "The timetable constrained distance minimization problem," *Annals of Operations Research*, vol. 171, no. 1, p. 45, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10479-008-0384-4

[15] S. Westphal and K. Noparlik, "A 5.875-approximation for the traveling tournament problem," *Annals of Operations Research*, vol. 218, no. 1, pp. 347–360, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10479-012-1061-1

[16] D. N. S. Choubey, "A novel encoding scheme for traveling tournament problem using genetic algorithm," *IJCA Special Issue on Evolutionary Computation*, no. 2, pp. 79–82, 2010, full text available.

[17] D. C. Uthus, P. J. Riddle, and H. W. Guesgen, "An ant colony optimization approach to the traveling tournament problem," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 81–88.

[18] A. Anagnostopoulos, L. Michel, P. V. Hentenryck, and Y. Vergados, "A simulated annealing approach to the traveling tournament problem," *Journal of Scheduling*, vol. 9, no. 2, pp. 177–193, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10951-006-7187-8