Journal of Theoretical and Applied Information Technology
31st August 2020. Vol.98. No 16

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A SCATTER SEARCH HYBRID APPROACH FOR TEAM ORIENTEERING PROBLEM

[1]HAMZAH ALKHAZALEH, [2]MASRI AYOB, [3]AMER IBRAHIM, [4]NAHIL ABED, [5]MOHAMMAD HABLI, [6]TAWFIK SAID

IT Department, School of Engineering and Technology, Aldar University College, Dubai, UAE
[2]Faculty Of Information Science & Technology, Universiti Kebangsaan Malaysia (UKM), 43600 Bangi, Selangor, Darul Ehsan, Malaysia.
[1]hamzah@aldar.ac.ae, [2]masri@ukm.edu.my, [3]amer@aldar.ac.ae, [4]nahilabed@aldar.ac.ae, [5]m.habli@aldar.ac.ae, [6]tawfik.said@aldar.ac.ae

ABSTRACT

The Team Orienteering Problem (TOP) is a particular vehicle routing problem in which the aim is to search a fixed number of paths that maximize the scores associated with a set of given locations within a limited time. Scatter Search explores a search space of solutions systematically by evolving a small set of reference solutions. It has strategies for diversification (in diversification generation and subset generation methods); and intensification (in the improvement and updating method). However, all these methods are very time consuming. This paper proposes a scatter search hybrid approach (SSHA) to deal with the TOP by reduce processing time and maintaining a good set of references solutions in terms of diversity and quality. It uses some new operators, called reference set queen bee-method to initializing and updating the *RefSet*, and greedy select parents to selecting pairs from a reference set for the combination method to generate a new solution. Furthermore, to improve the quality of the solution, a local search is employed, called steepest descent to explore neighborhood in a fully deterministic manner and then selects the best neighbour. Experiments conducted on the standard benchmark of TOP clearly show that proposed approach outperforms the solving methods in the scientific literature. Our algorithm detects all but one of the best known solutions. A statistical test was conducted to determine the algorithm that performed better compared with the others. The results revealed that SSHA outperformed all state-of-the-art algorithms and was comparable to one algorithm.

Keywords: *Optimization, Metaheuristic, Team Orienteering Problem, Scatter Search Algorithm, Local Search.*

1. INTRODUCTION

The Team Orienteering Problem (TOP) is a development of the orienteering problem (OP). It is a well-known, challenging combinatorial optimization problem that was first highlighted and heuristically tackled by Butt and Cavalier [1]. The TOP introduced by Chao, et al. [2]. The TOP can be defined as a team consisting of several players (e.g., two, three, or four players). Each player should start at the same point and must visit the subset of points, which have their own scores, before reaching the same end point in a specific time. Once the team player visits a point, no other player can visit that particular point. In other words, each point can be visited only once. The objective is to maximize the total team scores that have been collected before it reaches the end point within the allowed time [2, 3]. The TOP has been considered an NP-hard problem [4] because it goes beyond and is even more difficult than the OP.

The applications of TOP include athlete recruiting [2], technician routing [3] and tourist trip planning [5]. In this paper, we are interested in TOP as the core variant of OP for multiple vehicles. TOP have attracted a good deal of attention in operational research and artificial intelligent community. Apart from the difficulty of solving it, TOP has been selected for two reasons: it represents various real-world applications and its results are still possible to be improved. So far different exact method have been developed for TOP proposed by Boussier, et al. [6], Butt and Ryan [7], Dang, et al. [8] and Nicola, et al. [9]. However, unless $P = NP$, there is no algorithm which can find an optimal solution within time polynomial in the size of customers. An alternative approach to the TOP is metaheuristic, which aims to yield a satisfactory solution within reasonable time.

In contrast to exact solving approaches, a number of heuristics and metaheuristics have been developed for TOP. To date, ant colony optimization (ACO)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[10]that constructs the candidate solutions by employing four constructive heuristics: sequential, deterministic–concurrent, random–concurrent, and simultaneous methods. guided local search [11], large neighborhood search with three improvement methods: local search improvement, shift and insertion improvement, and replacement improvement [12], memetic algorithm with optimal split procedures for chromosome evaluation (MA) [13], particle swarm optimization [14], path relinking [15], Pareto mimic algorithm [16], tabu search with an adaptive memory procedure [3], and variable neighborhood search [5] have been applied to the TOP. The interested reader is referred to [17] for a survey. Among the current metaheuristics, the Pareto mimic algorithm in [16] ranked first when testing on the instances of [2]. Moreover, the results on larger new generated instances demonstrate that their algorithm is very effective though it consumes relatively long running time.

This research aims to investigate the SS algorithm that may improve the available search approaches for the TOP. The main goal is to adapt the SS algorithm, utilize its strengths, cover its weaknesses by hybridizing it with other metaheuristic algorithms, and attain a suitable balance between exploration and exploitation.

In this paper, a metaheuristic, called scatter search hybrid algorithm, is proposed. The main contribution of this paper are summarized as follows: (1) it uses different selective strategies in subset generation method called, greedy selecting parents (GSP), to generate new solution by selecting suitable parents for solution combination method, and different updating strategies in references set update method called, references set queen bee (RefSetQB), to update the references set solution with new solution. (2) To hybridize the SS algorithm with other local search algorithm called, steepest decent (SD), to enhance the exploitation search. Experiments conducted on the standard benchmark of TOP clearly show that SSHA outperforms most of the existing solution methods of the literature. It detects all but one of the best known solutions. Moreover, a strict improvement was found for one instance of the benchmark. The remainder of this paper is structured as follows. Section 2 provides a description of the TOP. SSHA are described in Section 3. In Section 4, the experimental study of the parameters is discussed. The computational results on the standard benchmark is described in Section 5. Finally, in Section 6, some conclusions and further developments are discussed.

2. DESCRIPTION OF THE TOP

The TOP can be represented as a complete graph $G = (V, E)$, where $V = [1, 2,.., n+1]$ is a set of points and $E= [(i, j) \mid i, j \in V]$ is a set of edges. The aim of the TOP is to find m routes that start at point 0 and end at point $n+1$, such that the total score of the visited points is maximized. Each point can be visited only once. For each route, the total time taken to visit the points cannot exceed the predetermined time limit T_{max}. Let c_{ij} be the travel time of edge $(i, j) \in E$, T_i is the service time for point i, and s_i is the score of point i. $K = [1, 2,..,m]$ is a set of routes, $V' = [1, 2,.., n]$ is a set of points visited by the kth route, and U is a subset of V that is not included in the solution. $y_{ik} = 1$ if the ith point is visited in the kth route. Otherwise, $y_{ik} = 0$. $x_{ijk} = 1$ if the edge (i,j) is visited in the kth route and point j is visited directly after point i. Otherwise, $x_{ijk} = 0$. The evaluation function, $f(x_p)$ for solution x, x_p can be formulated as Equation (1). The TOP can be formulated as follows [2, 10, 18]:

$$f(x_p) = Max \sum_{k=1}^{m}\sum_{i=1}^{n} s_i y_{ik}, \quad (1)$$

Subjected to:

$$\sum_{i=0}^{n}\sum_{j=0}^{n+1} c_{ij} x_{ijk} + \sum_{i=1}^{n} T_i y_{ik} \le T_{max}, \quad \forall k \in K \quad (2)$$

$$x_{ijk} \in X \subseteq \{0, 1\} \quad (3)$$

Where constraint (2) is the time limit constraint, which indicates that the total traveling time of each route must not exceed the T_{max}. X is a decision variable used in constraints (4)–(9).

$$\sum_{j=0}^{n} x_{jik} = \sum_{j=1}^{n+1} x_{ijk}, \quad \forall i \in V', k \in K \quad (4)$$

$$\sum_{j=1}^{n+1} x_{0jk} = 1, \quad \forall k \in K \quad (5)$$

$$\sum_{i=0}^{n} x_{ilk} = 1, \quad l = n + 1; \ \forall k \in K \quad (6)$$

$$\sum_{i=1}^{n} x_{ijk} = y_{ik}, \quad \forall i \in V', \forall k \in K \quad (7)$$

$$\sum_{k=1}^{m} y_{ik} \le 1, \quad \forall i \in V' \quad (8)$$

$$\sum_{i,j \in U} y_{ik} \le |U|-1, \quad U \text{ is an arbitrary subset of } V'; 2 \le |U| \le n; \forall k \in K \quad (9)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$y_{ik}, x_{ijk} \in \{0, 1\}$ (10)

Constraint (4) guarantees the connectivity of each route. Constraints (5) and (6) ensure that the route will start at point 0 and end at n+1. Constraint (7) describes the relation between x and y. Constraint (8) ensures that each vertex is visited only by one route. Constraint (9) imposes restrictions on the variables.

3. A SCATTER SEARCH HYBRID ALGORITHM

The SS is a population-based metaheuristic introduced by Glover [19]. It constructs the solutions by combining other elite solutions to exploit their positive attributes. The main aspect of the mechanisms within SS are not restricted to single uniform design allows the exploration of strategic possibilities that may prove effective in particular implementation [20, 21].

The search process of the SS algorithm starts with constructing a number of trial diverse solutions as an input population. Each solution in the population pool is enhanced using the improvement method. After initializing the population, a small set of good solutions is selected from the population to build a small population called a reference set, where the "good" criteria are based on the objective function value and diversity value of the solutions. After building a reference set, new subsets are generated using the subset generation method by selecting the solutions from the reference set pool in lexicographical order. Each subset of solutions is combined in a systematic way to produce one or more new solutions, each of which is mapped into an associated feasible solution. Each new solution is enhanced by the improvement method. The reference set pool is updated by a new trial solution using a steady-state replacement mechanism. This step is known as a dynamic update of the search experience [22, 23]. The pseudo-code of the SS is illustrated in Algorithm 1.

Algorithm 1. The main procedure pseudo-code of the SS

Set $P = \emptyset$, $|P| = PSize = 50$.
While (until $|P| = PSize$)
  Step 1: Use the *Diversification Generation Method* to construct a population of solutions.
  Step 2: Apply the *Improvement Method*. Let χ be the resulting solution. If $\chi \notin P$ then add χ to P, otherwise, discard χ.
End While
  // employ nearest neighbour to construct the population of initial solutions.
  // utilize steepest decent to improve all the solutions in the population.
  Step 3: use the *Reference Set Update method* to build *RefSet*, $|RefSet| = 20$, $RefSet = [X^1, \dots, X^b], [Y^1, \dots, Y^d]$with the "best" b_1 and "diverse" b_2 solutions in P. Order the solutions in b_1 according to their objective function value such that χ^1 is the best solution and χ^b the worst.
While (StoppingCriterion)
  Step 4: Generate *NewSubsets* by the *Subset Generation Method*.
    // apply greedy selecting parents to generate the subset.
  While (*NewSubsets* = Ø) *// this is repeated until all NewSubsets are combined.*
    Step 5: select the *Subset s* in lexicographical order to be combined.
    Step 6: Apply the *Solution Combination Method* to *NewSubset s* to obtain one new solution χ'.
      // share 25% of the divers solution Y^d attribute in the quality solution X^b to produce χ'.
      // convert the infeasible solution to feasible using repair strategy.
    Step 7: Apply the *Improvement Method* to the new solution χ'.
      // utilize steepest decent to improve the produced new solution from Solution Combination Method.
    Step 8: apply the *Reference Set Update method* to update *RefSet*.
      // utilize references set queen bee strategy.
    Remove *s* from the *Subsets*
  End While
  If (*RefSet* has not updated)
    Apply *Diversification Generation Method* to initial new population of solutions

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

// performing nearest neighbour.
Apply the *Improvement Method* to the new solutions in the population.
// apply steepest decent.
Replace the diverse solutions in the *RefSet* by the new solutions in the population.
End If
End While
Step 9: Return the best found solution

The reference set pool is exploited to avoid the search from redundantly exploring the solutions that have been evaluated and prevent the duplication of solutions in the pool. The structure of reference set pool stimulates search diversification and intensification and may help the search to escape from the local optima because of the types of solutions in the pool and the updating strategy [24, 25].

Hence, the SS concentrates on maintaining the balance between intensification and diversification of the search. This balance is achieved by combining all possible pairs in the reference set pool to generate one or more new solutions, where the new solution may contain attributes that are not available in the original pairs. The SS exploits knowledge derived from the search space and is thus considered an information-derived algorithm [26].

The elements in the SS can be implemented in a variety of ways and degree of sophistication because of the flexibility of the SS framework [21, 22, 25]. For example, diversification generation method step might apply several constructive heuristic to build the population. Moreover, subset generation method can utilize different selection mechanism to select the parent for combination method [21].

A SS template consists of five procedures, as described by Glover, et al. [22], Glover, et al. [23], and Laguna and Martí [27]. Further details on the algorithm are given in the following subsections.

3.1 A Diversification Generation Method

Initializing the population using a diversification generation method plays an important role in developing the method that balances diversification and intensification [28]. The population generation method aims to generate good starting solutions to solve the TOP. The collection of initial solutions are constructed uniformly and distributed in the search space as input. This operation starts from scratch (empty solution) and constructs a solution by assigning points to one decision variable at a time until a complete solution is generated. As shown in Algorithm 1, a large number of diverse solutions are constructed using the nearest-neighbor greedy algorithm (NNGA) to obtain a diverse region in the solution space [29]. All the generated solutions considered in this work are feasible.

3.2 An Improvement Method

In SS, an improvement method is introduced to transform a trial solution into one or more enhanced trial solutions. Thus, this component is used to enhance the generated solution quality via a local search procedure that drives the solution to the local optima [22]. The aim of this operation is to explore the neighborhood of the generated solutions by modifying it [21]. The solutions will be accepted if they are still feasible. We employed five common neighborhood structures (i.e. insert, swap, move, 2-opt, and replace), all of which were drawn from literature [11]. These neighborhood structures aims to shorten the total travel time and increase the total received score. The neighbors that improve the current solution are randomly selected at each iteration. We use a steepest descent heuristic as the improvement method to search for a better quality solution. We consecutively use a number of iterations as a termination criterion.

3.2.1 Steepest Descent (SD)

SD explores neighborhood *N* in a fully deterministic manner and then selects the best neighbor (solution) [30, 31]. Hence, SD exhaustively explores the neighborhood *N* of solution *x*, and all probable moves are tried for solution *x* to return the best neighboring (solution x'). In this work, SD is applied after solution *x* is generated (*x* is generated by the diversification generation or the combination method) to further improve its quality. Each of the five neighborhood *N* structures is selected randomly and explored for each solution *x* to return the one with the highest quality (x'). SD is terminated when it reaches the maximum number of iterations *NI*. The pseudo-code of the best improvement hill climbing (steepest descent) adopted in this work is presented in Algorithm 2.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm 2. Steepest Descent

x = Initial solution;
NI = number of iteration;
While (stopping criterion is not met)
    Choose neighborhood $N()$ randomly;
    Select best neighbor x' where $x' \in N(x)$;\\ x'new solution;
    If $f(x') > f(x)$ **then** \\ $f(x)$the objective function value of solution x
        $x = x'$
    End if
    End while

3.3 A reference set update method

The reference set is the heart of a SS procedure, which may result in important modifications during the search process because of its initial composition [21]. The initial *RefSet* is built by selecting the elite solutions from the population and inserting them into the *RefSet*. These elite solutions are selected based on the objective value (solution quality) and diversity value (solution diversity). The diversity value is calculated by diversity measurement method, for more details see [32]. The *RefSet* is divided into two sets of solutions, namely, $RefSet_1$, which contains high-quality solutions; and $RefSet_2$, which contains the most diverse solutions. Moreover, the reference set update method is employed to update the *RefSet* after generating the new trial solution x through the combination and improvement methods. In this work, a queen bee strategy is used to build and update the reference set.

3.3.1 Reference set queen bee-method (RefSet-QB)

The reference set queen bee-method idea came from the bee colony optimization algorithm [31]. This method consists of initializing and updating the *RefSet*. Initializing the *RefSet* starts with selecting the 10 best solutions from the population based on their objective function values. Then, the 10 most diverse solutions are selected from the population based on their diversity values. All the selected solutions (20 solutions) are placed and sorted in one pool (*RefSet*). The best solution in the initial *RefSet* is marked as the queen bee. Algorithm 3 illustrates the initialization step of the queen bee method.

Algorithm 3. Initial RefSetQB

Set *RefSet* = Ø, | *RefSet* | = *RefSet_Size* = 20.
Step 1: sort the population p of solutions based in the objective function $f(x)$.
Step 2: chose the better 10 quality solutions and put them in the *RefSet*.
Step 3: sort the population p of solutions based in the diversity vlaue $d(x)$.
Step 4: chose the better 10 diverse solutions and put them in the *RefSet*.
Step 5: sort the *RefSet* based on the objective function $f(x)$.
Return *RefSet* = $[X^1, \dots, X^n]$

Meanwhile, the *RefSet* is updated by comparing the new trial solution (x') with the worst quality solution (x_b) in the *RefSet*. If the new trial solution (x') is better in terms of the objective function value, it will supersede the worst quality solution (x_b). Once the *RefSet* cannot be updated anymore, the new trial solution (x') will supersede the worst diverse solution (x_y) if it is better in terms of the diversity value. When the updating strategy fails to update the *RefSet*, the solutions in the *RefSet,* except for the queen bee (best solution in the *RefSet*), are replaced by the other solutions generated by the diversification generation method. Algorithm 4 illustrates the updating step of the queen bee method.

Algorithm 4. Update RefSetQB

Now_Sol_List = $[X^1, \dots, X^n]$; //*new trial solutions generated by combination method and improved by improvement method.*
While (*Now_Sol_List* = Ø)
    x_b = worst *quality* solution in the *RefSet* based on the objective function value.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

x_y = worst *diversity* solution in the *RefSet* based on the diversity value.
If ($f(x') > f(x_b)$) // *f(x') objective function of new trial solutions*;
replace the worst quality solution x_b in *RefSet* by x'.
remove x' from *Now_Sol_List*.
Else If ($d(x') > d(x_y)$) // *d(x') diversity value of new solutions to RefSet;*
replace the worst *diversity* solution x_y in *RefSet* by x'.
remove x' from *Now_Sol_List*.
Else
remove x' from *Now_Sol_List*.
End If
End While
If (*RefSet* has not updated)
Apply *Diversification Generation Method* to initial new population of solutions by performing nearest neighbour.
Apply the *Improvement Method* to the new solutions in the population.
// *apply hill climbing.*
Replace the *solutions except queen bee* in the *RefSet* by the new trial solutions in the population.
End If

3.4 A subset generation method

Subset generation method (SGM) plays an important role in selecting pairs from a reference set for the combination method to generate a new solution [23]. The most common subset generation method is to generate the subset of pairs using a lexicographical order mechanism, namely, *Type-I* selection (i.e., the size of all subsets is 2). In the present work, greedy select parents (GSP) is employed as selection mechanism to generate a subset of pairs for the combination method.

3.4.1 Greedy select parents (GSP)

This selection strategy operates on two lists of quality and diverse solutions in the reference set to generate a subset of two solutions. This method generates three types of subsets: two quality solutions, two diversity solutions, and one quality and one diversity solution [33]. Each solution is elected randomly from each list once. The pseudo-code of this strategy is illustrated in Algorithm 5.

Algorithm 5. Greedy select parents

TempListX = *RefSet* = $[X^1, \ldots, X^b]$
TempListY = *RefSet* = $[Y^1, \ldots, Y^d]$
SubsetList = Ø.
Subset S = Ø, $|S|$ = *SSize* = 2.
While (*TempList is empty*)
Generate empty subset S;
While (until $|S|$= *SSize*)
Switch :
case 1
Step 1: select x randomly from *TempListX*; // *x is solution quality*
Step 2: add x to S;
Step 3: remove x from *TempListX*;
case 2
Step 1: select y randomly from *TempListY*;//*y is solution diversity*
Step 2: add y to S;
Step 3: remove y from *TempListY*;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
                case 3
                        apply case 1 & case 2;

    Step 5: add S to SubsetList
    End while
    Step 6: return SubsetList
```

3.5 A solution combination method

The main goal of this component is to generate one or more trial solutions from various regions of the solution search space. These new trial solutions are subjected to the improvement method. In this study, we use the work of Martí, et al. [34] to design the combination method, which produces one trial solution in a systematic way. The solution in the subset with the highest quality is selected, and then 25% is taken from other solutions with the highest quality. The new solution has 75% of the quality solution attributes and 25% percent of the diverse solution attributes and is perhaps infeasible.

Repair strategy is applied on the infeasible solution to restore its feasibility by removing the repeated points and the lowest score so that the solution does not exceed the time limit. For instance, a strategy is applied in the case where the search operators used by the combination method may generate infeasible solutions.

The SS features the role of the reference set update method in maintaining elite solutions (in terms of solutions of high quality and high diversity) and the SGM in providing pairs for producing a new solution. A greedy select parents (GSP) is employed as selection mechanism to generate a subset of pairs in SGM which considers both quality and diversity solutions in the *RefSet* to be selected as a subset during the search process and produces new solutions that maintain the diversity of the *RefSet*. Steepest Descent is single-solution-based metaheuristics can improve the quality of the obtained results further by exploiting the promising area.

In this work, the repairing strategy first checks the repeated points. If the point is repeated in the solution, one of these points is randomly selected and then removed from the others. In the second step, the time budget for each route is examined. If the solution exceeds the available time, the edge that causes a long travel time between points is removed. In the next section, the performance of the SSHA on the standard benchmark for TOP is discussed.

4. EXPERIMENTAL SETUP

We have conducted an experimental study to evaluate our proposals and to compare the algorithm, termed SSHA, with the state of the art. In this study, we have considered the standard benchmark dataset most widely used in the literature, namely, team orienteering problem [2], making a total of 387 instances with different sizes and flexibility (the number of points, time limit, and number of player).

The SS is coded using Java 1.7 and performed on a personal computer (Intel Pentium (R) Core i5 CPU at 3.40 GHz with 4 gigabyte RAM), running on Windows 8 operating system (64-bit). The proposed SS executed 31 independent runs with different random seeds for some instances of the TOP to assess the effectiveness of using different parameters. Talbi [31] recommended performing minimum 30 runs in the statistical analysis of the algorithm performance. Executing 31 runs can easily calculate the median value without the need for interpolation.

4.1 Parameter settings

There are three parameters in SSHA, that is, the population size (*Popsize*), the number of iteration ($Iter_{max}$) based on the fix number (k), and references set size (*RefSet*). The parameter values are determined using statistical study. The tested values of these parameters are as follows: *Popsize* = [30; 50; 100], k = [10; 20; 30; 40; 50], and *RefSet size* = [6; 10; 20]. Based on experimental results, we observed that when *Popsize* = 20, k = 40, and *RefSet size* = 20, SSHA works best. To analyze the effect of each parameter, only one parameter is varied while others are fixed.

The settings of the parameters of the SS HA used for the TOP are presented in Table 1. We performed trial

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

and error methods called offline parameter tuning [31].

Table 1. Parameter Settings

#	Parameters	Values
1-	Population size (Pop_{size})	50 solutions
2-	References set size ($RefSet$)	20 solutions (b_1 = 10, b_2 = 10)
3-	References set quality ($RefSet_1$)	b_1 = 10
4-	References set diversity ($RefSet_2$)	b_2 = 10
5-	Number of Iteration ($Iter_{max}$)	$Iter_{max}$ =k . n/m; k = 30
6-	Subset generation size	2 solutions in each subset
7-	Solution Combination rate	1.0

We set the termination condition for the SS to solve the TOP according to the work of Dang, et al. [14]. The SSHA stops when the number of elapsed consecutive iterations reaches the obtained results of Equation (10).

$$Iter_{max} = k\,.\,n/m \qquad (10)$$

where k is the fixed number, n is the number of points, and m is the number of paths. The termination condition is changed based on the number of points and number of paths.

5. RESULTS AND COMPARISON

This section is divided into two subsections. The first subsection presents the assess the value of incorporating the Steepest Descent (SD), Reference set queen bee-method (RefSet-QB), and Greedy select parents (GSP) with the proposed algorithm. The second subsection compares the results of proposed algorithm against the other algorithms that have been introduced in the literature for TOP.

5.1 Effectiveness Evaluation

This experiment examines the value of using of incorporating the Steepest Descent (SD), Reference set queen bee-method (RefSet-QB), and Greedy select parents (GSP) with SS. We have tested the proposed algorithm (denoted as SS) with RefSet-QB (denoted as SS-RefSetQB), GSP (denoted as SS-GSP), both together (denoted ISS) and incorporated ISS with SD (denoted as HISS-SD). To ensure a fair comparison between SS, SS-RefSetQB, SS-GSP, ISS, and HISS-SD the initial solution, number of runs, stopping condition and computer resources are the same for all instances. All algorithms have been tested 31 runs over all instances. Table 2 lists, for each instance, the maximum (*Max*), average (*Avg*), standard deviation (*Stdv*), and CPU average time (CPU_{Avg} *Time*) achieved by SS, SS-RefSetQB, SS-GSP, ISS, and HISS-SD. In the table, the maximum value is preferred (Maximization problem) and best results achieved by the compared algorithms are presented in boldfont.

From table 2, it can be notice that HISS-SD obtained the best results in all instance and outperformed the other tested algorithms, except ISS, across most instances. Moreover, both the average and standard deviation results gained by HISS-SD are better than other tested algorithms on all instances. The results revel that the cooperative SS with RefSet-QB, GSP, and SD contribute to HISS-SD performance and has an effect on the ability of HISS-SD in producing high quality and consistent results across all tested instances.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2. The computational results of SS, SS-RefSetQB, SS-GSP, ISS, and HISS-SD

		Ins. Name											
Algorithms		**p4.2.l**	**p4.3.p**	**p4.4.s**	**p5.2.y**	**p5.3.r**	**p5.4.v**	**p6.2.n**	**p6.3.n**	**p6.4.l**	**p7.2.o**	**p7.3.s**	**p7.4.l**
SS	***Max***	1071	**1222**	1254	**1645**	1125	**1320**	**1260**	**1170**	**696**	**945**	1079	**590**
	Avg	1064	1200.06	1225.84	1621.61	1124.03	**1320**	1245.1	1167.68	**696**	935.71	1073.35	589.52
	Stdv	7.19	16.32	18.5	12.27	2.01	0	11.26	4.82	0	7.51	5.39	1.98
	CPU_{Avg} Time	269.37	248.72	185.79	172.4	103.56	53.51	109.87	104.12	46.01	336.83	229.75	91.57
SS-GST	***Max***	**1074**	**1222**	1256	**1645**	**1125**	**1320**	**1260**	**1170**	**696**	**945**	**1081**	**590**
	Avg	1070.19	1208.77	1235.81	1635	**1124.84**	**1320**	1253.23	1168.84	**696**	941.23	1074.97	589.81
	Stdv	2.51	13.76	14.66	8.76	0.9	0	8.73	3.61	0	6.17	4.92	0.6
	CPU_{Avg} Time	199.12	105.26	69.75	138.45	90.3	**42.34**	86.73	98.85	**38.15**	219.1	**160.71**	78.13
SS-RefSetQB	***Max***	1073	**1222**	1256	**1645**	**1125**	**1320**	**1260**	**1170**	**696**	**945**	**1081**	**590**
	Avg	1067	1206.94	1225.77	1629.19	1122.9	**1320**	1246.26	1168.65	**696**	939.94	1070.68	589.48
	Stdv	4.95	18.44	15.71	9.58	5.59	0	9.32	3.7	0	4.63	3.91	2.53
	CPU_{Avg} Time	200.01	143.37	129.25	**134.59**	90.46	45.3	**80.51**	94.29	42.7	254.85	189.19	**71.79**
ISS	***Max***	**1074**	**1222**	1257	**1645**	**1125**	**1320**	**1260**	**1170**	**696**	**945**	**1081**	**590**
	Avg	1071.06	1215.16	1245.16	1635.65	1124.35	1320	1254.58	1168.89	**696**	942.61	1076.61	589.94
	Stdv	3.93	11.92	6.94	8.44	1.87	0	6.9	3.96	0	3.33	5.03	0.36
	CPU_{Avg} Time	**173.47**	**93.84**	**66.8**	137.04	**90.29**	46.04	**89.3**	**91.5**	41.2	257.05	186.82	78.32
HISS-SD	***Max***	**1074**	**1222**	**1260**	**1645**	**1125**	**1320**	**1260**	**1170**	**696**	**945**	**1081**	**590**
	Avg	**1073.48**	**1221.68**	**1257**	**1645**	**1125**	**1320**	**1260**	**1170**	**696**	**945**	**1081**	**590**
	Stdv	**0.51**	**0.54**	**3.44**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	CPU_{Avg} Time	195.49	107.07	86.32	152.8	99.64	50.12	95.11	96.61	42.62	285.95	197.45	83.3

To farther support these positive results, the significant difference between the proposed algorithms were examined by performing a Wilcoxon rank test with the significance interval of 95% ($\alpha = 0.05$). Pair comparisons were executed for HISS-SD and the other proposed algorithms. Table 3 presents the *p*-value of the HISS-SD versus SS, SS-RefSetQB, SS-GSP, and ISS for the TOP instances. In Table 3, the symbol "+" denotes that HISS-SD is statistically better than the contending proposed algorithm (p-value < 0.05), "–" denotes that HISS-SD was outperformed by the contending proposed algorithm (p-value > 0.05), and "~" denotes that HISS-SD has the same performance as the contending proposed algorithm (p-value = 0.05).

Table 3. The *p-value* of HISS-SD versus SS, SS-RefSetQB, SS-GSP, and ISS.

HISS-SD vs.	**SS**	**SS-GST**	**SS-RefSetQB**	**ISS**
Ins. Name	*p-value*	*p-value*	*p-value*	*p-value*
p4.2.l	+	+	+	+
p4.3.p	+	+	+	+
p4.4.s	+	+	+	+
p5.2.y	+	+	+	+
p5.3.r	~	~	+	~
p5.4.v	~	~	~	~
p6.2.n	+	+	+	+
p6.3.n	+	~	~	+
p6.4.l	~	~	~	~
p7.2.o	+	+	+	+
p7.3.s	+	+	+	+
p7.4.l	~	~	~	~

The results in Table 3 show that HISS-SD is statistically better than the contending other proposed algorithms in most instances. These results support the fact that the SD algorithm improved the performance of the SS, SS-RefSetQB, SS-GSP, and ISS, while the HISS-SD obtained much better results than the other proposed algorithms. Indeed, the use the SD algorithm can effectively enhance the search performance to obtain very good results for all tested instances.

5.2 The comparison of HISS-SD with other algorithms for TOP.

This section compares the performance of the proposed algorithms with state-of-the-art of algorithms used to solve team orienteering problem (TOP). The results of the proposed algorithms for TOP are compared with state-of-the-art algorithms in terms of maximum obtained value, average CPU time, relative percentage error (RPE), and average RPE (ARPE). A statistical test is also performed to compare the proposed algorithms with other algorithms implemented to solve TOP.

The state-of-the-art algorithms used to compare the results are presented in Table 4. The symbols of these algorithms, including the descriptions and references are presented in Table 4.

The comparison of 157 instances with the standard benchmark for TOP, which are set as 4, 5, 6, and 7, is reported. Other instances (i.e., sets 1, 2, and 3) are eliminated from the comparison because all the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

compared algorithms obtain the same best known results.

Tables 5 – 9 present the results obtained through state-of-the-art of algorithms for all compared instances and through the proposed algorithms. In addition, a comparison is also performed with respect to the best known results for TOP. The best results are displayed in boldfont and with a dark background.

Table 4: Abbreviations of the compared methods for TOP.

#	Symbol	Descriptions	References
1-	ASe	Sequential Ant Colony Optimization	(Ke et al. 2008)
2-	ADC	Deterministic Concurrent Ant Colony Optimization	(Ke et al. 2008)
3-	ARC	Random Concurrent Ant Colony Optimization	(Ke et al. 2008).
4-	ASi	Simultaneous Ant Colony Optimization	(Ke et al. 2008)
5-	MA10	Memetic Algorithm	(Bouly et al. 2010)
6-	PSOMA	Particle Swarm Optimization Based Memetic Algorithm	(Dang et al. 2013)
7-	PSOiA	Particle Swarm Optimization Inspired Algorithm	(Dang et al. 2013)
8-	PMA	Pareto Mimic Algorithm	(Ke, Zhai et al. 2016)

According to the results in Tables 5 – 9, SS, ISS, and HISS-SD generate excellent results for most instances of TOP. HISS-SD obtains the best results compared with the proposed algorithms in 153 out of 157 instances. This result indicates that this algorithm generalizes well among all TOP instances instead of producing excellent results for few instances only.

Table 5. Best results of SS, ISS, and HISS-SD compared with other algorithms for test set 4

Ins.	*HISS-SD*	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*	*Best Known*
p4.2.a	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**
p4.2.b	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**
p4.2.c	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**
p4.2.d	**531**	**531**	**531**	**531**	**531**	530	**531**	**531**	**531**	**531**	**531**	**531**
p4.2.e	**618**	**618**	**618**	**618**	600	600	613	**618**	**618**	**618**	**618**	**618**
p4.2.f	**687**	**687**	**687**	**687**	672	672	672	**687**	**687**	**687**	**687**	**687**
p4.2.g	**757**	756	756	**757**	756	756	756	**757**	**757**	**757**	**757**	**757**
p4.2.h	**835**	834	**835**	827	819	819	820	**835**	**835**	**835**	**835**	**835**
p4.2.i	**918**	**918**	**918**	**918**	900	**918**	**918**	**918**	**918**	**918**	**918**	**918**
p4.2.j	**965**	962	**965**	**965**	962	962	962	**965**	**965**	**965**	**965**	**965**
p4.2.k	**1022**	**1022**	**1022**	**1022**	1016	1016	1016	**1022**	**1022**	**1022**	**1022**	**1022**
p4.2.l	**1074**	**1074**	**1074**	1071	1070	1071	1069	1071	1071	**1074**	**1074**	**1074**
p4.2.m	**1132**	**1132**	**1132**	1130	1115	1119	1113	**1132**	**1132**	**1132**	**1132**	**1132**
p4.2.n	**1174**	1173	1172	1168	1149	1158	1169	**1174**	**1174**	**1174**	**1174**	**1174**
p4.2.o	**1218**	1216	1206	1215	1209	1198	1210	**1218**	**1218**	**1218**	**1218**	**1218**
p4.2.p	**1242**	1241	1239	**1242**	1229	1233	1239	**1242**	1241	**1242**	**1242**	**1242**
p4.2.q	1267	1264	1258	1263	1253	1252	1260	1267	1267	**1268**	**1268**	**1268**
p4.2.r	**1292**	1288	1276	1288	1278	1278	1279	**1292**	**1292**	**1292**	**1292**	**1292**
p4.2.s	**1304**	**1304**	1290	**1304**	**1304**	1303	**1304**	**1304**	**1304**	**1304**	**1304**	**1304**
p4.2.t	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**
p4.3.c	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**
p4.3.d	**335**	**335**	**335**	**335**	333	333	**335**	**335**	**335**	**335**	**335**	**335**
p4.3.e	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**
p4.3.f	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**
p4.3.g	**653**	**653**	**653**	**653**	652	**653**	652	**653**	**653**	**653**	**653**	**653**
p4.3.h	**729**	**729**	**729**	720	713	713	713	728	**729**	**729**	**729**	**729**
p4.3.i	**809**	**809**	**809**	796	793	793	786	**809**	**809**	**809**	**809**	**809**
p4.3.j	**861**	**861**	860	**861**	857	855	858	**861**	**861**	**861**	**861**	**861**
p4.3.k	**919**	**919**	**919**	918	913	910	910	**919**	**919**	**919**	**919**	**919**
p4.3.l	**979**	978	**979**	**979**	958	976	966	**979**	**979**	**979**	**979**	**979**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

p4.3.m	**1063**	1056	1061	1053	1039	1028	1046	**1063**	**1063**	**1063**	**1063**	**1063**
p4.3.n	**1121**	**1121**	**1121**	**1121**	1109	1112	1103	**1121**	**1121**	**1121**	**1121**	**1121**
p4.3.o	**1172**	**1172**	1170	1170	1163	1167	1165	**1172**	**1172**	**1172**	**1172**	**1172**
p4.3.p	**1222**	**1222**	**1222**	1221	1202	1207	1207	**1222**	**1222**	**1222**	**1222**	**1222**
p4.3.q	**1253**	1246	1251	1252	123	1239	1238	**1253**	**1253**	**1253**	**1253**	**1253**
p4.3.r	**1273**	1272	1262	1267	1263	1263	1263	**1273**	**1273**	**1273**	**1273**	**1273**
p4.3.s	**1295**	1293	1282	1293	1291	1289	1291	**1295**	**1295**	**1295**	**1295**	**1295**
p4.3.t	**1305**	**1305**	1299	**1305**	1304	1303	1304	**1305**	1304	**1305**	**1305**	**1305**
p4.4.e	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**
p4.4.f	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**
p4.4.g	**461**	**461**	**461**	**461**	**461**	460	460	**461**	**461**	**461**	**461**	**461**
p4.4.h	**571**	**571**	**571**	**571**	556	556	556	**571**	**571**	**571**	**571**	**571**
p4.4.i	**657**	**657**	**657**	**657**	653	653	653	**657**	**657**	**657**	**657**	**657**
p4.4.j	**732**	**732**	**732**	**732**	731	731	731	**732**	**732**	**732**	**732**	**732**
p4.4.k	**821**	820	**821**	**821**	820	818	818	**821**	**821**	**821**	**821**	**821**
p4.4.l	**880**	878	878	**880**	877	875	875	**880**	**880**	**880**	**880**	**880**
p4.4.m	**919**	916	**919**	918	911	911	911	916	**919**	**919**	**919**	**919**
p4.4.n	976	976	967	961	956	956	956	969	969	976	976	**977**
p4.4.o	**1061**	**1061**	**1061**	1036	1030	1029	1029	**1061**	**1061**	**1061**	**1061**	**1061**
p4.4.p	**1124**	1120	**1124**	1111	1108	1110	1110	**1124**	**1124**	**1124**	**1124**	**1124**
p4.4.q	**1161**	1160	1157	1145	1150	1148	1148	**1161**	**1161**	**1161**	**1161**	**1161**
p4.4.r	**1216**	1213	1211	1200	1195	1194	1194	**1216**	**1216**	**1216**	**1216**	**1216**
p4.4.s	**1260**	1257	1256	1249	1256	1252	1252	**1260**	1259	**1260**	**1260**	**1260**
p4.4.t	**1285**	1284	1282	1281	1281	1281	1281	**1285**	**1285**	**1285**	**1285**	**1285**

Table 6. Best results of SS, ISS, and HISS-SD compared with other algorithms for test set 5.

Ins.	*HISS-SD*	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*	*Best Known*
p5.2.h	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**
p5.2.j	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**
p5.2.k	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**
p5.2.l	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**
p5.2.m	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**
p5.2.n	**925**	**925**	**925**	**925**	920	920	**925**	**925**	**925**	**925**	**925**	**925**
p5.2.o	**1020**	**1020**	**1020**	**1020**	**1020**	1010	1010	**1020**	**1020**	**1020**	**1020**	**1020**
p5.2.p	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**
p5.2.q	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**
p5.2.r	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**
p5.2.s	**1340**	**1340**	1330	**1340**	1330	1330	1330	1330	**1340**	**1340**	**1340**	**1340**
p5.2.t	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**
p5.2.u	**1460**	**1460**	1430	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**
p5.2.v	**1505**	**1505**	1490	**1505**	1495	1500	1495	**1505**	**1505**	**1505**	**1505**	**1505**
p5.2.w	**1565**	1560	1550	1560	1555	1555	1555	1560	1560	**1565**	**1565**	**1565**
p5.2.x	**1610**	**1610**	1595	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**
p5.2.y	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**
p5.2.z	**1680**	**1680**	1670	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**
p5.3.k	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**
p5.3.l	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**
p5.3.n	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**
p5.3.o	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**
p5.3.q	**1070**	**1070**	**1070**	**1070**	1065	1065	1065	**1070**	**1070**	**1070**	**1070**	**1070**
p5.3.r	**1125**	**1125**	**1125**	**1125**	1120	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**
p5.3.s	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	1185	**1190**	**1190**	**1190**	**1190**	**1190**
p5.3.t	**1260**	**1260**	**1260**	**1260**	1250	1255	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**
p5.3.u	**1345**	**1345**	**1345**	**1345**	1330	1335	1335	**1345**	**1345**	**1345**	**1345**	**1345**
p5.3.v	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	1420	**1425**	**1425**	**1425**	**1425**	**1425**
p5.3.w	**1485**	**1485**	1475	**1485**	1465	1465	1465	**1485**	**1485**	**1485**	**1485**	**1485**
p5.3.x	**1555**	**1555**	1540	1540	1535	1540	1540	**1555**	**1555**	**1555**	**1555**	**1555**
p5.3.y	**1595**	**1595**	1580	1590	1590	1590	1590	1590	**1595**	**1595**	**1595**	**1595**
p5.3.z	**1635**	**1635**	1630	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

p5.4.m	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**
p5.4.o	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**
p5.4.p	**765**	**765**	**765**	**765**	760	760	760	760	**765**	**765**	**765**	**765**
p5.4.q	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**
p5.4.r	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**
p5.4.s	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**
p5.4.t	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**
p5.4.u	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**
p5.4.v	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**
p5.4.w	**1390**	**1390**	**1390**	**1390**	1380	**1390**	1380	1380	1385	**1390**	**1390**	**1390**
p5.4.x	**1450**	1445	1445	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**
p5.4.y	**1520**	**1520**	**1520**	**1520**	1510	1510	1500	**1520**	**1520**	**1520**	**1520**	**1520**
p5.4.z	**1620**	**1620**	1595	**1620**	**1620**	1575	1580	**1620**	**1620**	**1620**	**1620**	**1620**

Table 7. Best results of SS, ISS, and HISS-SD compared with other algorithms for test set 6.

Ins.	*HISS-SD*	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*	*Best Known*
p6.2.d	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**
p6.2.j	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**
p6.2.l	**1116**	**1116**	**1116**	**1116**	1110	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**
p6.2.m	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**
p6.2.n	**1260**	**1260**	**1260**	**1260**	**1260**	1254	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**
p6.3.g	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**
p6.3.h	**444**	**444**	**444**	**444**	**444**	438	438	**444**	**444**	**444**	**444**	**444**
p6.3.i	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**
p6.3.k	**894**	**894**	**894**	**894**	888	888	**894**	**894**	**894**	**894**	**894**	**894**
p6.3.l	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**
p6.3.m	**1080**	**1080**	**1080**	**1080**	1074	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**
p6.3.n	**1170**	**1170**	**1170**	**1170**	1164	1164	1164	**1170**	**1170**	**1170**	**1170**	**1170**
p6.4.j	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**
p6.4.k	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**
p6.4.l	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**

Table 8. Best results of SS, ISS, and HISS-SD compared with other algorithms for test set 6.

Ins.	*HISS-SD*	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*	*Best Known*
p7.2.d	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**
p7.2.e	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**
p7.2.f	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**
p7.2.g	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**
p7.2.h	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**
p7.2.i	**580**	579	**580**	**580**	579	579	579	**580**	**580**	**580**	**580**	**580**
p7.2.j	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**
p7.2.k	**705**	704	**705**	**705**	704	704	704	**705**	**705**	**705**	**705**	**705**
p7.2.l	**767**	**767**	761	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**
p7.2.m	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**
p7.2.n	**888**	**888**	**888**	**888**	878	878	878	**888**	**888**	**888**	**888**	**888**
p7.2.o	**945**	**945**	**945**	**945**	**945**	940	941	**945**	**945**	**945**	**945**	**945**
p7.2.p	**1002**	**1002**	**1002**	**1002**	991	993	993	**1002**	**1002**	**1002**	**1002**	**1002**
p7.2.q	1043	1043	**1044**	1043	1042	1043	1043	**1044**	**1044**	**1044**	**1044**	**1044**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

p7.2.r	**1094**	1089	1092	**1094**	1093	1088	**1094**	**1094**	**1094**	**1094**	**1094**	**1094**
p7.2.s	**1136**	1134	1128	**1136**	**1136**	1134	1131	**1136**	**1136**	**1136**	**1136**	**1136**
p7.2.t	**1179**	1174	1164	**1179**	**1179**	**1179**	**1179**	**1179**	**1179**	**1179**	**1179**	**1179**
p7.3.h	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**
p7.3.i	**487**	**487**	**487**	**487**	**487**	486	**487**	**487**	**487**	**487**	**487**	**487**
p7.3.j	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**
p7.3.k	**633**	**633**	**633**	**633**	632	**633**	**633**	**633**	**633**	**633**	**633**	**633**
p7.3.l	**684**	**684**	682	**684**	683	**684**	**684**	**684**	683	**684**	**684**	**684**
p7.3.m	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**
p7.3.n	**820**	**820**	**820**	**820**	819	819	**820**	**820**	**820**	**820**	**820**	**820**
p7.3.o	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**
p7.3.p	927	**929**	927	**929**	925	926	925	**929**	927	**929**	**929**	**929**
p7.3.q	**987**	**987**	**987**	**987**	**987**	**987**	**987**	**987**	**987**	**987**	**987**	**987**
p7.3.r	**1026**	1024	1024	**1026**	1024	1021	1022	**1026**	**1026**	**1026**	**1026**	**1026**
p7.3.s	**1081**	**1081**	**1081**	**1081**	**1081**	**1081**	1077	**1081**	**1081**	**1081**	**1081**	**1081**
p7.3.t	**1120**	1118	1116	1118	1117	1103	1117	**1120**	**1120**	**1120**	**1120**	**1120**
p7.4.g	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**
p7.4.h	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**
p7.4.i	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**
p7.4.k	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**
p7.4.l	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**
p7.4.m	**646**	**646**	**646**	**646**	644	**646**	**646**	**646**	**646**	**646**	**646**	**646**
p7.4.n	**730**	726	**730**	**730**	725	725	726	726	726	**730**	**730**	**730**
p7.4.o	**781**	**781**	**781**	**781**	778	**781**	778	**781**	**781**	**781**	**781**	**781**
p7.4.p	**846**	**846**	**846**	**846**	**846**	838	842	**846**	**846**	**846**	**846**	**846**
p7.4.q	**909**	908	**909**	**909**	**909**	**909**	**909**	**909**	**909**	**909**	**909**	**909**
p7.4.r	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**
p7.4.s	**1022**	**1022**	**1022**	**1022**	1019	1021	1019	**1022**	**1022**	**1022**	**1022**	**1022**
p7.4.t	**1077**	**1077**	**1077**	**1077**	1072	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**

Table 9: Performance comparison based on RPE average for each data set of the associated instance.

Data set	*RPE average for each dataset*										
	HISS-SD	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*
Test Set 4	0.003	0.090	0.196	0.312	0.903	0.861	0.775	0.030	0.026	0.002	0.002
Test Set 5	0.000	0.015	0.246	0.036	0.231	0.257	0.284	0.061	0.015	0.000	0.000
Test Set 6	0.000	0.000	0.000	0.000	0.152	0.201	0.124	0.000	0.000	0.000	0.000
Test Set 7	0.007	0.058	0.097	0.006	0.146	0.188	0.150	0.013	0.021	0.000	0.000
RPE AVG	0.003	0.041	0.132	0.089	0.358	0.377	0.333	0.026	0.016	0.000	0.000
Eq. Best	153	123	115	128	80	80	84	146	146	156	156

The efficiency of the algorithm was measured by the relative percentage error (RPE) and the average RPE (ARPE). The RPE is clarified as the relative error between the best known results obtained in scientific literature and the best results obtained by the proposed algorithm. It shows the performance of the proposed algorithm over a number of runs, as in Equation (11). On the other hand, the ARPE is clarified as the relative error between the best known results obtained in scientific literature and the average results obtained by the proposed algorithm over a number of runs. It shows the robustness of the proposed algorithm, as in Equation (12).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$RPE = \frac{x_{best} - x_{max}}{x_{best}} . 100 \quad (11)$$

$$ARPE = \frac{x_{best} - x_{avg}}{x_{best}} . 100 \quad (12)$$

where x_{best} is the best known result found in literature, x_{max} is the best results obtained by the proposed algorithm over all the independent runs, and x_{avg} is the average of the obtained results by the proposed algorithm over a number of runs.

Tables 9 presents the RPE average value for each set of standard benchmark instances. SS, ISS, and HISS-SD are compared with state-of-the-art of algorithms based on the RPE average value. Table 8 evidently shows that the RPE average values of SS, ISS, and HISS-SD are 0.132, 0.041, and 0.003, respectively. SS outperforms 3 algorithms (ADC, ARC, and ASi) out of 8 algorithms. Meanwhile, ISS outperforms 4 algorithms (ASe, ADC, ARC, and ASi) out of 8 algorithms. HISS-SD outperforms 6 algorithms (ASe, ADC, ARC, ASi, MA10, PSOMA) out of 8 algorithms. The last row in Table 8 presents the number of solutions that is equal to the best known results.

Table 10 presents the ARPE average for each set of standard benchmark instances. SS, ISS, and HISS-SD are compared with the state-of-the-art algorithms, wherein the average results are reported in scientific literature based on ARPE average value. The table shows that SS and ISS outperform 4 compared algorithms (ASe, ADC, ARC, and ASi) out of 8 algorithms. HISS-SD outperforms 7 compared algorithms (ASe, ADC, ARC, ASi, MA10, PSOMA, and PMA) out of 8 algorithms.

Table 10: Performance comparison based on ARPE average for each data set of the associated instance.

Data set	*ARPE average for each dataset*										
	HISS-SD	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*
Test Set 4	0.166	0.893	1.066	1.866	1.867	2.063	1.703	0.207	0.285	0.110	0.318
Test Set 5	0.050	0.232	0.712	0.823	1.161	1.107	1.181	0.095	0.090	0.034	0.090
Test Set 6	0.000	0.017	0.056	1.071	1.170	1.210	1.065	0.017	0.000	0.000	0.293
Test Set 7	0.056	0.389	0.439	0.512	0.617	2.532	2.457	0.106	0.179	0.030	0.128
ARPE AVG	0.068	0.383	0.568	1.068	1.204	1.728	1.601	0.106	0.139	0.044	0.208

The average values obtained by SS, ISS, HISS-SD, and the state-of-the-art algorithms are used to compare the performances of these algorithms through a Friedman statistical test. The *p*-values obtained through the Friedman and Iman–Davenport statistical tests are less than the critical level (0.05), which indicates the statistically different performances of the compared algorithms. Hence, the difference among the compared algorithms is detected through a Holm and Hochberg post-hoc statistical test (see Derrac, et al. [35] for more details).

Table 10 summarizes the average ranking (lower is better) for each algorithm (SS, ISS, and HISS-SD are included in the ranking). The ranking shows that SS is ranked 6.66, ISS is 6.05, and HISS-SD is 3.3. Therefore, PSOiA becomes the controlled method in the Holm and Hochberg statistical test because it is ranked as the 1st algorithm. The *p*-values computed by the Friedman test is 0.000, which is way below the significance interval of 95% (α = 0.05). This result shows that there is a significant difference among the observed results listed in Tables 5-9.

Next, the Holm and Hochberg statistical test [35] is performed to obtain the adjusted p-values for each comparison between PSOiA (the control method) and other algotherms. The adjusted p-value of the Holm and Hochberg statistical test is summarized in Table 12. PSOiA algorithm is statistically better than ARC, ADC, ASi, ASe, SS, ISS, PMA, PSOMA, and MA10 with a critical level of 0.05 (adjusted p-value < 0.05). Whereas, the results also reveal that HISS-SD is comparable to PSOiA with a critical level of 0.05 (adjusted p-value > 0.05). Notably, SS and ISS are included in the comparison with the state-of-the-art algorithms.

Table 11: Average rank obtained through a Friedman test for the compared algorithms.

Algorithm	Ranking
ASe	8.15
ADC	8.78
ARC	9.05
ASi	8.58
MA10	3.86
PSOMA	4.30
PSOiA	2.80
PMA	4.42

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SS	6.66
ISS	6.05
HISS	3.30

In summary, multiple comparison statistical tests indicate that HISS-SD an effective and efficient solution method for the TOP compared with the state-of-the-art algorithms.

Table 12: Adjusted p-value obtained through a Holm and Hochberg statistical test for the compared algorithms

i	Algorithm	Unadjusted *P*	*pHolm*	*pHochberg*
1	ARC	0	0	0
2	ADC	0	0	0
3	ASi	0	0	0
4	ASe	0	0	0
5	SS	0	0	0
6	ISS	0	0	0
7	PMA	0.000	0.000	0.000
8	PSOMA	0.000	0.000	0.000
9	MA10	0.004	0.009	0.009
10	HISS	0.188	0.188	0.188

The average execution CPU times for SS, ISS, and HISS are then compared with the algorithms for datasets 1–7 in Table 5.13. The execution times for some of the algorithms (ARC and ASi) are reported for datasets 4–7.

Table 13: Average execution time for each standard benchmark dataset.

	HISS-SD	*ISS*	*SS*	*ASe*	*ADC*	*ARC*	*ASi*	*MA10*	*PSOMA*	*PSOiA*	*PMA*
Test Set 1	0.17	0.08	0.149	4.877	5.238	-	-	1.95	0.18	2.15	6.81
Test Set 2	0.00	0.01	0.009	2.776	2.994	-	-	0.24	0.01	0.41	1.43
Test Set 3	0.15	0.08	0.163	5.607	5.923	-	-	2.06	0.49	3.18	9.56
Test Set 4	80.39	69.19	145.782	31.469	32.735	8.6	367.4	182.36	83.89	218.58	109.34
Test Set 5	50.24	46.34	54.974	14.331	15.132	2.9	119.9	35.33	14.72	49.5	22.86
Test Set 6	62.02	58.34	68.402	13.813	14.465	2.1	89.6	39.07	7.59	47.08	26.38
Test Set 7	120.39	111.34	138.521	23.312	24.652	6.3	272.8	112.75	49.09	97.47	54.56
AVG Time	44.77	40.77	58.286	13.741	14.448	4.975	212.425	53.394	22.281	59.767	35.58

6. CONCLUSION

We have proposed a Scatter Search algorithm for Team Orienteering problem. SS follows the general framework of the population-based metaheuristic [31] while it has five prominent components: (1) *A Diversification Generation Method*: SS employ this component to generate good starting solutions to solve the TOP. (2) *An Improvement Method*: The aim of this operation is to explore the neighborhood of the generated solutions by modifying it [21]. In our algorithm, we used a Steepest Descent (SD) heuristic as the improvement method to search for a better quality solution. (3) *A reference set update method:* this component is the heart of a SS procedure, which may result in important modifications during the search process because of its initial composition [21]. In this work, a Queen Bee (QB) strategy is used to build and update the reference set. (4) *Subset generation method (SGM)*: plays an important role in selecting pairs from a reference set for the combination method to generate a new solution [23]. In the present work, greedy select parents (GSP) is employed as selection mechanism to generate a subset of pairs for the combination method. (5) *A solution combination method*: The main goal of this component is to generate one or more trial solutions from various regions of the solution search space.

We introduced new improved SS called ISS. In the ISS, two main modifications were employed: a selection strategy within the subset generation method (greedy select parents (GSP)) and an updating strategy within the reference set update method (Queen Bee (QB) strategy). The performance of the ISS can be further enhanced because the SS is a population-based algorithm and concentrates more on exploration (diversification) than on exploitation (intensification) [31]. Furthermore, a hybridization of the ISS algorithm with Steepest Descent (SD), namely, HISS-SD was presented for the TOP. Generally, the hybridization in this paper managed to improve the quality of the obtained results by ISS further. The comparison between the proposed algorithms showed that HISS-SD is the best algorithm among the proposed algorithms in terms of yielding the best results. To further investigate the performance of HISS-SD, we compare the HISS-SD with state-of-the-art algorithms. The results were compared in terms of maximum obtained value, relative percentage error, average relative percentage error, and average CPU time. A statistical test was conducted to determine

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the algorithm that performed better compared with the others. The results revealed that SS and ISS only outperformed some of the state-of-the-art algorithms. By contrast, HISS-SD outperformed all state-of-the-art algorithms and was comparable to one algorithm.

The proposed algorithms have been tested on TOP benchmark datasets. The proposed algorithms could be tested and validated with respect to other variants of the orienteering problem, such as team orienteering problem with time windows, multi-constraint team orienteering problem with (multiple) time windows, orienteering problem with hotel selection, and orienteering problem with hotel selection and time windows.
In future work, we intend to test the proposed HISS-SD on other combinatorial optimization problems. And the performance of the proposed algorithms could be improved by dynamically changing the parameter of SS during the search process based on objective function value, which generates interesting results.

REFERENCES

[1] S. E. Butt and T. M. Cavalier, "A heuristic for the multiple tour maximum collection problem," *Computers & Operations Research,* vol. 21, no. 1, pp. 101-111, 1// 1994.

[2] I. M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European Journal of Operational Research,* vol. 88, no. 3, pp. 464-474, 2/8/ 1996.

[3] H. Tang and E. Miller-Hooks, "A TABU search heuristic for the team orienteering problem," *Computers & Operations Research,* vol. 32, no. 6, pp. 1379-1407, 6// 2005.

[4] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL),* vol. 34, no. 3, pp. 307-318, 1987.

[5] P. Vansteenwegen, W. Souffriau, G. Berghe, and D. Oudheusden, "Metaheuristics for Tourist Trip Planning," in *Metaheuristics in the Service Industry*, vol. 624, K. Sörensen, M. Sevaux, W. Habenicht, and M. J. Geiger, Eds. (Lecture Notes in Economics and Mathematical Systems: Springer Berlin Heidelberg, 2009, pp. 15-31.

[6] S. Boussier, D. Feillet, and M. Gendreau, "An exact algorithm for team orienteering problems," *4OR,* vol. 5, no. 3, pp. 211-230, // 2007.

[7] S. E. Butt and D. M. Ryan, "An optimal solution procedure for the multiple tour maximum collection problem using column generation," *Computers and Operations Research,* vol. 26, no. 4, pp. 427-441, // 1999.

[8] D.-C. Dang, R. El-Hajj, and A. Moukrim, "A Branch-and-Cut Algorithm for Solving the Team Orienteering Problem," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, vol. 7874, C. Gomes and M. Sellmann, Eds. (Lecture Notes in Computer Science: Springer Berlin Heidelberg, 2013, pp. 332-339.

[9] B. Nicola, M. Renata, and S. M. Grazia, "A branch-and-cut algorithm for the Team Orienteering Problem," *International Transactions in Operational Research,* vol. 25, no. 2, pp. 627-635, 2018.

[10] L. Ke, C. Archetti, and Z. Feng, "Ants can solve the team orienteering problem," *Computers & Industrial Engineering,* vol. 54, no. 3, pp. 648-665, 4// 2008.

[11] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "A guided local search metaheuristic for the team orienteering problem," *European Journal of Operational Research,* vol. 196, no. 1, pp. 118-127, 7/1/ 2009.

[12] B.-I. Kim, H. Li, and A. L. Johnson, "An augmented large neighborhood search method for solving the team orienteering problem," *Expert Systems with Applications,* vol. 40, no. 8, pp. 3065-3072, 2013/06/15/ 2013.

[13] H. Bouly, D.-C. Dang, and A. Moukrim, "A memetic algorithm for the team orienteering problem," (in English), *4OR,* vol. 8, no. 1, pp. 49-70, 2010/03/01 2010.

[14] D.-C. Dang, R. N. Guibadj, and A. Moukrim, "An effective PSO-inspired algorithm for the team orienteering problem," *European Journal of Operational Research,* vol. 229, no. 2, pp. 332-344, 9/1/ 2013.

[15] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden, "A Path Relinking approach for the Team Orienteering Problem," *Computers & Operations Research,* vol. 37, no. 11, pp. 1853-1859, 11// 2010.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[16] L. Ke, L. Zhai, J. Li, and F. T. S. Chan, "Pareto mimic algorithm: An approach to the team orienteering problem," *Omega,* vol. 61, pp. 155-166, 2016/06/01/ 2016.

[17] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research,* vol. 255, no. 2, pp. 315-332, 2016/12/01/ 2016.

[18] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research,* vol. 209, no. 1, pp. 1-10, 2/16/ 2011.

[19] F. Glover, "HEURISTICS FOR INTEGER PROGRAMMING USING SURROGATE CONSTRAINTS," *Decision Sciences,* vol. 8, no. 1, pp. 156-166, 1977.

[20] M. Laguna, R. Martín, and R. C. Martí, *Scatter search: methodology and implementations in C*. Springer, 2003.

[21] M. C. Resende, C. Ribeiro, F. Glover, and R. Martí, "Scatter Search and Path-Relinking: Fundamentals, Advances, and Applications," in *Handbook of Metaheuristics*, vol. 146, M. Gendreau and J.-Y. Potvin, Eds. (International Series in Operations Research & Management Science: Springer US, 2010, pp. 87-107.

[22] F. Glover, M. Laguna, and R. Marti, "Scatter Search," in *Advances in Evolutionary Computing*, A. Ghosh and S. Tsutsui, Eds. (Natural Computing Series: Springer Berlin Heidelberg, 2003, pp. 519-537.

[23] F. Glover, M. Laguna, and R. Marti, "Scatter Search and Path Relinking: Advances and Applications," in *Handbook of Metaheuristics*, vol. 57, F. Glover and G. Kochenberger, Eds. (International Series in Operations Research & Management Science: Springer US, 2003, pp. 1-35.

[24] F. Glover, "A template for scatter search and path relinking," in *Artificial Evolution*, vol. 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, Eds. (Lecture Notes in Computer Science: Springer Berlin Heidelberg, 1998, pp. 1-51.

[25] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and cybernetics,* vol. 39, no. 3, pp. 653-684, 2000.

[26] Z. Michalewicz and D. B. Fogel, "Traditional Methods — Part 1," in *How to Solve It: Modern Heuristics*Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 55-81.

[27] M. Laguna and R. Martí, "Scatter Search," in *Metaheuristic Procedures for Training Neutral Networks*, vol. 36, E. Alba and R. Martí, Eds. (Operations Research/Computer Science Interfaces Series: Springer US, 2006, pp. 139-152.

[28] V. Campos, F. Glover, M. Laguna, and R. Martí, "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem," (in English), *Journal of Global Optimization,* vol. 21, no. 4, pp. 397-414, 2001/12/01 2001.

[29] H. A. Alkhazaleh, M. Ayob, Z. Othman, and Z. Ahmad, "Constructive heuristics for team orienteering problems," *Journal of Applied Sciences,* vol. 13, no. 6, pp. 876-882, 2013.

[30] C. Blum and A. Roli, "Hybrid Metaheuristics: An Introduction," in *Hybrid Metaheuristics: An Emerging Approach to Optimization*, C. Blum, M. J. B. Aguilera, A. Roli, and M. Sampels, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1-30.

[31] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. John Wiley & Sons, 2009, p. 593.

[32] H. A. Alkhazaleh, M. Ayob, Z. Othman, and Z. Ahmad, "Diversity measurement for a solution of team orienteering problem," *International Journal of Advancements in Computing Technology,* vol. 5, no. 16, p. 21, 2013.

[33] G. M. Jaradat and M. Ayob, "Scatter search for solving the course timetabling problem," in *Data Mining and Optimization (DMO), 2011 3rd Conference on*, 2011, pp. 213-218.

[34] R. Martí, M. Laguna, and F. Glover, "Principles of scatter search," *European Journal of Operational Research,* vol. 169, no. 2, pp. 359-372, 3/1/ 2006.

[35] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation,* vol. 1, no. 1, pp. 3-18, 2011/03/01/ 2011.