Hybrid elitist–ant system for a symmetric traveling salesman problem: case of Jordan

Article *in* Neural Computing and Applications · July 2016

DOI: 10.1007/s00521-016-2469-3

1 author:

Ghaith M. Jaradat
Jerash University
62 PUBLICATIONS **98** CITATIONS

SEE PROFILE

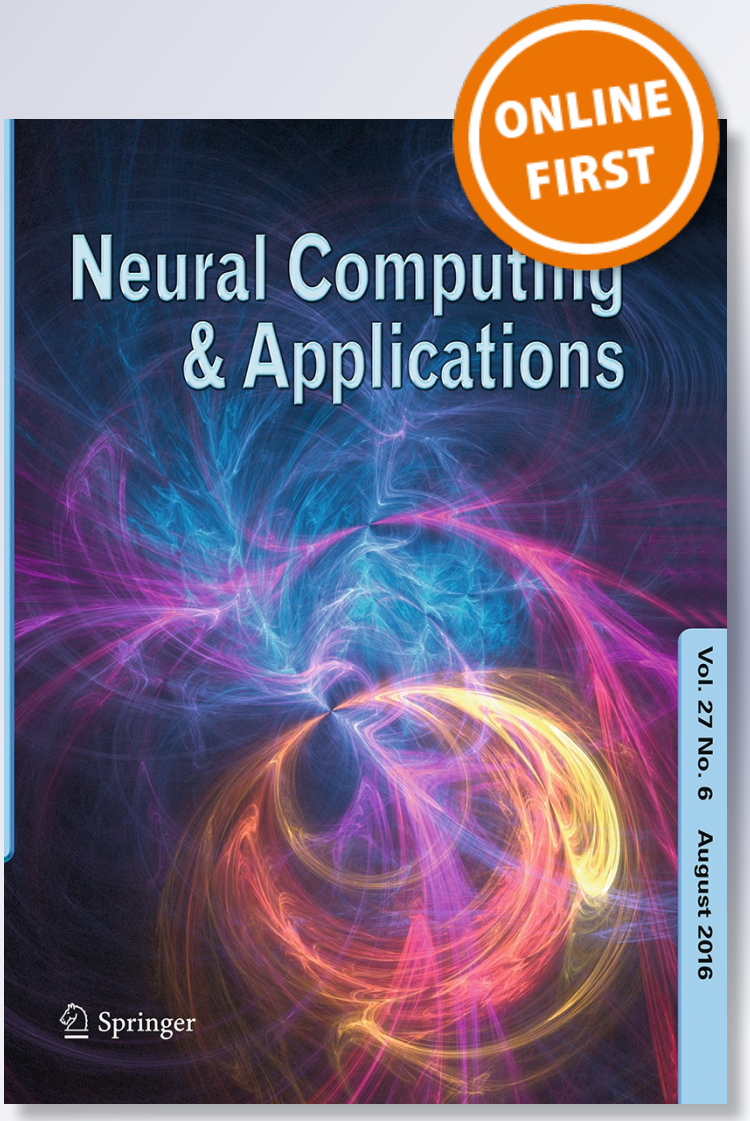
Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan

Ghaith M. Jaradat

Neural Computing and Applications

ISSN 0941-0643

Neural Comput & Applic
DOI 10.1007/s00521-016-2469-3



Springer

Neural Comput & Applic
DOI 10.1007/s00521-016-2469-3

ORIGINAL ARTICLE

Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan

Ghaith M. Jaradat[1]

Received: 6 October 2015 / Accepted: 6 July 2016

Abstract This work investigates the performance of a hybrid population-based meta-heuristic with an external memory structure of a hybrid elitist-ant system (elitist-AS). This memory is known as an elite pool, which contains high quality and diverse solutions to maintain a balance between diversity and quality of the search. This may guarantee the effectiveness and efficiency of the search, which could enhance the performance of the algorithm across different instances. A very well known and intensively studied NP-hard optimization problem has been selected to test the performance of the hybrid elitist-AS via its consistency, effectiveness and efficiency. This famous problem is the symmetric traveling salesman problem. The elitist-AS is a class of ant colony optimization techniques which are known to be outstanding for the traveling salesman problem where they have the ability to find the shortest tours guided by the heuristic and the pheromone trail information. An iterated local search is combined with elitist-AS to intensify the search around elite solution and maintains the solution's exploitation mechanism. Experimental results showed that the performance, compared to the best known results, is optimal for many instances. This finding indicates the effectiveness, efficiency and consistency in diversifying the search while intensifying high-quality solutions. This outstanding performance is due to the utilization of an elite pool along with diversification and intensification mechanisms. In addition, this work proposes two instances that consist of 26 Jordanian cities and 1094 Jordanian locations which have been generated based on coordinates and distances similar to the format of the selected symmetric traveling salesman problem. This step is meant to contribute to finding a solution for a real-world problem and further test the performance of the hybrid elitist-AS.

Keywords Hybrid elitist-ant system · Elite pool · Diversification · Intensification · Iterated local search · Traveling salesman problem

1 Introduction

The ongoing research to build a successful generic problem solver offers prospects for the creation of artificial intelligent systems that can generate practical solutions to many combinatorial optimization problems (COPs). Prioritizing computational effectiveness and efficiency, research in meta-heuristics has evolved rapidly in an attempt to find good quality solutions to these problems within a desired time frame [17]. There is a wide variety of meta-heuristics that have been introduced. They are classified as single-solution versus population-based approaches. On the one hand, there are the single-solution approaches focus on modifying and improving a single candidate solution, like simulated annealing for example [4]. On the other hand, population-based approaches maintain and improve multiple candidate solutions (population of solutions) like genetic algorithms for example [4]. Talbi [26] illustrated another class of meta-heuristics—namely swarm intelligence, which is a collective behavior of decentralized and self-organized agents in a population or swarm such as the ant colony optimization, particle swarm optimization and artificial bee colony.

The population-based method is used because of its capability to explore search space and can be easily

✉ Ghaith M. Jaradat
ghaith_jaradat@yahoo.com

[1] Department of Computer Science, Faculty of Information Technology, Jerash University, Jerash 26150-311, Jordan

Published online: 15 July 2016

combined with local search method in order to enhance solution exploitation mechanisms [26]. The local search method is utilized because of its capability to exploit solution spaces [4]. The strength of population-based method depends on the capability of recombining solutions to obtain new ones [4]. In population-based method, a structured solution recombination of elite solutions is performed explicitly, which involves swapping permutations in a solution, representing the information exchange between generations. One or more recombination operators are used, such as crossover and mutation [4] in genetic algorithms. The term explicit means that a solution is represented directly by the actual permutation and fitness values of solutions. Another kind of recombination is the one performed implicitly, where new solutions are generated using a distribution over the search space that is, in turn, a function of the earlier populations representing the search experience [4]. Ant systems use implicit recombination. It is understood that the term implicit here means that a solution is represented indirectly by the fitness values of the assignments or values of their contribution to search (e.g., constructing a solution). This step enables the search to perform a guided sampling of the search space.

However, the intensification mechanism in a population-based method still needs to be improved in order to obtain high-quality solutions. Hence, in order to enhance the intensification process, a meta-heuristic that has a capability in exploiting the solution space is usually hybridized with population-based methods. Many studies have recommended this hybridization, including [4, 26]. A local search method has the capability in exploiting the solution space, so it can be complemented with the population-based method to overcome the weakness in a population-based method and to further enhance the quality of solutions in the pool.

Generally, the term elite pool can be defined as an adaptive memory structure with a set of diverse and/or high-quality solutions that stores useful information about the global optima in the form of a diverse and elite set of solutions. This structure gives the search process the ability to recombine samples from the elite set, so as to exploit useful information about the global optima.

In addition, the utilization of an elite pool, that has high quality and diverse solutions, is used to control the diversity of search, and a dynamic manipulation of the population size is also recommended for better performance of hybrid meta-heuristics [26]. A good performance, with regards to consistency, efficiency and may be generality, is presented by maintaining a balance between diversification and intensification of the search [9]. Therefore, the author chose the elitist-ant system (elitist-AS) for this study to test it in the context of the traveling salesman problem (TSP). This elitist-AS had been developed and hybridized in his previous study with some diversification and intensification mechanisms to enhance its solution space exploration and exploitation. This has been achieved in his previous work by incorporating an elite pool that contains diverse and high-quality solutions alike as well as by employing a local search heuristic to intensify the search around elite solutions while maintaining a degree of diversity. It was applied to the course timetabling problem (see [11]). The reason behind choosing the elitist-AS is due to the features stated by [8]:

1. It has structured solution combinations using diversification strategies that rely on randomization. Therefore, this is a good reason to investigate the effect of incorporating an elite pool into the algorithm instead of randomization or pseudo-random selection strategies of solutions.
2. It evolves a strategy of updating the search in a form of exploiting an elitism strategy with a pool of diverse solutions, pheromone trails, to preserve good quality and diversity.
3. It provides useful information about the collection of elite and/or diverse solutions. However, it does not originally possess elite pool of high quality and diverse solutions. It is sufficient for exploring and exploiting the solution space but still insufficient to balance between diversity and quality.
4. It supports indirect solution representation in a Euclidean space that is easy to manipulate.

In addition, the elitist-AS is chosen to experiment the effect of employing an elite pool alongside their implicit solution recombination. It is also aimed at comparing the elitist-AS against other methods—i.e., genetic algorithm, as an explicit recombination-based meta-heuristic. Therefore, this study aims to investigate the effect of elite pool to the performance of a population approach. Based on the utilization of an elite pool, the study investigates the performance of the hybrid elitist-AS by testing it on a well-known class of NP-hard COPs, the TSP. In this study, a fixed size of the memory and a maintained update strategy of the memory structures are considered. The investigation also compared the proposed hybrid meta-heuristic to similar hybrid approaches and standalone methods such as simulated annealing, genetic algorithm, particle swarm optimization, harmony search, bat algorithm and ant systems.

Therefore, this research study a question boils down to the following: "Does the use of elite pool (a pool diverse and high-quality solution) in population-based meta-heuristic can enhance the performance of meta-heuristic compared to the one that only use diverse pool?" Hence, the objectives include the following:

1. Propose a population-based meta-heuristic by incorporating a memory structure (e.g., elite pool), which contains a set of diverse and high-quality solutions to strike a balance between diversification and intensification—exploration and exploitation—within the search space, *and*
2. Test the performance, i.e., effectiveness and consistency, of the proposed hybrid population-based meta-heuristic over the TSP domains, that is of a very different nature. The study also compares the results with the state-of-the-art population-based meta-heuristics.

The paper is organized as follows: Sect. 2 describes the problem chosen for the study. Some related works are presented in Sect. 3. The proposed hybrid meta-heuristic and its design are presented in Sect. 4. Sections 5 and 6 show and discuss the experimental results obtained by the proposed hybrid meta-heuristic. Finally, the conclusions are presented in Sect. 7.

2 Description of the problem

Before describing the problem, the reason behind choosing the TSP should be identified—although it has been intensively studied in the literature, it is very common and has been derived from real-world applications. Hence, it provides a very good platform for testing the impact of hybridization and elite pool.

The TSP is an important optimization problem that represents many fields of real-world problems in the areas of transportation, logistics and semiconductor industries [24]. The basic principle of TSP entails that the salesman starts his travels from a city to his destination before returning to the starting point while trying to obtain a closed tour (circle) with a minimum cost. Every city once must be visited only once during the tour. The cost of the tour directly depends on its length.

The TSP can be generally defined as the consideration of a set N of nodes representing the cities, and a set E of arcs that fully connect the nodes N, where d_{ij} is the length of the arc $(i, j) \in E$, that is the distance between cities i and j, with $i, j \in N$. The TSP is the problem of finding a minimal length (cost) Hamiltonian circuit (path) on the undirected graph $G = (N, E)$, where an Hamiltonian circuit of graph G is a closed tour that entails visiting once and only once all the $n = |N|$ nodes of G, and where its length is given by the sum of the lengths of all the arcs of which it is composed [6].

In this study, the concept of calculating the distances between cities from [12] is adopted, where the distance between city i and city $(i + 1)$, $d(T[i], T[i + 1])$ is calculated as the Euclidean distance using the following equation:

$$d(T[i], T[i+1]) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}. \quad (1)$$

The aim of solving TSP is to minimize the total closed tour length (see [22] for details). Total tour length f can be expressed as follows:

$$f = \sum_{i=1}^{n-1} d(T[i], T(i+1) + d(T[n], T[1])) \quad (2)$$

where n is the total number of cities.

Hence, the fitness (quality) of each solution fit_i is calculated using the following equation:

$$fit_i = \frac{1}{1 + f(x_i)} \quad (3)$$

where $f(x_i)$ is the tour length of solution x_i.

The hybrid elitist-AS generates a population of initial solutions using nearest neighbor construction heuristic similar to the one used in [1], and the distances between cities are measured by integer numbers as in [12]. The instances used in this work are taken from the TSPLIB benchmark library that was introduced by [22, 23]. These instances have been used in many studies, and some of them were extracted from practical applications of the TSP.

In symmetric TSP (STSP), the distance between two cities is the same in each opposite direction—forming an undirected graph. This symmetry halves the number of possible solutions, whereas in asymmetric TSP (ATSP), paths may not exist in both directions or the distances might be different—leading to the formation of a directed graph. Traffic collisions, one-way streets and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down.

In this study, a set of STSP instances is chosen—to be a platform for testing—due to a large variety of works in the related literature. All STSP instances used in the literature are taken from the TSPLIB benchmark library introduced by [22, 23]. These instances are used in many studies, and some of them were extracted from practical applications of the TSP. The structure of a STSP instance is shown in Figs. 1, 2 and 3 that feature new instances created by the author to solve the problems of 26 Jordanian cities and 1094 Jordanian locations. The instance with the dimension of 26 cities is generated in two formats;[1] one is based on the distances between cities (see Fig. 1), while the other is based on the coordinates of the cities (see Figs. 2, 3).

The third row, dimension, represents the number of cities. Elements of the matrix represent the distance (d) between cities i and $i + 1$.

[1] Coordinates and distances data were collected from the Ministry of Tourism and the Royal Geographic Centre in the Hashemite Kingdom of Jordan.

```
NAME: jor26
TYPE: TSP
DIMENSION: 26
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: UPPER_DIAG_ROW
DISPLAY_DATA_TYPE: NO_DISPLAY
EDGE_WEIGHT_SECTION
0     88    76    48    85    64    22    .....  235
88    0     32    38    23    50    88    .....  313
76    32    0     25    55    67    71    .....  303
.....  .....  .....  .....  .....  .....  .....  .....  .....
235   313   303   281   342   306   257   .....  0
EOF
```

Fig. 1 STSP instance for 26 Jordanian cities—distance based

```
NAME: jorE26
TYPE: TSP
DIMENSION: 26
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1   31.978259576126394    35.892333984375
2   32.51346758139904     35.8978271484375
3   32.318008790008015    35.742645263671875
.....  .....  .....
26  30.324428113515598    35.44567108154297
EOF
```

Fig. 2 STSP instance for 26 Jordanian cities—coordinate based

```
NAME : jorE1094
TYPE : TSP
DIMENSION : 1094
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1     31.978259576126394    35.892333984375
2     32.51346758139904     35.8978271484375
3     32.318008790008015    35.742645263671875
.....  .....  .....
1094  32.21845360659106     37.09264934062958
EOF
```

Fig. 3 STSP instance for 1094 Jordanian locations—coordinate based

3 Related work

A large variety of methodologies were applied to different classes of TSPs. Most common and interesting methodologies are presented in the following subsection. While a variety of heuristics and meta-heuristics has been widely used and successfully applied to solve the STSP for several years, the elitist-AS has not yet been applied to solve the TSPs.

3.1 TSP

Some examples of meta-heuristics and hybrid approaches applied on the TSPs include combinatorial an artificial bee colony [12], ant systems and ant colony systems [2, 5, 13, 15, 20, 21], a genetic algorithm [27, 29], a scatter search [19], a particle swarm optimization [14], a harmony search [3], simulated annealing [28], discrete cuckoo algorithm [18] and an improved bat algorithm [16].

The following discussion is dedicated to the selected methods for the comparison study in Sect. 6:

- An accumulated experience ant colony system was developed by [15] to solve the STSP. The search is updated based on accumulating ants' experience of building a shortest tour length with the help of an external memory structure.
- Ant system and two-stage ant colony system with 2opt were developed by [20] to solve the STPS. The first method is the conventional ant system, while the second method is an ant colony system that relies on two stages of search improvement on the bases of a 2opt approach.
- An ant colony system was developed by [21] to solve the STSP, which is based on updating the search via the ant's pheromone on two stages of search improvement and re-initialization.
- A hybrid ant colony optimization with simulated annealing was developed by [2] to solve the STSP, where the search process starts with construction of a tour using the ant colony optimization and then the simulated annealing is employed for the improvement.
- A conventional genetic algorithm and a hybrid generalized chromosome genetic algorithm with a local search were developed by [29] to solve the STSP. The second hybrid method is developed to diversify the search by the generalized chromosome strategy, while intensifying the search is made by the local search.
- A hybrid genetic algorithm was developed by [27] to solve the STSP. The genetic algorithm is improved by with two local search methods. The first method is a four vertices and three lines inequality applied to local Hamiltonian paths to generate shorter circuits. The second method is then executed to reverse the local Hamiltonian paths with more than two vertices, which also generates shorter circuits.
- A conventional harmony search was developed by [3] and adapted to solve the STSP. Originally, it has a memory structure to store only high-quality solutions.
- An improved simulated annealing was developed by [28] to solve the STSP. The improvement is made by incorporating a memory structure to diversify the search.
- A discrete cuckoo search algorithm was developed by [18] to solve the STPS. It was improved by reconstructing the population of solutions via incorporating the Levy flights strategy to control the balance between intensification and diversification.

- An improved discrete bat algorithm is developed by [16] to solve both STSP and ATSP. In this improved version, intelligent bats are deployed to follow different patterns of movement depending on the point of the solution space in which they are located. Hamming distances between bats play an important role in generating new solutions (diversification), and two operations are considered for the intensification (namely 2opt and 3opt).

A computer code for the symmetric TSP called Concorde TSP solver is presented in [23]. The code is written in the ANSI C programming language and it is available for academic research use. The solver has been used to obtain the optimal solutions to the full set of 110 TSPLIB instances, the largest having 85,900 cities. The solver has a callable library which includes over 700 functions permitting users to create specialized codes for TSP-like problems. The solver allows users to solve TSP instances online.

Some population-based meta-heuristics have no elite pool to store good quality and diverse solutions, whereas the hybrid elitist-AS has an incorporated elite pool. These approaches include particle swarm optimization, ant colony system and artificial bee colony [25]. By default, path relinking and scatter search have a built-in elite pool (*aka* reference set), while the rest have been modified to include such elite pools. However, there has been no investigation of elitist-AS for solving traveling salesman problems to the author's best knowledge. Furthermore, there is no instance specifically generated for Jordanian cities. Therefore, this work contributes to solving a real-world problem for 26 cities and 1094 locations in the Hashemite Kingdom of Jordan. That is to aid the services in the Kingdom such as logistics, transportation and warehousing.

3.2 Elite pool

From the above brief review of different methods, it is concluded that many attempts have been made to solve TSPs by combining different approaches (hybridization). Two main properties of all the above methods in the literature can be summarized as follows: (1) first, use a heuristic method to obtain a feasible initial solution; (2) second, hybridize the meta-heuristic method with another heuristic method to improve the solution during the iteration process. Those works have shown significant improvements to the TSPs' optimal solutions with the implementation of, mainly, population-based hybridization. For example, a combination of population-based methods with multiple phase neighborhood search and/or greedy randomized adaptive search and/or local search, or heuristics to select or to generate heuristics, has been effectively solved the TSPs. The purpose of this hybridization is to expand the neighborhood strategy in the population-based method.

In addition, the adaptive memory structure is a major component of an efficient and effective hybrid meta-heuristic, e.g., scatter search algorithm. It stresses out the concepts of memory, intensification—exploitation—and diversification—exploration. A memory stands for the information collected by the algorithm on the objective function distribution. It can be represented as complex structures, like pheromone trails in the elitist-AS. Intensification exploits the information obtained, in order to improve the current solutions. This is typically a local search routine, while diversification aims at collecting new information, by exploring the search space.

The components presented (e.g., memory, intensification and diversification) are not always clearly distinct and are strongly interdependent in an algorithm. Therefore, the author prefers to utilize their advantages via a complex data structure that updates the search information more effectively, called the elite pool. It aims at taking full advantage of an adaptive memory; the author uses it as an improvement method of the best solutions obtained after combinations.

Some of the relationships are mentioned as follows. A pool is defined as a data structure used to store a number of solutions, which have turned out to be potentially useful throughout the search [10]. Members of a pool are called elite solutions. Hence, the concept of elite pool can be presented as an adaptive memory, which [29] used the idea of genetic algorithms of combining solutions to construct new solutions and employed a tabu search as an improvement procedure. Another similar approach that uses the concept of elite pool is the granular tabu search [28], which restricts the neighborhood size by removing edges from the graph that are unlikely to appear in an optimal solution.

Except for the scatter search, path relinking and hyper-heuristic, all of the above-mentioned approaches in the subsections have no elite pool of good quality and diverse solutions, while the proposed hybrid elitist-AS possess an incorporated elite pool. The above-mentioned approaches are selected to be compared to the proposed hybrid elitist-AS meta-heuristic in the paper due to the fact that they are applied on the same selected instances. See Sect. 5. It is worthwhile to evaluate the performance of the proposed hybrid elitist-AS algorithm for solving the STSP.

4 Hybrid elitist-ant system

This work investigates the impact of hybridization and elite pool on the effectiveness, efficiency and consistency of an elitist-AS from [11]. This is achieved by testing it on STSP

instances. The elitist-AS was originally proposed by [7]. Since the elitist-AS is set to strike a balance between search diversity and intensification while maintaining the quality of solutions in a previous work shared with Ayob [11], where the authors enhanced its capability by hybridizing the elitist-AS with an iterated local search (ILS), diversification and intensification mechanisms and an external memory to store elite solutions. The pheromone model is utilized in a way to (1) exploit the problem-specific knowledge effectively and rapidly in the solution construction phase, and to (2) enhance the diversity of constructing new solutions as well as the exploration of solution's neighborhoods in the improvement phase. This assists both pheromone evaporation and elite pheromone updating, as well as eventually gain control over search exploration and exploitation.

The diversification mechanism helps avoid early convergence, while the intensification mechanism helps in exploring the neighbors of a solution more effectively. The ILS is employed as an intensification mechanism to improve the individual ant solution. The diversification mechanism is employed by restarting the ant search when it stagnates to explore different regions when no further possible improvements of the search space. This will strengthen the ability of the pheromone deposition, intensification, and evaporation, diversification, in diversifying the search while maintaining the quality. Based on a collection of diverse and good quality solutions in the external memory and a predefined number of non-improvements in the ILS, the intensification mechanism will be initiated to improve solutions obtained from the ILS further. By contrast, the diversification mechanism will be applied once the *ILS* fails to improve the quality of solutions by re-initializing the pheromone trails. A generic pseudo-code of the hybrid elitist-AS is illustrated in Fig. 4 [11].

The hybrid elitist-AS algorithm starts by initializing all parameters as in Table 1 (Step 1). Then (Step 2) starts the search by constructing, from scratch, a population of initial solutions using a nearest neighbor constructive heuristic that is guided by pheromone trails. Each ant presents a solution that will be improved using the ILS for a significant enhancement of its quality. Once an elite solution, better quality and more diverse, is found, it will be stored in the elite pool—external memory. This solution will be utilized in the successive iterations as a reference to guide the search toward a global solution. If the elite solution is updated or a better solution is found, the intensification phase will proceed to further explore neighbors around the new elite solution in order to generate a better elite solution. If there is no improvement for a predefined number of iterations, stagnation state, the intensification phase will be skipped and the diversification phase commences. The diversification phase will re-initialize the pheromone trail values to restart the search. All steps will be repeated until the stopping criterion is met, which is either the maximum number of iterations or a global (lower or upper bound; problem dependent) solution is found. For further details about the phases of the hybrid elitist-AS, please refer to [11].

Once all ant solutions are feasible, they are improved using the ILS, guided by the pheromone trail matrix (Step 3). The ILS is simply implemented in five types of perturbation moves, where each move is applied for a predetermined number of iterations, 100 iterations are used in

Fig. 4 Generic pseudo-code of the hybrid elitist-AS for the TSP

```
Step 1: Initialization phase
While StoppingCriterion is not met do
Step 2: Construction phase
For each ant                                              //solution construction
        Apply Nearest Neighbour construction heuristic;
End for
Step 3: Improvement phase
   While non-improvement stopping criterion is not met do
            Locally improve each constructed solution;        // employ ILS
            Update size & content of external memory (elite pool);
End While
If the best solution is updated then
  Step 4: Intensification phase;
        Randomly explore the neighbors of the best solution found so far (elite solution);
  Step 5: Global Pheromone update phase;
        Update pheromone trails route appearing in solution;     // diversity pool
Else
  Step 6: Diversification phase;
        Pheromone evaporation;                                   // diversity control
        Reinitialize pheromone trails;
        Generate new population of ants using elite solutions in the elite pool by some perturbations;
End If
End While
Step 7: Return Best ant                                          // best solution
```

Table 1 Parameter settings used by the hybrid elitist-AS for TSP

Parameter	Value
Number of ants	25
Number of iterations	100,000
Number of non-improvement iterations	100
Pheromone initial values	0.01
Evaporation rate	$0.1 \in [.04, .06]$
Controlling ratio (exploration vs. exploitation)	$0.95 \in [0, 1]$
Importance of distances (penalty)	2.0
Initial external memory (elite pool) size	5 (elite solutions)
Local search routine	Iterated local search
Elitism (Search update)	Use the best ant to update global pheromone

this work, to determine the strength of the perturbation by increasing or decreasing pheromone values. A new solution is accepted if it is better than the current one. This leads to a first improvement descent in the space of the local optima. After applying the local search, the elite pool is updated including its contents and size containing elite solutions found (see Step 3). The elite pool stores the elite solutions found. It is used to maintain good search experience information in order to guide the search efficiently. It is also used in constructing new solutions in successive iterations from elite solutions by performing some perturbations rather than constructing them from scratch.

In Step 4, the intensification mechanism is employed to explore the neighbor of good solutions more effectively after the best solution found so far has been improved. This is done using a random descent heuristic, where some neighbors from all neighborhoods of a solution are generated and the best one is chosen until no more improvement is possible. Then, all ants start their next iteration with the best permutation performed to construct new solutions from previous ones, as in Step 2 but by performing perturbations to elite solutions rather than constructing new solutions from scratch.

Only the best ant, elite solution, updates the pheromone trails matrix, global update (see Step 5). An elite solution is one that has the optimal, best, fitness value.

In addition, the best performed neighbor that results an elite solution of a best solution in the current iteration will be used as the starting permutation of solutions in the next iteration in the improvement phase. This is to provide further intensification around promising regions when constructing new solutions. In this work, neighborhood structures are simply explored using a simple descent heuristic (Step 3) at each iteration. This step proceeds for a predetermined number of iterations.

The diversification mechanism (Step 6)—also called pheromone evaporation—is employed after performing a predefined number of non-improvement iterations of the best solution found so far in the local search routine. This helps avoid early algorithm convergence. As the diversification mechanism is employed each time, the mechanism periodically erases all the pheromone trails. Pheromone values are re-initialized once the intensification mechanism failed to improve all ant solutions. In other words, if the neighborhood structure is explored and no solution improvement is done, the pheromone trails must be erased to track a new path toward better quality solutions. Each ant will generate a new solution from the elite solution in the elite pool for all ants, except the one with best solution found so far, by performing some perturbations to the elite solutions. The whole process of the algorithm is repeated until the stopping criterion is met, i.e., either the best or optimal solution has been found, or the number of iterations has reached its limit.

5 Computational results

In this study, a hybrid elitist-AS is tested over twenty-nine STSP instances from TSPLIB.[2] As recommended by the related literature (e.g., [21]), the author ran the hybrid elitist-AS 25 times on each instance for 100,000 iterations as a stopping condition. The experiments were performed on Intel Core i7 2.30 GHz processor, 8 GB RAM, and implemented in Java NetBeans IDE v 7.4 parameters listed in Table 1 are determined experimentally (e.g., elite pool size) and based on related work (e.g., elitism). For example, the population size in the elitist-AS is preferred to be relatively small [9].

The author tends to fine-tune and determine the best suitable parameters in a certain stage (when no further improvement is possible) during the run time while solving the problem, noting that variable parameters are modified depending on the possible solution; a predefined sequence of values are adaptively changed due to the occurrence of a

[2] OR library website: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

certain state of the algorithm. There are too many instances for the STSP (e.g., 112 instances for the STSP). Therefore, the author determined to test the hybrid elitist-AS on some common instances tested across the literature.

Table 2 shows the best results obtained in this work for the test on STSP compared to best known or optimal results from the literature (*Optimal quality*). Best results are highlighted in bold (*Best Sol.*). The average (*Avg.*),

Table 2 Results of the hybrid elitist-AS applied to the STSP (20 runs for each case)

Instance[a]	Optimal quality[b]	Elitist-AS				
		Best sol.	Δ (%)	Avg.	Std.	Time (s)
att48	10,628	**10,628**	0	10,684	0	5
att532	27,686	28,147	1.66	28,147	19.22	612
a280	2579	**2579**	0	2593	13.03	92
bays29	2020	**2020**	0	2020	0	96
berlin52	7542	**7542**	0	7550	0	38
bier127	118,282	**118,282**	0	119,363	325.5	17
ch130	6110	**6110**	0	6265	22.1	56
ch150	6528	**6528**	0	6550	11.22	53
d198	15,780	15,888	0.68	15,940	64.83	103
d2103	80,450	80,688	0.29	80,760	1066.3	905
d15112	1,573,084	1,573,162	0.004	1,573,178	25,576.1	1301
eil51	426	**426**	0	429	0	25
eil76	538	**538**	0	551	0	42
fl1577	22,249	22,977	3.27	22,977	217.22	862
gr24	1272	**1272**	0	1272	0	67
hk48	11,461	**11,461**	0	11,461	2.5	13
kroA100	21,282	**21,282**	0	21,305	0	41
kroB150	26,130	26,185	0.21	26,556	0	42
kroA200	29,368	29,420	0.17	29,685	1.03	51
lin318	42,029	44,169	5.09	43,028	18.63	228
pcb442	50,778	51,286	1.0	52,303	908.1	199
pr76	108,159	**108,159**	0	108,503	0	32
rat575	6773	**6773**	0	6773	0	166
rat783	8806	**8806**	0	8834	1.27	388
rd100	7910	**7910**	0	7986	0	31
rd400	15,281	**15,281**	0	15,501	0	213
st70	675	**675**	0	675	0	95
tsp225	3919	**3919**	0	4285	2.63	52
vm1084	239,297	**239,297**	0	240,125	1859.9	427
Average deviation			**0.43 %**			**215.5**
jor26[c]	–	**2293**	0	2296		5
jorE26[d]	–	**14,707**	0	14,935		39
jorE1094	–	**55,356**	2.25	56,710		2155
Average deviation			**0.75 %**			**733**

The instances of STSP (TSPLIB95) have been officially declared in 2007 (by its founder [23]) as solved to optimality

[a] The number in the name of an instance gives the instance dimension/number of cities

[b] Optimal solutions are generated by the Concorde TSP solver. They are reported in the official website of the solver: http://www.math.uwaterloo.ca/tsp/concorde/benchmarks/bench.html

[c] In this study, the instance *jor26* is generated based on the *distances* format (edge-weight-type: EXPLICIT; and edge-weight-format: UPPER-DIAG-ROW)

[d] In this study, the instances *jorE26* and *jorE1094* are generated based on the *coordinates* format (edge-weight-type: EUC_2D)

standard deviation (*Std.*) and time in seconds for the hybrid elitist-AS are also presented. Table 3 shows the best results obtained in this work compared to those reported in the literature obtained through similar hybrid meta-heuristics. Many published works reported the deviation percentage of—Δ (%)—for their best solutions compared to the best known counterparts (e.g., [25]). Therefore, the author of this work reports his results in a method similar to other works and means of illustration.

Many published works reported the deviation percentage—Δ (%)—(or brief statistics) of their best solutions from best known solutions (e.g., [25]). Therefore, in this work the author reports his results similar to their illustration. The deviation percentage—Δ (%)—from the best known result found in the literature is calculated for each instance as follows: $\Delta\ (\%) = ((a - b)/b) \times 100$, where a is the best result returned over 25 independent runs by the hybrid elitist-AS, and b is the best known result found in the related literature.

Based on Table 2, the hybrid elitist-AS meta-heuristic has shown to be competitive in most cases with the best known results. The table also shows the gap between the best known solution and the hybrid elitist-AS. A zero gap indicates that best known solutions, or rather optimal, were obtained. Overall, in comparison to the best known solutions, the hybrid elitist-AS meta-heuristic can produce very good solutions. Elitist-AS has obtained optimal solutions for 20 instances out of 29. Across 29 instances—25 runs for each instance—the average Δ (%) is only 0.43 % for the elitist-AS. In addition, the consistency of the hybrid elitist-AS method can be defined by the *Std.* over 25 runs, where the *Std.* produced by elitist-AS is relatively small for all instances of the *STSP* (except ch130, d198, fl1577 and lin318). These indicate that the proposed methodology is very efficient and competitive to solve the STSP compared in terms of solution quality and consistency. Hence, meeting those criteria leads to the generality of the proposed hybrid elitist-AS meta-heuristic over different sizes of instances.

Figure 5 below shows the shortest tour obtained by the hybrid elitist-AS for instances *jorE26* and *jorE1094*.

6 Discussion

This section is devoted to assessing the performance of the hybrid elitist-AS meta-heuristic against other conventional and hybrid methods in the literature. So, the study aims at (1) assessing the benefit of integrating an elite pool within the proposed hybrid elitist-AS and (2) testing the generality and consistency of the hybrid elitist-AS over the TSP. This is achieved by comparing it against other conventional or hybrid meta-heuristics.

In order to support the hypothesis of the impact of the elite pool on the performance of population-based meta-heuristics, the study investigated this issue by comparing the hybrid elitist-AS meta-heuristic with a number of conventional and hybrid meta-heuristics which have no elite pool. For example, a typical genetic algorithm has a pool (specifically an explicit memory) of diverse solutions, but it does not possess a pool of elite solutions (diverse and high quality) [4, 26]. That is, why it has a great search diversification mechanism, but it lacks an efficient intensification mechanism [4]. In some algorithms such as memetic algorithms, the use of elite pool can affect the performance of meta-heuristic algorithms in solving variety of optimization problems [4]. Hybridization of local search method and diversification and intensification mechanisms within an elitist-AS is also hypothesized of the impact on the performance of a population-based meta-heuristic.

Table 3 shows results obtained by the hybrid elitist-AS Compared to other population-based and hybrid methods that do not possess an elite pool of high quality and high-diversity solutions. The computational time of the compared methods ranges from 500 to 1500 s. In almost all of the instances, the hybrid elitist-AS obtained better results than those produced by the similar approaches. For example, it has obtained a much better result (e.g., berlin52 = 7542) than the original AS (e.g., berlin52 = 7601); than the enhanced ACS-2opt (e.g., a280 = 2618); and HS (e.g., bier127 = 118,282).

Regarding the consistency verification and as listed in Table 2, the average error Δ (%) for the hybrid elitist-AS stands at 0.43 %. The average running time for the hybrid elitist-AS is 215.5. Experimental results show that hybrid elitist-AS is promising, competitive (efficient and accurate) and comparable when elite pool, local search and diversification and intensification mechanisms are integrated. Experimental results also demonstrate that the elite pool, local search and diversification and intensification mechanisms are very important in the hybrid elitist-AS when searching for the optimal solution. To see the effect in terms of the quality of utilizing an elite pool, the author also implemented a conventional elitist-AS (CEAS) without an elite pool and compared it to the hybrid elitist-AS with an elite pool; please see the last column in the table.

From Table 3, it can be seen that almost across all instances, the hybrid elitist-AS outperforms other methods (e.g., AS, ACS-2opt, GA, GCGA-LS, ISA, TPASHO, IDCS and CEAS). For example, the hybrid elitist-AS (with an elite pool) has obtained a much better result (e.g., berlin52 = 7542) than the conventional AS and CEAS. For all instances, the hybrid elitist-AS obtained competitive results (if not optimal) compared to PTS-ACO, AEAC, HS, IBA and HGA. However, it is outperformed by the PTS-ACO and HGA for 4 instances.

Table 3 Best TSP results obtained by the hybrid elitist-AS compared to similar meta-heuristics

Instance	Elitist-AS	AS	ACS-2opt	PTS-ACO	AEAC	GA	GCGA-LS	HS	ISA	IBA	TPASHO	HGA	IDCS	CEAS	*P* value
att48	**10,628**	N/A	N/A	N/A	N/A	10,782	10,638	**10,628**	N/A	N/A	N/A	N/A	N/A	10,736	+
att532	28,147	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	31,861	+
a280	**2579**	2687	2618	**2579**	N/A	N/A	N/A	N/A	2586	N/A	N/A	2659	N/A	3092	+
bays29	**2020**	2077	**2020**	N/A	**2020**	N/A	N/A	**2020**	N/A	N/A	N/A	N/A	N/A	2022	+
berlin52	**7542**	7601	**7542**	**7542**	N/A	N/A	N/A	N/A	7544	**7542**	N/A	7544	**7542**	7548	+
bier127	**118,282**	N/A	N/A	N/A	N/A	120,978	119,363	118,498	N/A	N/A	N/A	N/A	**118,282**	120,516	+
ch130	**6110**	N/A	N/A	N/A	N/A	N/A	N/A	**6110**	**6110**	N/A	6118	**6110**	**6110**	6265	+
ch150	**6528**	6696	6570	**6528**	N/A	N/A	N/A	6553	N/A	N/A	N/A	6530	**6528**	6759	+
d198	**15,888**	N/A	N/A	N/A	N/A	16,108	15,940	N/A	N/A	N/A	N/A	15,896	N/A	16,066	+
d2103	**80,688**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	80,882	+
d15112	**1,573,162**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1,591,630	+
eil51	**426**	N/A	N/A	N/A	N/A	430	427	**426**	N/A	**426**	**426**	428	**426**	431	+
eil76	**538**	N/A	N/A	N/A	N/A	552	550	**538**	544	539	N/A	544	**538**	551	+
fl1577	**22,977**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	30,102	+
gr24	**1272**	N/A	N/A	**1272**	**1272**	N/A	N/A	**1272**	N/A	N/A	N/A	N/A	N/A	**1272**	~
hk48	**11,461**	N/A	N/A	N/A	**11,461**	N/A	N/A	**11,461**	N/A	N/A	N/A	N/A	N/A	14,541	+
kroA100	**21,282**	N/A	N/A	N/A	N/A	21,554	21,292	**21,282**	21,285	**21,282**	N/A	21,285	**21,282**	22,802	+
kroB150	26,185	N/A	N/A	N/A	N/A	26,698	26,556	N/A	26,130	N/A	N/A	**26,127**	26,130	26,906	+
kroA200	29,420	31,662	29,470	**29,368**	N/A	30,166	29,759	N/A	29,368	N/A	29,475	29,369	29,382	30,478	+
lin318	44,169	43,651	42,086	**42,029**	N/A	43,607	43,600	N/A	N/A	N/A	N/A	42,624	42,125	45,716	+
pcb442	51,286	53,469	51,790	**50,778**	N/A	52,923	54,670	N/A	N/A	N/A	N/A	52,874	N/A	58,231	+
pr76	**108,159**	N/A	N/A	N/A	N/A	109,132	108,308	**108,159**	108,159	N/A	108,219	**108,159**	**108,159**	109,729	+
rat575	**6773**	N/A	N/A	**6773**	N/A	N/A	N/A	N/A	N/A	N/A	6858	N/A	N/A	7579	+
rat783	**8806**	8950	8815	**8806**	N/A	N/A	N/A	N/A	N/A	N/A	8949	N/A	N/A	9554	+
rd100	**7910**	8603	8159	**7910**	N/A	8203	7986	**7910**	N/A	N/A	N/A	**7910**	N/A	8328	+
rd400	**15,281**	N/A	N/A	N/A	N/A	15,933	16,215	N/A	N/A	N/A	15,333	16,049	N/A	17,413	+
st70	**675**	752	722	**675**	N/A	683	**675**	**675**	677	**675**	N/A	677	**675**	**675**	~
tsp225	3919	N/A	4102	3919	N/A	N/A	N/A	N/A	3920	N/A	N/A	**3878**	N/A	4007	+
vm1084	**239,297**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	241,758	+

All approaches in Table 3 reported similar running times as the running time of the hybrid elitist-AS reported in Table 2

CEAS conventional elitist-AS developed in this study, *AS* ant system [20], *ACS-2opt* two-stage ant colony system and 2opt [20], *PTS-ACS* pheromone two-stage ant colony system [21], *AEAC* accumulated experience ant colony [15], *GCGA-LS* generalized chromosome genetic algorithm and local search [29], *GA* genetic algorithm [29], *HS* harmony search [3], *ISA* improved simulated annealing [28], *HGA* hybrid genetic algorithm with two local search methods [27], *IBA* improved discrete bat algorithm [16], *TPASHO* 2-phase ant colony optimization and simulated annealing [2], *IDCS* improved discrete cuckoo search [18]

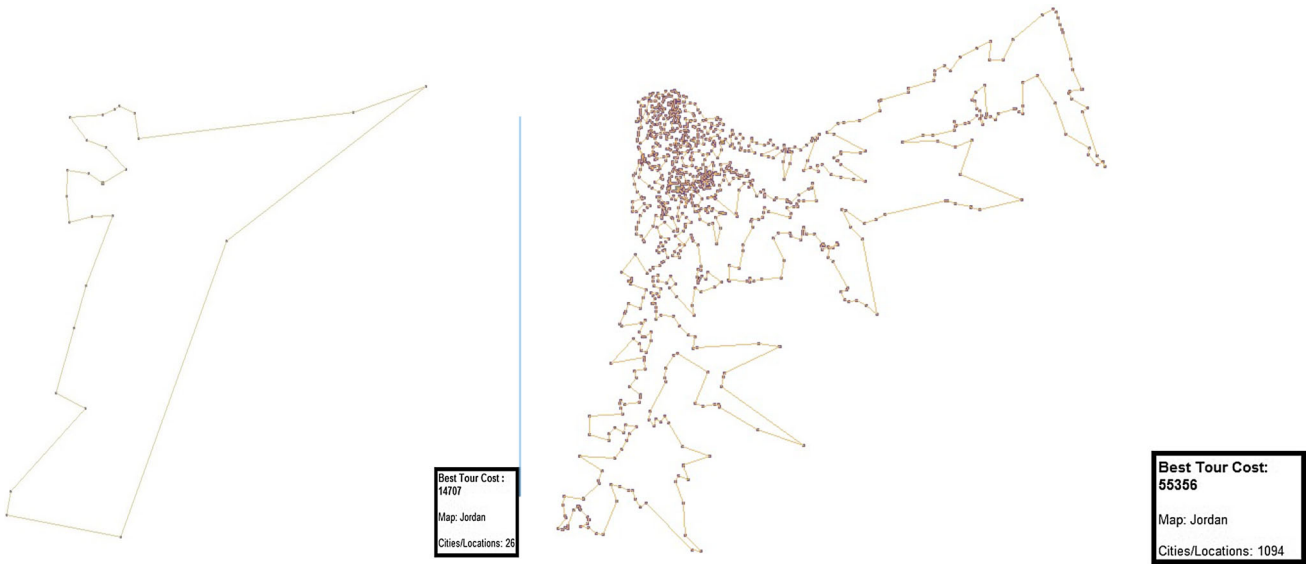


Fig. 5 Graphical representation of the shortest tour and coordinates instances *jorE26* and *jorE1094*

All of the methodologies reported in the table above did not use an explicit memory. This may cause the lack of maintaining a balance between diversity and quality of the search. They also lack a systematic selection strategy or a solution combination strategy, where this might be the reason behind their poor results or their ability to tackle only one specific problem. On the other hand, some works reported in the literature used a memory to store the best solution (such as the ant colony algorithm in [13]), but unfortunately they did not provide a clear description of the memory and a few experimental results.

From the table, for most instances, it can be clearly seen that the proposed methodology is better than those without elite pools (e.g., see *att48*). This demonstrated the effectiveness of using an elite pool (of diverse and high-quality solutions) in a population-based method. Some of the results obtained are exactly the same as others (e.g., see *ch150*). As mentioned before, the elitist-AS meta-heuristic possess an implicit memory to store high quality and diverse solutions; yet, it could be exhaustive to directly apply permutations and perturbations, e.g., apply specific neighborhood structures, to good quality or diverse solutions for more quality improvements. It is believed that the transition between implicit and explicit solution representations during the search process is a hard task. Generally, the hybrid elitist-AS is fairly effective and consistent.

In addition to the results above, it is worthwhile to draw some statistically significant conclusions regarding the performance of elitist-AS and the CEAS. Therefore, the Wilcoxon test (pairwise comparisons) with significant level of 0.05 is performed. The p value of the Wilcoxon test of elitist-AS versus CEAS is presented in the last column of Table 3, where "+" indicates elitist-AS is statistically better than CEAS (p value < 0.05), "−" indicates elitist-AS outperformed by CEAS (p value > 0.05) and "$\sim$" indicates both algorithms having the same performance (p value $= 0.05$). The results in Table 3 (last column) show that elitist-AS is statistically better than CEAS on 27 instances, not statistically better than CEAS on none of the instances and perform the same as CEAS on 2 instances out of 29 tested instances of the considered problem domain.

To summarize, the results demonstrate that elitist-AS is better than CEAS in terms of consistency, efficiency and generality with regards to the tested instances of the considered problem domain. This is mainly due to the use of elite pool within elitist-AS, which has a positive effect on the ability of elitist-AS in producing good quality and consistent results compared to CEAS. The *Std.* and Δ (%) of elitist-AS reveal that its results are stable and very close to the best results obtained by other population-based meta-heuristic methods. All of the observations above serve as evidence that the elitist-AS is capable of producing good quality results over all instances, instead of just a few ones.

As shown in this experimental study, the hybrid elitist-AS obtained competitive results, if not optimal ones in some instances, when compared against the best known results reported in the literature. The percentage deviation demonstrates that the results of the hybrid meta-heuristic are very close to the best known ones. This is proven based on the hybrid elitist-AS' consistency and generality. This is due to the factor of incorporating (hybridizing) an explicit memory structure (e.g., reference set) in a meta-heuristic to diversify the search by exploring different regions of the search space, or rather by escaping local optima, while preserving high-quality solutions. Overall, the result implies that hybridizing an explicit memory structure with

a population-based meta-heuristic has a significant impact on the performance of population-based meta-heuristics in solving *STSP*.

It is interesting to consider the computational complexity of the algorithm in order to know which portion is most compute—intensive in the proposed hybrid elitist-AS meta-heuristic. Based on the analysis, the computational complexity of the search in the proposed hybrid meta-heuristic is not high when compared to most meta-heuristics. This presents another benefit of the hybrid elitist-AS—i.e., the general and computational efficiency for a range of problems. To provide some indication where most of the time is being spent, the computational complexity of the hybrid meta-heuristic is briefly analyzed below:

1. The overall computational complexity for the elitist-AS meta-heuristic (without hybridization) is $O(p \times n)$, where p is the population size, and n is the number of iterations.
2. However, the proposed hybrid meta-heuristic starts by initializing the population of initial solutions, Initialization $= O(m^2 + p)$, where p is the population size, and m is the dimension of the solution.
3. A constructive heuristic is employed in the population initialization, nearest neighbor with complexity $O(n^2)$. Regardless a specific type of constructive heuristics, generally the complexity is $= O(m^2 \times p)$.
4. Next, the algorithm generates a population of solutions for the adaptive memory mechanism (e.g., external memory or elite pool) which has complexity $O(s \times m)$, where s is the memory size, and m is the dimension of the solution (e.g., number of cities).
5. The proposed hybrid meta-heuristic will then employ a local search routine, iterated local search with complexity $O(n)$, to generate new better quality solutions.
6. The hybrid meta-heuristic will call a local search routine every n iterations (every n consecutive non-improvement iterations) to generate a new population from scratch to re-initialize the search as follows:

(a) Selection $= O(m)$.
(b) Perturbations $= O(m - 1)$.
(c) Deposit pheromone trail $= O(m^2 \times p)$.
(d) Update pheromone trail $= O(m^2)$.

7. The hybrid meta-heuristic will update the adaptive memory mechanisms (e.g., external memory or elite pool) and sort the solutions of the adaptive memory mechanism, which has the complexity $O(s \log(s))$.
8. The search is updated by the elitism. The search will examine solutions in the memory whether to loop or to stop:

(e) Elitism $= O(m \times p^2)$.
(f) Loop or Exit (stopping criterion) $= O(m \times p)$.

Thus, the overall complexity for the proposed hybrid meta-heuristic is $O(m^2 + p) + O(m^2 \times p) + O(s \times m) + O(n) + O(m - 1) + O(m^2 \times p) + O(m^2) + O(s \log(s)) + O(m \times p^2) + O(m \times p)$. Note that $O(m^2 \times p)$ can be replaced by $O(n^2)$.

Intentionally, the author has not simplified the complexities as he wishes to draw out the complexity of the various stages. Since the proposed hybrid meta-heuristic has been tested and uses a variety of local search routines, the complexity of the utilized local search routine is not presented in detail. By inspection, it can be observed that the complexity of the local search routine ($O(n)$) is very low, e.g., *w.r.t.* the parameters of n, m and p, etc., therefore do not present to be a critical factor in the proposed hybrid meta-heuristic. The crucial factors are presented in the memory structures (external memory or elite pool) and the search update (see Steps 4 and 7, Steps 6 and 8, respectively). In some cases, constructive heuristics could be presented as crucial factors too (see Step 3).

Overall, the advantages of the hybrid elitist-AS are the ability to utilize the heuristic information about diverse and high-quality solutions during instance solving—via elite pool—to diversify the search while intensifying the improvement of a high-quality solution. Results demonstrate that the hybrid method provides a general mechanism regardless the nature and complexity of the instances and could be applied to other domains without many changes (i.e., the user only needs to change the constructive heuristics and neighborhood structures). Applying a methodology to other problem domains or even different instances of the same problem usually requires a considerable amount of modification (e.g., change algorithm parameters or structures). The study demonstrated the generality of the proposed hybrid method across different instances. It would be hoped that the proposed methodology would also generalize to other domains.

The performance of the hybrid meta-heuristic is evaluated by considering the following three criteria:

- *Generality* Generality is defined as the ability of the hybrid method to work well across different instances of the same problem.
- *Consistency* The ability of the hybrid method to produce stable results when executed several times for every instance. Typically, consistency is one of the most important criteria in evaluating any algorithm. This is due to the fact that many search algorithms have a stochastic component, prompting thus different solutions over multiple runs even if the initial solution is the same. The consistency of the hybrid method is

measured based on the average and the standard deviation over 25 independent runs.
- *Efficiency* The ability of the hybrid method to produce good results that is close or better than the best known value in the literature. The efficiency of the hybrid method is measured by reporting, for each instance, the best and the percentage deviation, Δ (%), from the best known results in the literature.

For all tested instances, results of the hybrid method are compared to similar methods in terms of solution quality rather than computational time. This can be attributed to the fact that different computer resources used made the comparison very difficult. Therefore, in this study, the number of iterations is set as the termination criteria as a result of the use of the adaptive memory (e.g., 20 min) in the hybrid elitist-AS. As a result, the execution time of the hybrid method is within the range of those published in the literature.

Hence, the innovation of the article can be highlighted as the hybrid elitist-AS is promising, competitive (efficient and accurate) and comparable when elite pool, iterated local search, diversification and intensification mechanisms are integrated.

7 Conclusion

This study demonstrated the efficiency and consistency of a hybrid population-based meta-heuristic (namely elitist-AS) on the *STSP*. This was achieved by investigating and testing the impact of hybridization (of intensification and diversification mechanisms) and utilization (of an elite pool) on the performance of a population-based meta-heuristic. The hybrid elitist-AS utilizes an elite pool (namely external memory) consisting of a collection of diverse and high-quality solutions. This memory structure helps in maintaining a balance between diversity and quality of the search. For example, escaping local optima (minima) can be made through the employment of generating new solutions from those diverse ones in the elite pool. This way, the search might be diversified for new promising region. In addition, the search can be converged toward better quality solutions by focusing the search around good quality solutions from the elite pool.

Results show that the hybrid elitist-AS produces high-quality solutions, if not optimal, and that their performance are (or can be considered) consistent, efficient and effective across all instances of the TSP (including the generated 26 cities and 1094 Jordanian locations). The study concludes that the hybridization of diversification and intensification mechanisms and an elite pool within a population-based meta-heuristic can enhance the performance of the population-based meta-heuristic toward producing high-quality solutions (which are optimal). The analysis and discussion on the computational complexity are another proof that the hybrid elitist-AS is efficient across a range of instances. The generated instances for the 26 Jordanian cities and 1094 locations could be considered for further complexity and modification (e.g., upgrading it to be a vehicle routing problem).

As an extension of a previously reported work, the author highlights the importance of the hybrid elitist-AS and elite pool, diversification and intensification mechanisms, as well as the local search for optimal solution searching, which is compared with classical or hybrid ACOs. Numerical experiments show that hybrid elitist-AS with elite pool, diversification and intensification mechanisms, and local search are very competitive and have a good trend with regard to implementation efficiency when compared to the classical ACOs and as it is applied to larger size TSP instances.

The main contributions of this work are as follows:

- The development of the hybrid elitist-AS method that possesses elite pool and performs heuristic perturbations, demonstrating that strengths of different search algorithms can be merged into one hybrid population-based meta-heuristic methodology (e.g., constructive heuristics and meta-heuristics, population-based and local search methods).
- The hybridization of an adaptive memory mechanism (e.g., elite pool) that contains a collection of high quality and diverse solutions, with a population-based meta-heuristic, and manages to obtain consistent results in addition to producing high-quality solutions that are either competitive or better than other similar methods in some cases.
- The development of a hybrid meta-heuristic which can be easily applied to different instances without much effort, e.g., the user only needs to change some parameter or neighborhood structures.

In particular, the use of an elite pool provides a diversity of high-quality solutions from which the hybrid elitist-AS meta-heuristic can start its search process for better solutions. The elite pool also provides a mean of implement cooperation and to achieve faster convergence. In future work, the author intends to investigate the effectiveness of the hybrid meta-heuristic across other COPs.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest.

References

1. Albayrak M, Allahverdi N (2011) Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. Expert Syst Appl 38(3):1313–1320

2. Bao H (2015) A two-phase hybrid optimization algorithm for solving complex optimization problems. Int J Smart Home 9(10):27–36. doi:10.14257/ijsh.2015.9.10.04
3. Bouzidi M, Riffi ME (2014) Adaptation of the harmony search algorithm to solve the travelling salesman problem. J Theor Appl Inf Technol 62(1):154–160
4. Blum C, Roli A (2008) Hybrid metaheuristics: an introduction, studies in computational intelligence. In: Blum C, Aguilera MJB, Roli A, Samples M (eds) Hybrid metaheuristics: an emerging approach to optimization, vol 114. Springer, Berlin, pp 1–30
5. Dong G, Guo W (2010) A cooperative ant colony system and genetic algorithm for TSPs. ICSI 1:597–604
6. Dorigo M, Di Caro G, Gambardella LM (1999) Ant algorithms for discrete optimization. Artif Life 5:137–172
7. Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
8. Dorigo M, Stützle T (2010) Ant colony optimization: overview and recent advances. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, international series in operations research & management science, Chapter 8, vol 146. Springer, Berlin, pp 227–263
9. Glover F, Laguna M, Martí R (2002) Scatter search. In: Ghosh A, Tsutsui S (eds) Theory and applications of evolutionary computation: recent trends. Springer, Berlin, pp 519–529
10. Greistorfer P, Voß S (2005) Controlled pool maintenance in combinatorial optimization. In: Rego C, Alidaee B (eds) Conference on adaptive memory and evolution: tabu search and scatter search, Chapter 18, University of Mississippi, Kluwer Academic Publishers, pp 387–424
11. Jaradat G, Ayob M (2010) An elitist-ant system for solving the post-enrolment course timetabling problem. In: Zhang Y et al (eds) DTA/BSBT 2010, vol 118. Springer, Heidelberg, pp 167–176
12. Karaboga D, Gorkemli B (2011) A combinatorial artificial bee colony algorithm for traveling salesman problem. In: 2011 international symposium on innovations in intelligent systems and applications (INISTA), pp 50–53
13. Li B, Wang L, Song W (2008) Ant colony optimization for the traveling salesman problem based on ants with memory. In: Fourth international conference on natural computation, IEEE 978-0-7695-3304-9/08. doi:10.1109/ICNC.2008.354
14. Marinakis Y, Marinaki M (2010) A hybrid genetic—particle swarm optimization algorithm for the vehicle routing problem. Expert Syst Appl 37:1446–1455
15. Montgomery J, Randall M (2003) The accumulated experience ant colony for the travelling salesman problem. Int J Comput Intell Appl 3(2):189–198
16. Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Eng Appl Artif Intell 48:59–71
17. Osman IH, Kelly JP (1996) Meta-heursitic: an overview. In: Osman IH, Kelly JP (eds) Meta-heuristic: theory and applications. Kluwer Academic Publishers, Berlin
18. Ouaarab A, Ahiod B, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24:1659–1669
19. Pantrigo JJ, Duarte A, Sánchez Á, Cabido R (2005) Scatter search particle filter to solve the dynamic travelling salesman problem. In: GR Raidl, J Gottlieb (eds) EvoCOP 2005. LNCS 3448:177–189
20. Puris A, Bello R, Martinez Y, Nowe A (2007) Two-stage ant colony optimization for solving the traveling salesman problem. Mira J, Alvarez JR (eds) IWINAC 2007, Part II. LNCS 4528:307–316
21. Puris A, Bello R, Herrera F (2010) Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: case of study with TSP and QAP. Expert Syst Appl 37:5443–5453
22. Reinelt G (1991) TSPLIB-a travelling salesman problem library. ORSA J Comput 3:376–384
23. Reinelt G (1995) The TSPLIB symmetric traveling salesman problem instances. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html
24. Rodriguez A, Ruiz R (2012) The effect of the asymmetry of road transportation networks on the traveling salesman problem. Comput Oper Res 39(7):1566–1576
25. Szeto WY, Wu Y, Ho SC (2011) An artificial bee colony algorithm for the capacitated vehicle routing problem. Eur J Oper Res 215(1):126–135
26. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, USA
27. Wang Y (2014) The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Comput Ind Eng 70:124–133
28. Wang Y, Tian D, Li YH (2013) An improved simulated annealing algorithm for travelling salesman problem. Int J Online Eng 9(4):28–32
29. Yang JH, Wu CG, Lee HP, Liang YC (2008) Solving traveling salesman problems using generalized chromosome genetic algorithm. Prog Nat Sci 18:887–892