Soft Computing (2021) 25:3775–3794
https://doi.org/10.1007/s00500-020-05406-5

METHODOLOGIES AND APPLICATION

Bi-heuristic ant colony optimization-based approaches for traveling salesman problem

Nizar Rokbani[1,4] · Raghvendra Kumar[3] · Ajith Abraham[2] · Adel M. Alimi[4] · Hoang Viet Long[5,6] · Ishaani Priyadarshini[7] · Le Hoang Son[8,9]

Published online: 4 November 2020

Abstract
Heuristic computational intelligence techniques are widely used in combinatorial optimization problems, essentially in large size configurations. Bio-inspired heuristics such as PSO, FA or FPA showed their capacities to solve such problems. Bi-heuristic optimization consists of using a couple of techniques and a collaboration mechanism. This paper reviews the major contributions in solving TSP with bi-heuristics and presents a new hybridization scheme based on FPA, ACO with Ls, ant supervised by flower pollination with local search, ASFPA-Ls; as well as the impact of social and cognitive PSO for the ant supervised by PSO with local search, ASPSO-Ls. AS-chaotic-PSO-Ls which stands for ant supervised by chaotic PSO local search is also investigated. The meta-heuristic algorithms (FPA, PSO or chaotic PSO) and ant colony optimization are used with a hierarchical collaboration schema in addition to a local search mechanism. In this work, the local search strategy, Ls, used is the 2-opt method; the proposals are called, respectively, ASFPA-Ls, cognitive ant supervised by PSO with local search, Co-ASPSO-Ls, social-ASPSO-Ls and AS-chaotic-PSO-Ls; where ACO is coupled with a local search heuristic mechanism called 2-opt, and a set of meta-heuristics, FA, FPA and PSO asked to adapt ACO parameters while running. Comparative experimental investigations are conducted using the TSP test bench. Proposed hybridizations attended fair solutions for the TSP problems used in the experimental investigations including berlin52, St70, eil76, rat 99, eil101, KroA100, Ch150 and kroA200. A good balance performance/time is found with the social ant supervised by PSO with local search, So-ASPSO-Ls. The cognitive ant supervised by PSO with local search, Co-ASPSO-Ls comes in the second position in terms of time effectiveness with close performances to AS-chaoticPS0-LS, all proposed approaches returned low rate errors or BKS for used test benches: berlin52, st70, eil76, rat99, kroA100 and kroA200.

Keywords Bio-inspired heuristics · Simplified PSO · Local search (ls) · Ant supervised by PSO (ASPSO) · Traveling salesman problem (TSP)

Communicated by V. Loia.

✉ Hoang Viet Long
hoangvietlong@tdtu.edu.vn

Nizar Rokbani
nizar.rokbani@ieee.org

Raghvendra Kumar
raghvendraagrawal7@gmail.com

Ajith Abraham
ajith.abraham@ieee.org

Adel M. Alimi
adel.alimi@ieee.org

Ishaani Priyadarshini
ishaani@udel.edu

Le Hoang Son
lehoangson2@duytan.edu.vn

Extended author information available on the last page of the article

1 Introduction

The optimization problem consists of solving problems by minimizing or maximizing their functions (Dhiman 2019). Each function can generate many possible solutions. In optimization problems, we aim to search for the closest optimum, which respects specific constraints. An optimization problem can be defined as a group of parameters where the correct ones determine the optimal solution. The traveling salesman problem (TSP) is a combinatorial optimization problem in which a salesman aims to visit a set of cities and reach back to his starting locality. It is required to pass each city once at a minimum cost. In general, the tour length stands for cost of a solution; it is the total distance that the salesmen have to cross. The relevance of heuristic and bio-inspired solvers for such a problem comes from the fact that such techniques are able to find out an acceptable solution with respect to time/performance balance. The main bio-inspired swarm techniques are flower pollination algorithm (Yang 2013), particle swarm optimization (Kennedy and Eberhart 1995), ant colony optimization (Dorigo and Birattari 2007), firefly algorithms (Yang 2010), bee algorithm (Karaboga 2005; Karaboga and Gorkemli 2011) or bat algorithm (Saji and Riffi 2016).

FA, firefly algorithm, is typically a continuous optimization heuristic, discrete variant was then proposed for the combinatorial optimization problems such as traveling salesman problem, TSP. To get initial solutions, authors use a uniform distribution in the search space. For initial solution, the algorithm starts with a greedy or nearest neighborhood algorithm. Initial random generations help the algorithm to find good solutions. To construct the tour, two phases are needed: distance function and movement. The first one consists of calculating distance between two fireflies (permutation), two rules to calculate distance are proposed which are Hamming and swap distance. The closer permutations are solutions with small fitness function. Fireflies are modeled with a permutation schema. To represent the distance between two fireflies, authors use the mutation inverse. New paths are built without missing the previous ones (Kumbharana and Pandey 2013). FA with greedy approach was proposed in Saraei et al. (2015) to solve the TSP and where the greedy algorithm was inserted to increase the quality of solutions. In Wang et al. (2016), integrated the 2-opt algorithm, which is a local search algorithm, into the firefly to increase the algorithm accuracy and accelerate the convergence. In addition to the combing method and the use of 2-opt, the article introduces a new fluorescent brightness mechanism. A combination of firefly and simulated annealing was proposed in (Nekouie and Yaghoobi 2015) in order to merge the advantages of both heuristics consisting of searching for a global optimum while considering the local one. A combination of ACO and FA appears in Olief et al. (2016) in which ACO search for the global solution and the FA focuses on local optimums using its neighborhood mechanism. FA is the first to search for local best tours then the ACO is involved in searching for the best tour ever found from the obtained FA results.

In Ariyaratne et al. (2016), the authors used the firefly algorithm to optimize ACO settings. They aim to find the global best path by optimizing α, β and ρ. This proposed approach is applied to the symmetric traveling salesman problem. The FA algorithm is integrated into the ACO to tune its parameters. Tsai et al. (2004) used discrete firefly to solve the distributed network configuration problem. To optimize the best tour, a modified FA-based on fuzzy control was proposed in (Bidar and Kanan 2013). In Rizk-Allah et al. (2013), a novel hybridization method is based on ant colony optimization and firefly algorithm. This approach integrates two phases: The first one is running ACO to find initial solutions by the group of ants, while the second one is running FA to find the best ACO inputs. With fireflies' population equal to ants group size, they obtain the same solution number as the previous one. Firefly algorithm and PSO are approaches proposed in Kora and Rama Krishna (2016) to optimize the detection of bundle branch block. To assess the algorithm performance, authors use MIT-BIH database for their tests. One of the most FAPSO features is to not consider local optima. FA is integrated into ACO to solve TSP problem. In Taengtang et al. (2013), authors changed the pheromone equation basing on FA algorithm where the substrate of the first algorithm (ACO) plays the role of attractiveness in the second one (firefly). In (Twir et al. 2018), an ant supervised by firefly with local search mechanism is proposed, where the ACO is in charge of solving the TSP, the FA is in charge of optimizing ACO parameters and the 2-opt local search mechanism is used to avoid local optimum (Kaveh and Ghazaan 2019).

FPA was combined with several techniques and it shows its efficiency. In AL-Wagih (2015), used a proposed method based on flower pollination algorithm and chaos theory to solve an integer programming problem and chaotic maps to optimize the FPA variable. The authors proved the proposed algorithm performance to solve the IP system as an NP-hard problem (IFPCH). The proposed method shows better solution than the standard FPA. It helps the algorithm to ignore the local solution and run to find the global one. A modified flower pollination is proposed in Nabil (2016) based on hybridization between FPA and the clonal selection theory (CSA). CSA consists to form copies from the highest affinity antibodies to improve the algorithm efficiency; the proposed method is applied to

23 test benches. The modification is appeared in the pollination local search process using the local search coefficient. In Zhang et al. (2016), an improved flower pollination algorithm is proposed by Zhang. This algorithm consists to combine Adaptive Gauss Mutation and Shuffled Frog Leaping to ignore the local search optimum and the late convergence. This combination enhances the swarm diversity and increases the local search ability. This algorithm shows a high ability in looking for a global solution in addition to a faster convergence with precise solutions. In order to solve FPA problems, authors use the Gauss mutation to accelerate the search and ameliorate the solution quality. The proposed method is applied to four test benches in which it finds higher quality solution more than the standard FPA; the algorithm has a lower time consuming**.** In Valenzuela et al. (2017), a proposed method based on the flower pollination algorithm and the fuzzy inference system aims to enhance the ability of exploitation and exploration by inserting a fuzzy system to control the switching probability among the two behaviors' of FPA; it was then applied to solve set of mathematical test functions and showed better results compared to ACO, PSO and GA.

The bat algorithm, BA, is also a bio-inspired technique inspired from natural bat swarms, where all bats are assumed to use eco-localization and that they are moving by updating their positions in random way using a velocity (vi) and a frequency (fi) (Yang and Gandomi 2012). The BA is assumed to outperform the classical continuous optimization techniques such as PSO or GA (Mirjalili et al. 2014). In binary bat algorithm, the position update leads to a switching between (0) and (1) values (Mirjalili et al. 2014). A binary BA version was proposed in Rizk-Allah and Hassanien (2018b) where authors combined a rough set scheme (RSS) binary bat algorithm, BBA, with a local search strategy, LSS. The new proposal is applied for the 0–1 knapsack problems which is a combinatorial optimization one. A discrete BA, DBA, was proposed by Saji and Riffi (2016) with application to TSP. In DBA, bats select a frequency within an integer interval [1, n] where (n) is the number of cities of a sub-tour and where positions and velocities are obtained by a crossover operator instead of the classical bat velocity equations.

Hybrid schema used is also to be investigated, hybrid methods are based on a hybridization of a limited set of bio-inspired techniques to get the advantages of each one or to avoid the weaknesses of another (Elloumi et al. 2009). The bi-heuristic optimization consists of combining two or more heuristic techniques in order to solve a problem. Combining techniques needs a hybridization schema and is in general focused on global optimality criteria. In this study, the focuses are made on proposing a hybridization aiming to ensure to a heuristic to be self-adaptive for combinatorial discrete problems. It consists of defining a tuning strategy allowing resuming the heuristic dependence to its own parameters. Basically, the proposed schema is combining a discrete heuristic with continuous one; the discrete heuristic is naturally suitable to handle combinatorial optimization problems such as the TSP, the continuous one is handling the first heuristic parameters. This hybridization allowed the proposal to be self-adaptive. A classical hybridization model involving PSO and ACO was proposed in Rokbani et al. (2013a), where the ACO was asked to solve the TSP while PSO used to adapt ACO parameters the hybridization concerned inertia weight PSO with ACO, the simplified PSO-ACO and fuzzy PSO-ACO (Rokbani et al. 2013a), meanwhile early applications used to be done on a local TSP test bench of Tunisia. The same hybridization schema was then used in Mahia et al. (2015) as well as in Kefiet al. (2016a, b), where at the difference of Rokbani et al. (2013a, b) a local search mechanism was added to the main hybridization schema with a comparative application to a set of standard TSP instances, the impact of swarm size for AS-PSO was also investigated in Kefi et al. (2016). In Twir et al. (2018), the fuzzy PSO-ACO local search, as well as the simplified PSO-ACO, with same idea while with a different PSO variant was investigated in Twir et al. (2017), the focus was made on simplified PSO with application to classical TSP instances; in Rokbani et al. (2019a), investigated the fuzzy PSO-ACO hybridization while the gravitational PSO-ACO was subject to a comparative study in Rokbani et al. (2019b). A hybridization between the firefly algorithm, FA, and ACO was proposed in Twir et al. (2018) where the local search mechanism was also used.

Meanwhile and regardless of the selected meta-heuristics or bio-inspired technique to solve a problem, the performances are always affected by the algorithms' parameters; in Yang et al. (2013), authors pointed this problem as a common problem to any meta-heuristics technique and proposed a framework to limit the impact of the parameters on performances. This paper investigates the same field while proposing to combine a couple of heuristics where one is in charge of the problem and the other is in charge of parameters fitting, this may allow to reduce the dependencies between algorithm performances and algorithm parameters.

This paper presents four new hybridization schema based on ant supervised by flower pollination, local search algorithm, ASFPA-Ls, the cognitive ant supervised by PSO with local search, Co-ASPSO-Ls, in which a cognitive PSO is used to supervise the self-adaptation of the ACO parameters. The So-ASPSO-Ls is a hybridization where a social PSO is used to self-tune ACO and the chaotic-ASPSO-Ls, where a chaotic PSO is introduced to self adapt ACO. The proposed hybridizations are comparatively evaluated based on a set of TSP instances.

The remaining of the paper is organized as follows; paragraph two presents the techniques and methods used in the paper, the proposed hybridizations are detailed in paragraph three. The experimental investigations are presented in paragraph four with comments and discussions prior to conclusions and perspectives.

2 Techniques and methods

This paragraph starts with the traveling salesman problem statement then details the main heuristics used in this paper with focus on ACO, FA, FPA, SPSO and 2-opt local search mechanism.

2.1 The traveling salesman problem

TSP, traveling salesman problem, is a mathematical problem that belongs to NP-hard problems consisting of visiting a set of fixed cities once, each one with respect to the shortest path. The salesman has to start from a given city and then to move back to it. The TSP could be addressed as a problem, and also as comparative test bench to measure solvers' capacities. A mathematical formulation of TSP could stand as follows in Eqs. (1), (2) and (3).

$$D = \sum_{i=1,j=2}^{N} \sum_{i \neq j}^{N} x_{ij} c_{ij} \tag{1}$$

$$x_{ij} = \left\{ \begin{array}{r} 1 \quad \text{if the path goes from city} i \text{to city} j \\ 0 \quad \text{otherwise} \end{array} \right\} \tag{2}$$

$$c_{ij} = \left\| x_i - x_j \right\| \tag{3}$$

where (d) stands for tour distance, which is also used as a fitness function for CSPO-ACO, (N) is the number of cities for given problem instance; x_{ij} is a binary coefficient; equal to 1 if an arc is going from city i to city j and equal to 0 otherwise, see Eq. (2). The distance between city i and city j is a 2D Euclidean distance, see Eq. (3). Equations (4) and (5) ensure that each node (city) is visited only one time.

$$\sum_{j=1,i \neq j}^{N} x_{ij} = 1, (i = 1, 2, 3, \ldots, N) \tag{4}$$

$$\sum_{i=1,i \neq j}^{N} x_{ij} = 1, (j = 1, 2, 3, \ldots, N) \tag{5}$$

2.2 Ant colony optimization

It is a heuristic method introduced by Marco Dorigo (2007). It showed a capacity in solving the TSP (Dorigo and Gambardella 1997a). The heuristic based on the natural ant's behavior in food search using a biologic marker. Agents use the pheromone to communicate with other ants and transmit information about the shortest path passed. TSP fitness is to find the best global tour passing all cities and return to the start point. Ants have to generate short tours using information accumulated during their search. Ant exploits the trail deposit in the path to choose which country will be visited from the current one. Ants prefer to choose a city in which its incident has a lot of pheromones. Initially, ants selected random cities, and then iteratively, ants update their pheromone until a tour is complete. Finally, ant having the shortest path updates its global tour, GT, using the equation of pheromone trail update (7). To evaluate the probability to engaged in a path from node (i) to node (i),$P_{i,j}^{k}$ Eq. (6) is used; where $\tau_{i,j}$ stands for the pheromone quantity between nodes i and j, Ω_i denotes i_{th} neighborhood, α, β are control parameters of the pheromone. $P_{i,j}^{k}$ stands for the probability that the ant(k) will be crossing the arc (i,j), see Fig. 1.

$$P_{i,j}^{k} = (\tau_{i,j}^{k-1})^{\alpha} * \eta_{i,j}^{\beta} + \sum_{j \int \Omega i} (\tau_{i,j}^{k-1})^{\alpha} * \eta_{i,j}^{\beta} \tag{6}$$

Ant has to choose the highest probability between $P_{i,o}^{k}$, $P_{i,j}^{k}$, $P_{i,l}^{k}$ and $P_{i,m}^{k}$ to pass for the current city i to another one (j, l, m, o) as in Fig. 1. In their food search process, ants start by their locality, here assumed to a city, move from a node to another and move back to the starting one, using the shortest path. Ants select the next node to be visited using a probabilistic approach, see Fig. 1. The pheromone acts as a marker helping in search space exploration, it gives a global map of the most used path on the search space, allowing a local optimum avoidance. To update the pheromone ants, use Eq. (7) where ρ is the pheromone decay coefficient as in Dorigo and Gambardella (1997b).

$$\begin{array}{l} \text{if}(i,j) \in \text{BestTour} \tau_{ij} = (1-\rho)\tau_{ij}^{(k-1)} + \rho \Delta_{ij}^{k} \\ else \tau_{ij}^{(k-1)} = \tau_{ij}^{(k-1)} \end{array} \tag{7}$$

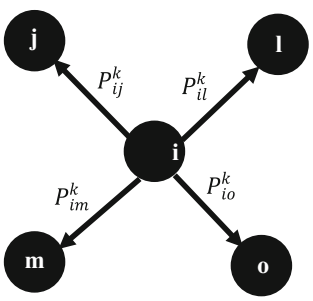


Fig. 1 Stochastic ant strategy

2.3 Flower pollination algorithm

FPA is a bio-inspired technique proposed by Xin-She Yang (2013), based on flower plants reproduction process where the pollen needs to be transferred from the reproduction system of a plant to another, cross-pollination or itself, self-pollination mainly by a pollinator which can be an insect or a bird assumed to move randomly using a Levy flight (AL-Wagih 2015). This process is called biotic pollination and is assumed to a global search mechanism. The abiotic process occurs essentially when a plan is self-pollinated and is assumed to be a local search mechanism. The FPA assumes also that reproduction probability is proportional to the similarity of the involved flowers. This means that if two flowers are similar, flower constancy is changed using reproduction probability. To move from the local search to the global search, a switching probability is used $\rho \in [0, 1]$. To move from a plant to another (global search), pollinators use Eq. (8) where x_j^t and x_k^t are two solutions of j and k plants, respectively. ε is a random value.

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{8}$$

For the local search, the pollinators use Eq. (9) where x_i^t is the solution of ith plant at step t, L is the strength of pollen and x_{gbest} is the current best solution.

$$x_i^{t+1} = x_i^t + L\left(x_i^t - x_{gbest}\right) \tag{9}$$

2.4 Particle swarm optimization

PSO is a well-known heuristic introduced by Eberhart and Kennedy (1995). It is based on miming the social intelligence of animals societies such as bird flocks or fish banks, where each individual is assumed to a particle with a very limited task to perform, moving, and with a simple communication facility allowing it to get informed about the best solution found by group and also about the best solutions in its own neighborhood.

Iteratively, the particles' positions in the swarm are updated according to Eqs. (10) and (11), where w stands for inertia weight, c1 and c2 are, respectively, the cognitive and the social coefficients,P_{lbest}, P_{gbest} are the best local and global positions.

$$v_{i+1} = wv_i + C1 * rand() * (P_{lbest} + x_i) + C2 * rand() * \left(P_{gbest} + x_i\right) \tag{10}$$

$$x_{i+1} = x_i + v_{i+1} \tag{11}$$

The local best solution is fixed according to a topology allowing fixing the size and the organization of the neighborhood; in this paper, a random selection of particles is used to fix it. The neighborhood size is fixed to the 20% of the size of the swarm.

The cognitive and social factors, c1 and c2, control the global behavior of the swarm, when c1 > c2, a cognitive behavior is observed, such a behavior is supposed to strengthen the exploration of the search space. When c2 > c1, the swarm tends to have more cohesion and focus on global optimum.

2.5 Simplified PSO

The simplified PSO was proposed by Pedersen and Chipperfield (2010); it is a PSO where the cognitive behavior was simply neglected and the local best is ignored; this simplifies PSO processing since the search for a local solution is simply removed. Particles' positions update is operated using Eqs. (11) and (12).

$$v_{i+1} = wv_i + C2 * \text{rand}() * \left(P_{\text{gbest}} + x_i\right) \tag{11}$$

$$x_{i+1} = x_i + v_i \tag{12}$$

A hybridization of ACO with simplified PSO and a local search mechanism was detailed in Rokbani et al. (2019a, b).

2.6 Chaotic PSO

Chaotic particle swarm optimization, CPSO, is consisting of using the classical PSO with the chaos search process to perturb the best solution found by PSO. The chaotic search algorithm, CSA, consists of setting a bounded search space around a good solution; the bounded search space is defined using a chaotic map applied to a given solution, in order to see if a better one is just around. PSO solution, represented by the swarm global best particle position, is then modified with a random chaotic flight as in Eq. (13),

$$P_{\text{GlobalBest}} = xP_{\text{GlobalBest}} + \text{rand}() * (2 * Y_n - 1) \tag{13}$$

where $P_{\text{GlobalBest}}$ is the best solution found by PSO, Y_n is the chaotic variable.

Several chaotic maps are possible such as logistics map, cubic map, Gauss Map, Sinc map, tent Map, Spense map or cusp map…. In this paper, we are using the logistics map in which the chaotic variable is iteratively modified following Eq. (14)

$$Y_{n+1} = \lambda * Y_n * (1 - Y_n); 0 < \lambda < 4 \tag{14}$$

where λ stands for chaotic control parameter.

2.7 Local search policy (2-opt)

The *K*-opt algorithm was proposed by Croes (1958), *K*-opt is a local heuristic search algorithm; it consists to remove for each node in the graph K connections, to reconnect them in other positions and then to evaluate the new proposition in terms of path length and to validate the shortest one as a solution. *K*-opt has a major advantage; it is an iterative paradigm inserted within heuristic while does not modify its structure (Helsgaun 2009). The most popular and known *K*-opt variants are 2-opt and 3-opt. Helsgaun (2009) gave an overview of *K*-opt and its hybridization with Lin–Kernighan algorithm to solve the traveling salesman problem. 2-opt algorithm (Dorigo and Stutzle 2004) consists of removing two connections from the current node and reconnect it to form another path without missing the tour construction; the new optioned tour is valid only if it is shorter than the initial one. 3-opt algorithm works like the 2-opt but in place to remove two edges and reconnect them. It consists of moving three edges and there are two ways to reconnect those edges (Matai et al. 2010). Figure 2 illustrates how the 2-opt local process is applied to a graph with four nodes.

2.8 Tour construction

Given a TSP of (N) cities, where Vi represents the city (i); using a discrete meta-heuristic algorithm, a solution can be iteratively built by combining cities configurations, so a solution may be represented by a set $TSPu = \{Vx, Vy, Vi, \ldots VN\}$. Then, a distance is computed according to Eqs. (1)–(5); this distance represents the tour length for the proposed configuration and also stands for the fitness function of the solution and is used to rank its quality. Local search mechanism, *Ls*, consists of changing a limited ($k < N$) part of this configuration in order to reduce the global distance. For $K = 2$, this consists in permuting *Vx*, and *Vy* positions in the solution configurations so that a new solution is generated $TSPu\text{-}Ls = \{Vy, Vx, Vi, \ldots, VN\}$. Processing consists of building several vertices, which are ranked in order to elect the best solution and to evolve toward a better one. The number of vertices is equal to the number of the meta-heuristic agents or instances.

3 Proposed bi-heuristic approaches

3.1 General architecture overview

Heuristic methods are known to be suitable in solving NP-hard problems with more or less balanced performance/quality ratio; essentially due to their capacity to return a possible solution in a relative acceptable time. Meanwhile, heuristic methods performances may suffer from the impact of its parameters on the quality of the results. The proposed bi-heuristic approaches are based on a couple of heuristics, as in Fig. 3, where the first manages the problem; it is called *problem-solving heuristic, PSH*.

The second heuristic acts to reduce dependencies of PSH to its parameters by an iterative parameter fitting process, it is called the supervising heuristic, S–H. For combinatorial and discrete optimization problems, discrete heuristics are more suitable to serve as problem-solving heuristics PHS. When the discrete heuristic got continuous parameters, self-adapting those parameters should be managed by a continuous supervising heuristic, S–H, such as PSO, FPA, FA or bat algorithm, BA.

The supervising heuristic, S–H, may have a direct feedback on solution quality by gathering fitness of the PSH to decide where to fix to keep changing the PSH parameters. A local search strategy is also added to avoid PSH to be trapped in local optimal solutions.

Such an architecture may be a generic solution for (Yang et al. 2013) investigations where authors called this process the hyperoptimization and pointed that meta-heuristics and bio-inspired algorithms performances depend on the tuning of their parameters; authors also agreed that a nice tuning leads the algorithm to better performances.

In this paper, the investigations concern a hybridization schema where ACO is the problem-solving heuristic, PSH, and where FPA, social PSO, cognitive PSO and chaotic

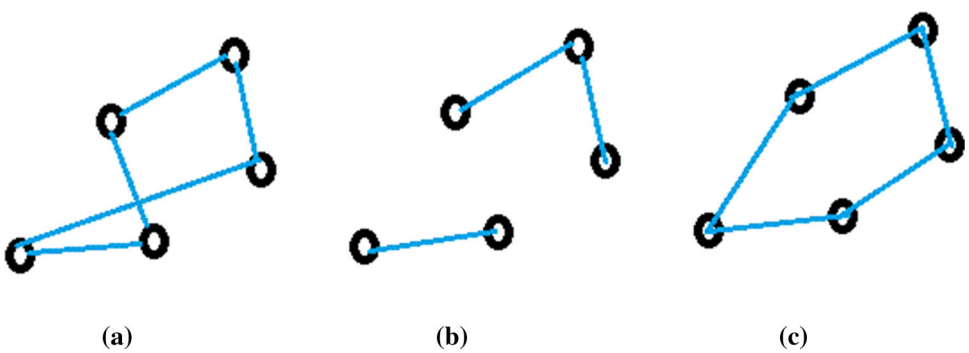


Fig. 2 Illustration of the 2-opt local search mechanism, **a** Possible tour, **b** Obtained configuration after removing two edges, **c** New tour obtained after new edges connections with a new configuration

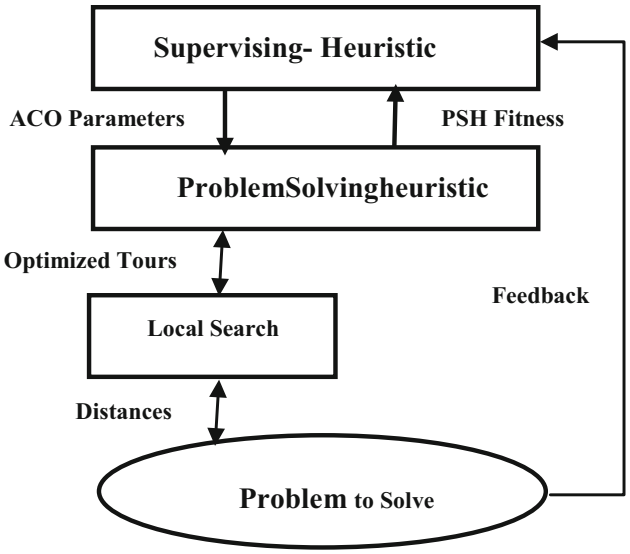


Fig. 3 Bi-heuristic local search architecture overview

PSO are investigated at meta-level as supervising heuristics.

3.2 Ant supervised by FPA with local search

This paragraph will focus on the ant supervised by flower pollination algorithm, AS-FPA; the schema is used for the four hybridizations of this paper. Since the problem to solve is the TSP, the ACO is selected as a PSH while, FPA, flower pollination algorithm is set as supervising heuristic S–H, and the 2-opt is used for local search, Ls, adjust parameters. During the initialization, we have to declare the meta-heuristic parameters, as in Fig. 4 flowchart. FPA helps ACO to find his optimized parameters, and the local search algorithm helps ACO to optimize the best tour obtained. The FPA pollen positions are assumed to ACO parameters (α, β and ρ). Pollen fitnesses are confused with TSP tour length returned by ACO. The local search, here 2-opt, is used to avoid ACO local optimums. The processing stop condition is observed when a tour length $<$ BKS (best known solution) is obtained or when the maximum iteration is achieved. The PFA processing steps are detailed in Fig. 4a:

4 Experimental investigations

4.1 Experimental protocol

Simulations are used to evaluate algorithms' performance in terms of algorithms convergence speed, solution quality and impact of ACO parameters on obtained solutions. Experimental results are obtained using Matlab software, R2015a. Matlab software runs on desktop machine with ACER Intel Core i7, 8 GB RAM size. For our experimental contribution, all test benches are used from TSPLib (Reinelt 1991). Selected test benches for the statistical analysis are eil51, berlin52, st70, st76, rat99, kroA100, eil101, ch150, kroA200. We place all tests and algorithms codes in the same machine folder. The fitness function is minimizing the total distance between N cities as Equation X (7). For each test bench, we have to determinate the results error (err.), average solution (avg.) and the standard deviation (SD). To compare to proposal to similar techniques, the results error is computed. The best global solution is a solution with the minimum error [min(err.)], see Eq. (15) where avg is the average solution, BKS stands for the best known solution.

$$\mathrm{Err} = ((\mathrm{Avg} - \mathrm{BKS}) \div \mathrm{BKS}) * 100 \tag{15}$$

In addition, we plot the evolution of fitness function in terms of iterations to see the convergence. To evaluate the impact of algorithm parameters on the results, the average, standard deviation, variance and error are used. The mean, the standard deviation and the variance are computed using Matlab tools (Mahi et al. 2015).

$$\mathrm{SD} = \sqrt{\left(\frac{1}{T}\sum_{t=1}^{T}(x_t - \mu)^2\right)} \tag{16}$$

The standard deviation SD is calculated using the Matlab predefined function "std," where T is iterations number, x_t is the best position in each iteration t and μ is the mean, see Eq. (16). The impact of ACO parameters (α, β, ρ) is estimated using the means of all implemented test benches. Algorithms evaluation process is based on two aspects, the fitness function which is the minimum of total distance achieved in passed tour and the time needed to converge. The best solution found was visualized using a Matlab Cartesian frame. The firefly algorithm was executed 50 times with 10 fireflies when the ACO runs for ant swarm size equal to city number. The flower pollination algorithm population size is 10, and the switching probability is 0.8. FPA is running 100 times; the same configuration was applied to PSO variants used in this paper.

4.2 Results and discussions

Discussion concerns essentially four hybridization scenarios: ant supervised by FPA-LS, ant supervised by social-PSO-Ls, ant supervised by cognitive-PSO-Ls and ant supervised by chaotic-PSO-Ls.

Step1 : 1.1 Initialization of the Supervising heuristic (FPA)
Population size, maximum iterations number, fitness function, xi = (α, β, ρ)
1.2 Initialization of Problem Solving Heuristic (ACO)
Population size, maximum iterations number, fitness function, (α, β, ρ)
1.3 Initialization of the Problem Instance & best known solution BKS.
Step2: FPA processing
2.1 Lunch SH Processing (FPA processing)
2.2 Evaluate pollen fitnesses
2.3Update pollen using boitic pollination
$x_i^{t+1} = x_i^t + \varepsilon(x_j^t - x_k^t)$
2.4 Update Pollen using abiotic pollination
$x_i^{t+1} = x_i^t + L(x_i^t - x_{gbest})$
Step3: Set ACO (parameters) = Pollen_Position(s)
Step4: Lunch ACO() instances
Step5: For each ACO instance
5.1 Lunch Local search()
5.2 Evaluate solution: Compute Tour length()
Step 6: if (not Stop condition)
6.1 Update Pollen_Fitness()
6.2 Loop to (2.4)
Step 7: if (Stop condition)
7.1 Return (TSP_Tour)
7.2 Return (Pollen_position)

(a)

Fig. 4 FPA pseudocode and ASFPA-Ls flowchart

4.2.1 Comparative results in terms of tour length

For the proposed hybridizations algorithms AS-FPA-Ls, AS-SoPSO-Ls, AS-CoPSO-Ls and AS-CPSO-Ls, a converging behavior is observed for the set of selected TSP test benches. For this experimentation, FPA maximum iteration was set to 50, and ACO maximum iteration is equal to 50 making to total iterative processing count of 2500 iterations, meanwhile, the algorithm gives the best solution in less than 250 iterations.

Concerning the ant supervised by chaotic-PSO with local search, AS-CPSO-Ls, Figure from 5.1 to 5.8 shows respective tour representations; in Fig. 5.1, the eil51 solution is presented showing that the proposal is able to converge to fair and acceptable solutions, the tour length obtained is about 452. Figure 5.2 illustrates berlin52 best tour which is 7642 which represents the BKS of the problem, while Fig. 5.3 is st70 with a route length of 725. Eil76 shortest tour is presented using Fig. 5.4, it is equal to 556 while the average is about 581. The shortest tour obtained by rat99 using AS-CPSO-Ls is 1282, the tour appears in Fig. 5.5. The best tour obtained for kroA100 is 21,821, see Fig. 5.6. Figure 5.7 is presenting eil101 best tour which is equal to 684. Kora 200 shortest tour is 31,774, Fig. 5.8.

Tables 5 and 6 show the average solutions, as well as the errors obtained with the proposed hybridizations. For AS-FPA-Ls, the best average results were observed for berlin52 TSP instance which error was about 0.63%. Acceptable results were observed for all remaining test benches with errors ranging from 0.63 to 6.6%. Best obtained solutions for each TSP test instance are illustrated in Fig. 6. Figure 6a shows the eil51 tour path, Fig. 6b illustrates the berlin52 tour, Fig. 6c shows the st70 TSP solution, while Fig. 6d demonstrates the eil 76 TSP instance. TSP instances rat99, eil101, Kroa100 and Ch 150 appear, respectively, in Fig. 6e–h.

For the ant supervised by social PSO-Ls, So-ASPSO-Ls, the social behavior of PSO was obtained by setting the cognitive factor c1 = 0.345, while the social parameter was about c2 = 3.45, making the social moderation factor 10 times more important than the cognitive factor. Obtained results shown in Tables 5 and 6 demonstrate that the proposed hybridization achieved the BKS for the following TSP instances: eil51, berlin52, st70, rat99, KorA100. With errors ranging from 0.014% for the Berlin52–3.05% obtained with eil101.

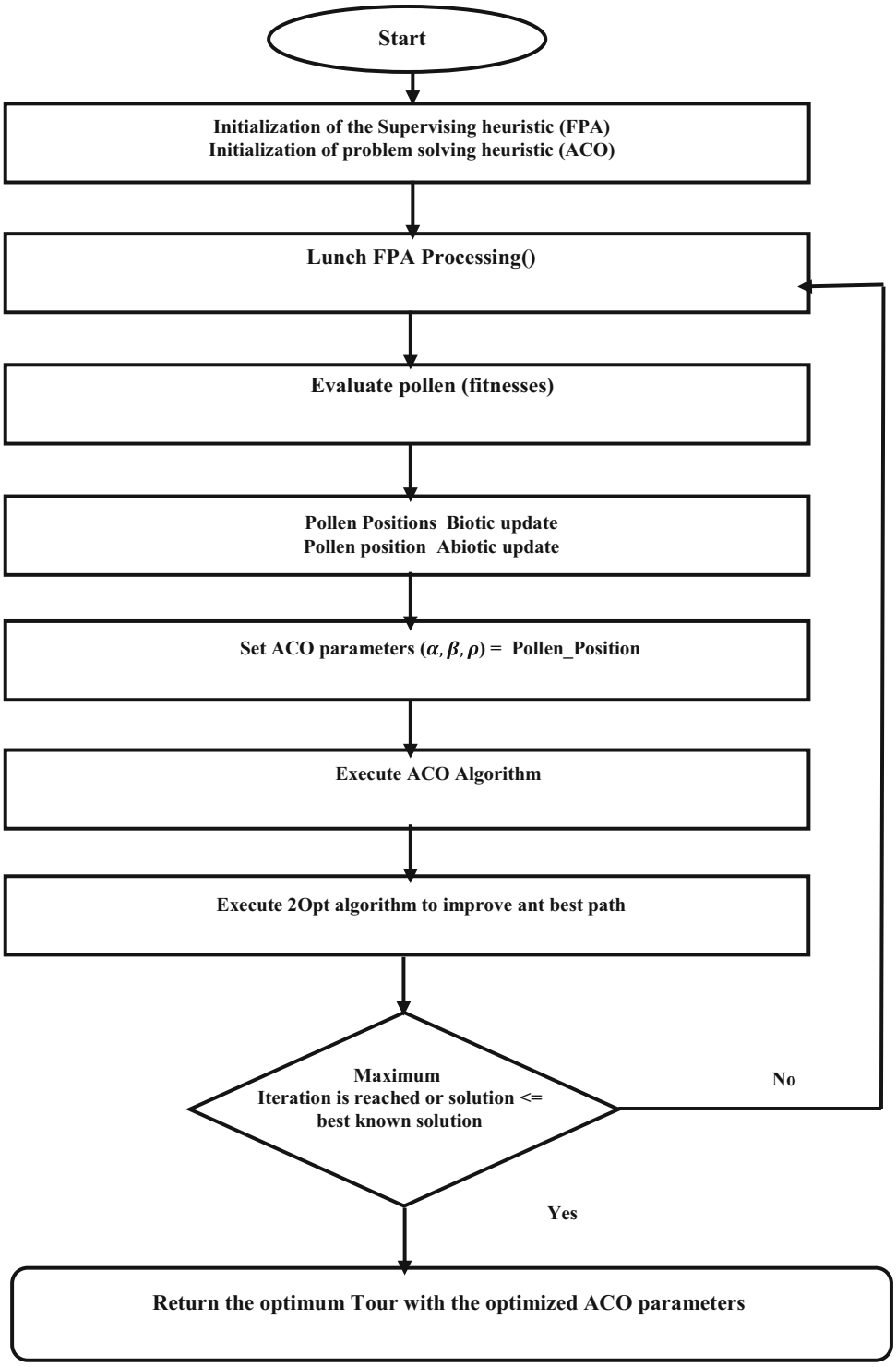
The cognitive ant supervised by PSO, Co-ASPSO-Ls, is the opposite configuration compared to the So-ASPSO-Ls: c1 = 3.45 and c2 = 0.345. For these PSO variants, the inertia weight was set to w = 0.8. Co-AS-PSO-Ls achieved fair solutions. The proposed hybridization achieved the BKS for the following TSP instances: eil51, berlin52, st70, rat99, KorA100, Ch150 and eil101, errors for these TSP instances ranged from 0.29% for berlin52–0.87 for ch150.The worst error obtained with this hybridization was 2.06% for rat99.

4.2.2 ACO self-adapted parameters

The simulated hybridization schema including AS-FPA-Ls, So-ASPSO-LS, Co-ASPSO-LS and AS-chaotic-PSO-Ls showed a capacity to retrieve stable ACO parameters allowing the proposed methods to achieve fair solutions of the TSP instances of this study.

ACO optimized parameters (α, β, ρ) allowing to achieve best results for AS-FPA-Ls appear in Tables 1, 2, 3, 4. Table 1 reports the best self-tuned parameters for the studied TSP instances with AS-FPA-LS; ACO parameters were, respectively, in the following intervals:

Fig. 4 continued



(b)

$\alpha = (0.5, 3.48), \beta(1.44, 5.89),$ and $\rho = (0.07, 0.7),$

ACO optimized parameters (α, β, ρ) allowing achieving best results for AS-chaotic-PSO-Ls appear in Table 2, ACO parameters were, respectively, in the following intervals:

$\alpha = (0.68, 2.64), \beta(1.86, 7.69),$ and $\rho = (0.18, 0.67),$

ACO optimized parameters (α, β, ρ) allowing achieving best results for So- ASPSO-Ls appear in Table 3, ACO parameters were, respectively, in the following intervals:

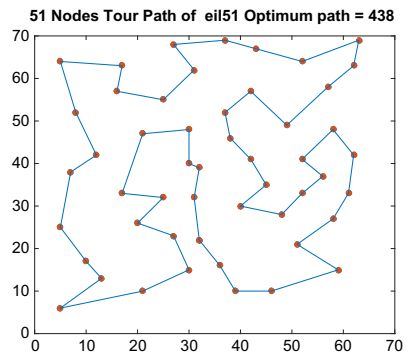


Figure 5.1 eil51Shortest Tour obtained by ACO-CPSO-2Opt

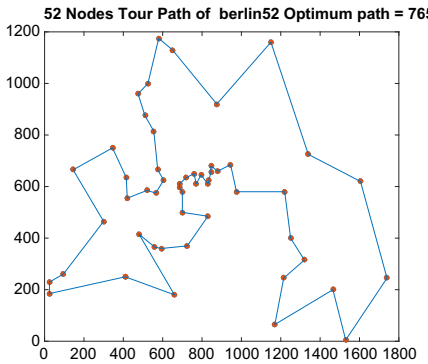


Figure 5.2 berlin52 Shortest Tour obtained by ACO-CPSO-2Opt

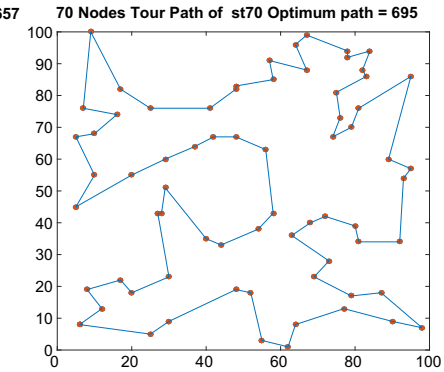


Figure 5.3st70 Shortest Tour obtained by ACO-CPSO-2Opt

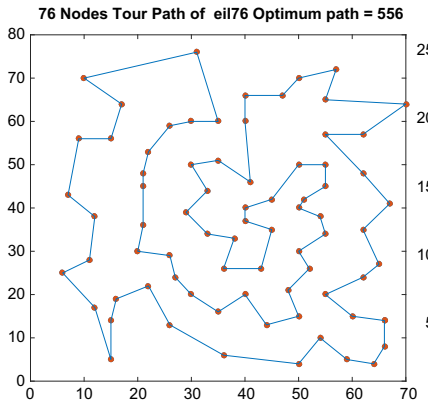


Figure 5.4 eil76 Shortest Tour obtained by ACO-CPSO-2Opt

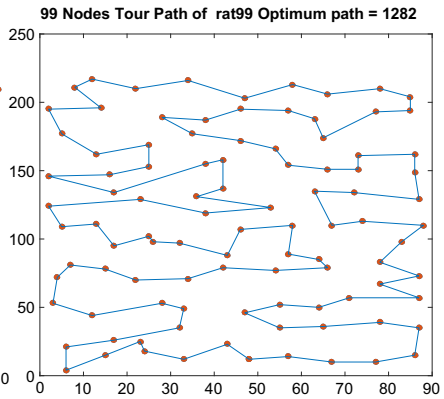


Figure 5.5rat99 Shortest Tour obtained by ACO-CPSO-2Opt

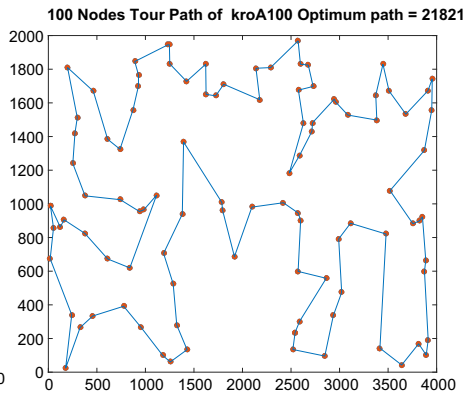


Figure 5.6kroA100 Shortest Tour obtained by ACO-CPSO-2Opt

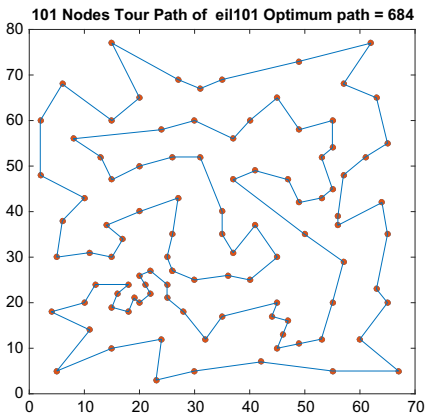


Figure 5.7eil101 Shortest Tour obtained by ACO-CPSO-2Opt

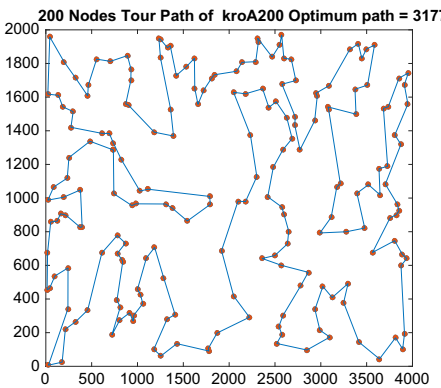


Figure 5.8kroA200 Shortest Tour obtained by ACO-CPSO-2Opt

Fig. 5 Shortest tours obtained by ACO-CPSO-2-opt

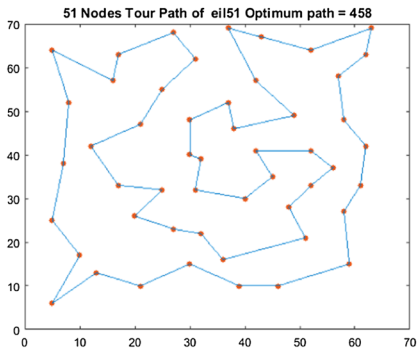
$\alpha = (0.5, 1.98), \beta(3.3, 4.98), and \rho = (0.04, 0.44),$

ACO optimized parameters (α, β, ρ) allowing achieving best results for Co-ASPSO-Ls appear in Table 4, ACO parameters were, respectively, in the following intervals:

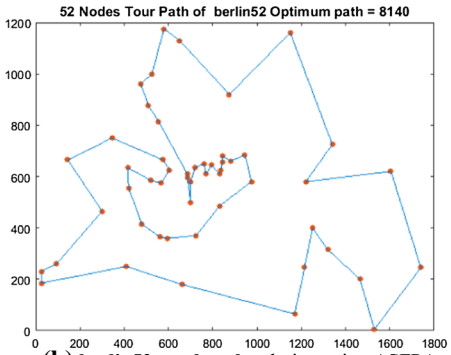
$\alpha = (0.5, 2.0), \beta(2.68, 5.1),$ and$\rho = (0.08, 0.51),$

Globally the returned parameters of the PSO variants for a given TSP instance are close, for example if we consider the eli76 TSP instance, AS-chaotic PSO-Ls parameters were (α, β, ρ) = (1.59, 4.22, 0.37), while for So-ASPSO-Ls, parameters used to be (α, β, ρ) = (1.51, 4.25, 0.41). The same parameters obtained with Co-ASPSO-Ls are 1.48, 4.67, 0.53.

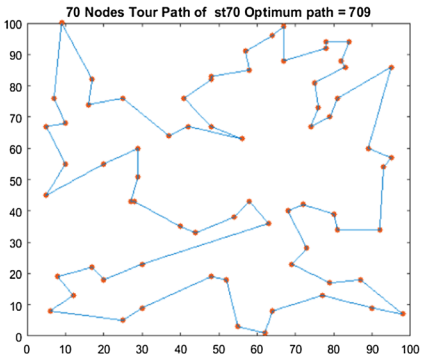
Fig. 6 ASFPA-Ls best test benches solutions



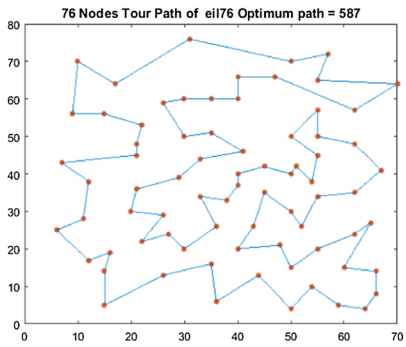
(a) Eil51best solution using ASFPA-Local search



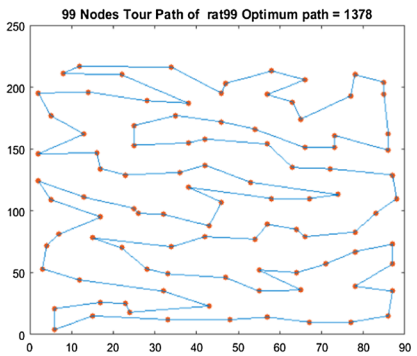
(b) berlin52 test bench solution using ASFPA-Local search



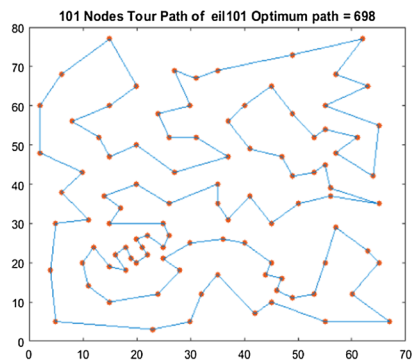
(c) st70 test bench best solution using ASFPA-Local search



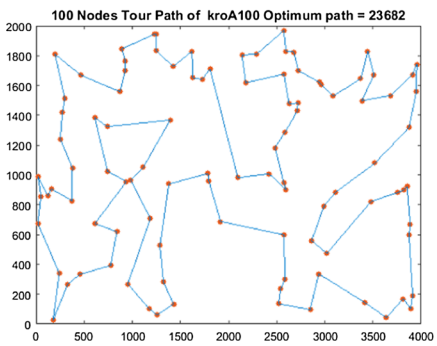
(d) eil76 test bench best solution using ASFPA-Local search



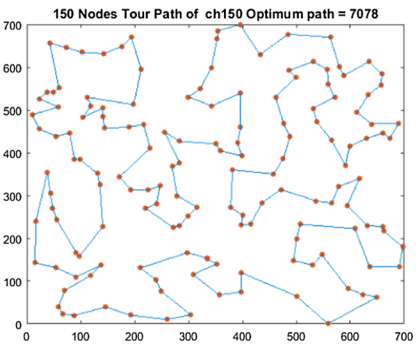
(e) rat99 test bench best solution using ASFPA-Local search



(f) eil101 test bench best solution using ASFPA-Local search



(g) kroA100 test bench best solution using ASFPA Local search



(h) ch150 test bench best solution using ASFPA-Local search

Table 1 Optimized ACO parameters for each test bench (AS-FPA-Ls)

Problems	α	β	ρ
eil51	1.560	1.447	0 .141
berlin52	3.483	5.898	0.701
st70	1.722	4.623	0.072
eil76	1.681	1.905	0.138
rat99	1.806	2.262	0.129
eil101	0.504	4.753	0.073
kroA100	1.264	2.541	0.068
ch150	1.727	4.091	0.348
kroA200	1.560	1.447	0.141

Table 2 Optimized ACO parameters for each test bench (AS-chaotic-PSO-Ls)

Problems	α	β	ρ
eil51	0.935	3.993	0.284
berlin52	1.462	5.000	0.510
st70	2.649	7.699	0.672
eil76	1.599	4.228	0.374
rat99	0.680	4.362	0.185
eil101	1.433	4.146	0.294
kroA100	1.549	4.944	0.034
ch150	0.868	1.862	0.349
kroA200	0.935	3.993	0.284

Table 3 Optimized ACO parameters for each test bench (So-AS-PSO-Ls)

Problems	α	β	ρ
eil51	0.977	4.983	0.216
berlin52	0.532	3.213	0.274
st70	1.983	3.479	0.026
eeil76	1.512	4.256	0.414
rat99	1.806	4.311	0.442
eil101	1.796	4.753	0.224
kroA100	1.654	3.307	0.326
ch150	0.949	3.829	0.218
kroA200	0.500	4.679	0.041

For AS-FPA-Ls hybridization and when observed for the same TSP instance, Eil76 best ACO parameters are 1.62, 1.90, 0.13, where only β parameter is out of range when compared to other three hybridization schemas.

Table 4 Optimized ACO parameters for each test bench (Co-AS-PSO-Ls)

Problems	α	β	ρ
eil51	0.921	5.100	0.200
berlin52	0.552	3.424	0.193
st70	2.003	3.671	0.038
eil76	1.489	4.678	0.534
rat99	1.856	5.004	0.514
eil101	1.897	4.981	0.312
kroA100	1.731	4.215	0.343
ch150	0.845	2.687	0.298
kroA200	0.812	4.012	0.088

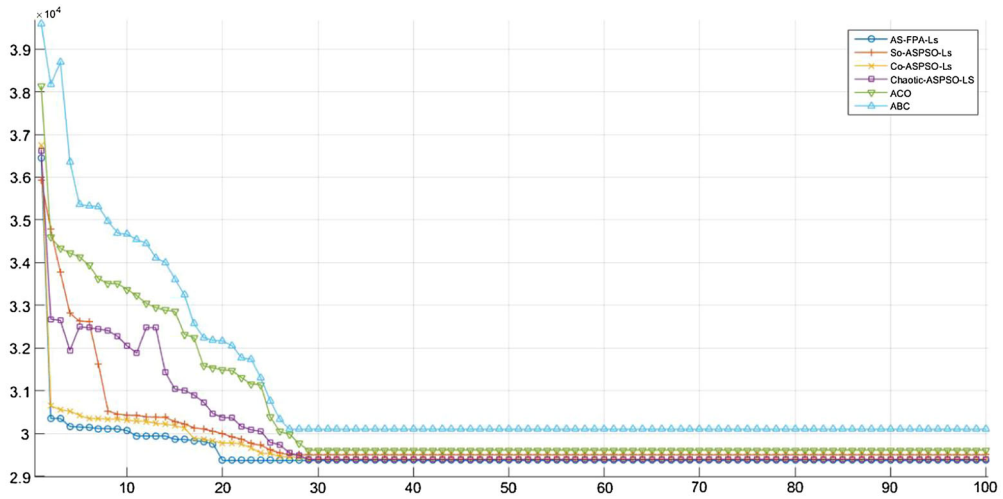
The evolution of fitness function is illustrated in Fig. 7 for FPA, over KroA200, CH150 and Berlin52 TSP instances where the hybrid approach proposed in this paper show an evident early convergence behavior to values close to BKS of the concerned test benches when compared to standard ACO solver for TSP, this is more visible when results are compared to artificial bee colony algorithm, ABC. Figure 8 illustrates how the proposed hybridizations are performing ACO parameters', α, β and ρ, adjustments with a converging behavior appearing from iteration 30 and following the same dynamic of the solutions' evolutions that appear in Fig. 7.

4.2.3 Execution times comparisons

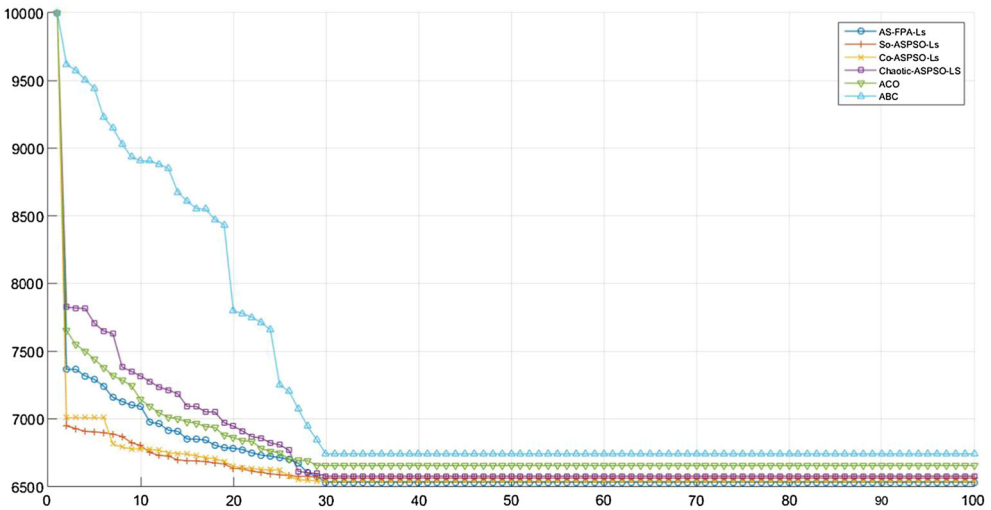
Time comparisons are based on execution time, not run time, an executable file is generated using Matlab resources, and then the executable file is executed on a Microsoft 10 operating system, an i7 processor with 8GB of RAM memory.

To compare time efficiency of the proposed methods, all meta-heuristics were compared to the (Abdulqader 2016) who hybridized the ACO algorithm with simulating annealing heuristic. The stop condition was fixed to 100 iterations or to results less or equal to BKS, best known solution. The time was measured using the tic-toc method of Matlab. In terms of time execution, Table 5 shows that PSO variants, Co-ASPSO-Ls and So-ASPSO-Ls and also AS-chaotic-PSO-Ls, are faster than ASFPA-Ls; the results are close to (Abdulqader 2016) with a relatively better time observed for Berlin52 for proposed methods. A relative faster processing is observed also for KroA200 when using So-ASPSO-LS.

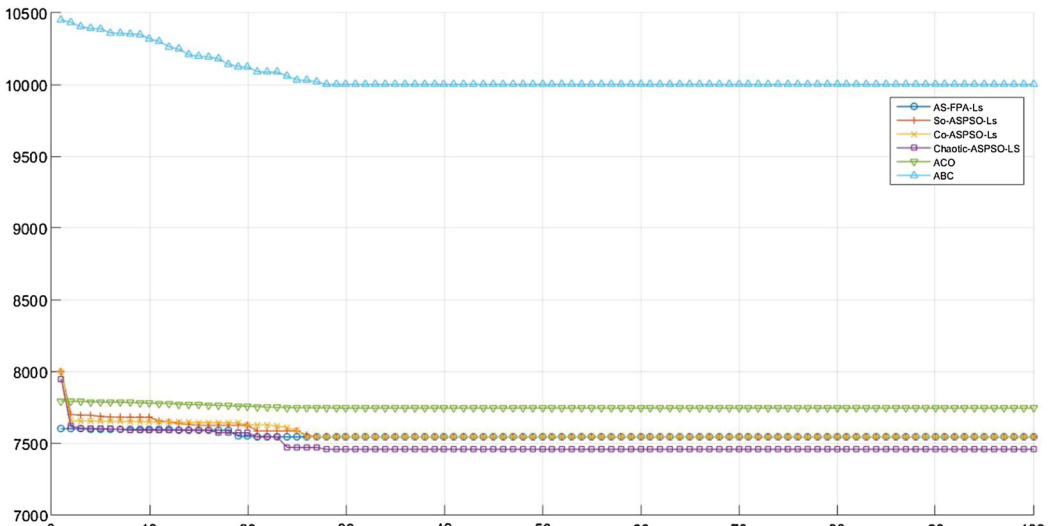
Computing times appearing in Table 6 are made for the same configuration of the meta-heuristics, the number of particles was set to 10 then to 20, and the maximum



(a)



(b)



(c)

◀**Fig. 7** Fitness function convergence for KroA200 (**a**), Ch150 (**b**) and Berlin52(**c**)

iteration number to 100. Note also that the times of Table 5 are the average time obtained for 10 tests. The most effective hybridization in terms of time is the SAS-PSO-Ls, followed by AS-chaotic-PSO-Ls which returned results in about 150 ms, for a swarm size of 10 particles. Time processing almost doubled when the number of particles is set to 20.

4.2.4 Nonparametric statistical analysis

4.2.4.1 Nonparametric statistical toward classical ACO and ABC The Wilcoxon test was implemented in order to evaluate the relative performances of the proposed hybridization with nominal ACO and ABC methods when applied to the same TSP instances. Proposed algorithms were also compared with each other, and the results are presented in Table 7. This test is based on an experimental statistical analysis over eight TSP test benches, with a series of 30 epochs of average tour costs for each method and for each set.

The Wilcoxon ranks, (Derrac et al. 2011), allowed to measure results variants of the proposed methods in comparison with ACO and artificial bee colony algorithm, ABC. When compared to ACO and ABC, the four proposed methods returned a p value less than 0.05, which means that the median of tested distributions is different from the median of the ACO and ABC distributions for all tested sets. The results of the table showed that FPA-ACO-

Table 6 Impact of the swarm size on execution time in (m) seconds for Berlin 52

	Ant Iteration Num = 50; Iteration heuristic1 = 100	
	Swarm size = 10	Swarm size = 20
AS-chaoticPSO-Ls	151.488	312.563
ASFPA-Ls	163.993	327.060
So-ASPSO-Ls	155.785	312.425
Co-ASPSO-Ls	149.785	306.554

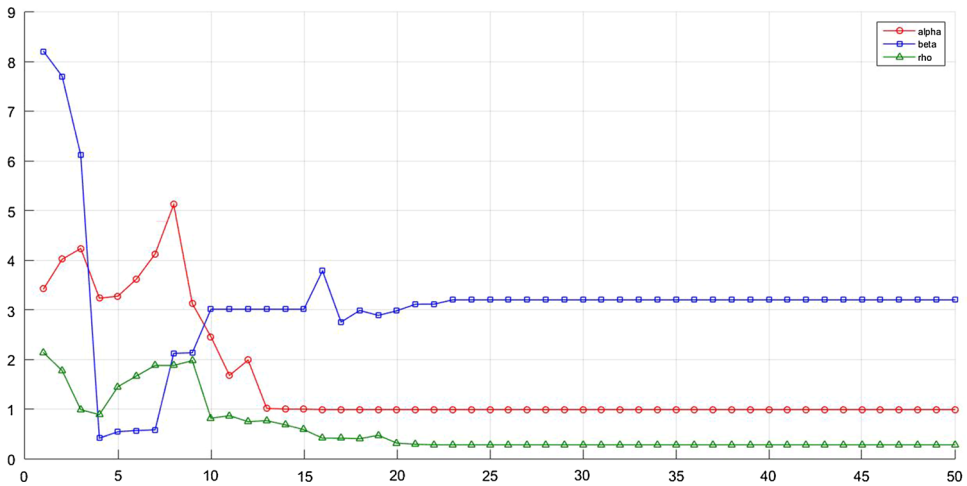


Fig. 8 ACO parameters evolution for (berlin52) with ASFPA-Ls

Table 5 Comparative execution time

TSP instance	ASFPA-Ls	So-ASPSO-Ls	Co-ASPSO-Ls	Chaotic	(Abdulqader 2016)
Eil51	0.08	0.04	0.05	0.07	0.00
Berlin52	0.16	0.15	0.14	0.15	0.06
St70	–	–	–	–	–
Eil76	0.01	0.01	0.01	0.02	0.01
Rat99	0.03	0.02	0.02	0.03	–
Eil101	0.04	0.02	0.02	0.03	0.02
KroA100	0.03	0.01	0.01	0.02	0.01
Ch150	0.02	0.02	0.03	0.06	0.03
KroA200	0.22	0.18	0.21	0.31	0.20

Table 7 Nonparametric statistical analysis based on Wilcoxon signed rank test

Test bench	Compared methods		Solution evaluation			
KroA100	Method 1	Method 2	R^-	R^+	ρ	Best method
	Chaotic-ASPSO-Ls	ACO	381	84	4.98e-04	Ch-PSO-ACO-LS
	Chaotic-ASPSO-Ls	ABC	465	0	3.01e-11	Ch-PSO-ACO-LS
	So-ASPSO-Ls	ACO	465	0	1.77e-10	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	465	0	3.01e-11	So-ASPSO-Ls
	FPA-ACO-Ls	ACO	444	21	4.44e-07	FPA-ACO-Ls
	FPA-ACO-LS	ABC	465	0	3.01e-11	FPA-ACO-Ls
	Co-ASPSO-Ls	ACO	465	0	3.01e-11	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	465	0	3.02e-11	Co-ASPSO-Ls
KorA200	FPA-ACO-Ls	ACO	446	19	4.44e-07	FPA-ACO-Ls
	FPA-ACO-Ls	ABC	449	16	1.45e-06	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	337	128	0.0281	Ch-PSO-ACO-LS
	Chaotic-ASPSO-Ls	ABC	342	123	0.0437	Ch-PSO-ACO-LS
	Co-ASPSO-Ls	ACO	443	22	1.60e-06	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	434	31	5.86e-06	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	430	35	1.28e-06	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	432	33	8.29e-06	So-ASPSO-Ls
Ch150	FPA-ACO-Ls	ACO	265	200	0.04290	FPA-ACO-Ls
Eil101	FPA-ACO-Ls	ABC	462	3	1.69e-09	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	225	240	0.0494	ACO
	Chaotic-ASPSO-Ls	ABC	451	14	6.53e-08	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	460	5	1.01e-08	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	465	0	3.02e-11	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	439	26	3.01e-07	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	465	0	4.50e-11	So-ASPSO-Ls
	FPA-ACO-Ls	ACO	465	0	5,49e-11	FPA-ACO-Ls
	FPA-ACO-Ls	ABC	465	0	3,02e-11	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	410	55	3.36e-04	Chaotic-ASPSO-Ls
	Chaotic-ASPSO-Ls	ABC	465	0	3,01e-11	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	463	2	4,08e-8	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	465	0	3,02e-11	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	465	0	3,01e-11	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	465	0	3,01e-11	So-ASPSO-Ls
Eil76	FPA-ACO-Ls	ACO	136	329	0.0315	ACO
	FPA-ACO-Ls	ABC	0	465	3.01e-11	ABC
	Chaotic-ASPSO-Ls	ACO	46	419	3.36e-04	ACO
	Chaotic-ASPSO-Ls	ABC	465	0	3.01e-11	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	465	0	3,01e-11	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	465	0	3,01e-11	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	465	0	3,01e-11	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	465	0	3,01e-11	So-ASPSO-Ls
Eil70	FPA-ACO-Ls	ACO	210	0	1.06e-07	FPA-ACO-Ls
	FPA-ACO-Ls	ABC	210	0	6.79e-08	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	30	180	0.0084	Chaotic-ASPSO-Ls
	Chaotic-ASPSO-Ls	ABC	210	0	6.79e-08	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	210	0	9.17e-08	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	210	0	6.79e-08	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	210	0	6.79e-08	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	210	0	6.79e-08	So-ASPSO-Ls

Table 7 continued

Test bench	Compared methods		Solution evaluation			
Berlin52	FPA-ACO-Ls	ACO	207	3	1.04e-06	FPA-ACO-Ls
	FPA-ACO-Ls	ABC	210	0	6.79e-08	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	210	0	1.04e-06	Chaotic-ASPSO-Ls
	Chaotic-ASPSO-Ls	ABC	210	0	6.78e-08	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	168	42	0.0256	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	210	0	6.79e-08	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	182	28	0.0010	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	210	0	6.79e-08	So-ASPSO-Ls
Eil51	FPA-ACO-Ls	ACO	465	0	3.54e-11	FPA-ACO-Ls
	FPA-ACO-Ls	ABC	455	10	4.61e-6	FPA-ACO-Ls
	Chaotic-ASPSO-Ls	ACO	465	0	3.01e-11	Chaotic-ASPSO-Ls
	Chaotic-ASPSO-Ls	ABC	465	0	3.01e-11	Chaotic-ASPSO-Ls
	Co-ASPSO-Ls	ACO	465	0	3.01e-11	Co-ASPSO-Ls
	Co-ASPSO-Ls	ABC	465	0	3.01e-11	Co-ASPSO-Ls
	So-ASPSO-Ls	ACO	454	11	7.20e-8	So-ASPSO-Ls
	So-ASPSO-Ls	ABC	445	20	2.58 e-6	So-ASPSO-Ls

Ls, So-ASPSO-Ls, Co-ASPSO-Ls and chaotic-ASPSO-Ls presented in this paper are better than nominal ABC for ST70, Berlin52, Eil51, Eil101 KroA100, Ch150 and KroA200 TSP instances, while ABC is better only for Eil76. Results observed in comparison with ACO are more moderate since it was observed that the chaotic-ASPSO-Ls is worst than nominal ACO for Ch150 TSP instance. The FPA-ACO-Ls and chaotic-ASPSO-Ls are worst than ACO for Eil76.

4.2.4.2 Nonparametric statistical analysis of the proposed methods to each others This paragraph focuses on some analysis based on the Wilcoxon rank sum test when applied to compare the four proposed methods to each other's based on the following TSP data sets (Tables 8, 9, 10, 11): KroA100, KroA200, Eil101 and Ch150 which are all TSP instances with more than 100 cities, for all tests, the p value was less than 0.05, analysis is made on the basis of R^- and R^+parameters. For KroA100, the Wilcoxon test of AS-FPA-Ls to other three propositions is resumed in Table 8, allowing see that FPA is worse than the So-ASPSO-Ls and the cognitive-PSO-ACO-Ls while better than the chaotic-ASPSO-Ls for this TSP instance. For KroA200, antagonist results were observed and give a relative advantage for the AS-FPA-Ls in regards other proposed methods in the paper

Table 8 Wilcoxon analysis of FPA to other proposed methods for KroA100

Compared methods		Solution evaluation		
Algorithm 1	Algorithm 2	R^-	R^+	ρ
FPA-ACO-Ls	So-ASPSO-Ls	118	347	0.0138
FPA-ACO-Ls	Co-ASPSO-Ls	15	450	7.08e-08
FPA-ACO-Ls	Chaotic-ASPSO-Ls	270	195	0.03042

Table 9 Wilcoxon analysis of FPA to other proposed methods for KroA200

Compared methods		Solution evaluation		
Algorithm 1	Algorithm 2	R^-	R^+	ρ
FPA-ACO-Ls	So-ASPSO-Ls	306	159	0.0018
FPA-ACO-Ls	Co-ASPSO-Ls	288	177	0.0022
FPA-ACO-Ls	Chaotic-ASPSO-Ls	450	15	1.3594e-07

Table 10 Wilcoxon analysis of FPA to other proposed methods for Ch150

Compared methods		Solution evaluation		
Algorithm 1	Algorithm 2	R^-	R^+	ρ
FPA-ACO-Ls	So-ASPSO-Ls	43	422	3.15e-05
FPA-ACO-Ls	Co-ASPSO-Ls	24	441	8.83e-07
FPA-ACO-Ls	Chaotic-ASPSO-Ls	266	199	0.0206

Table 11 Wilcoxon analysis of FPA to other proposed methods for Eil101

Compared methods		Solution evaluation		
Algorithm 1	Algorithm 2	R^-	R^+	ρ
FPA-ACO-Ls	So-ASPSO-Ls	115	350	0,0251
FPA-ACO-Ls	Co-ASPSO-Ls	43	422	7,1988e-05
FPA-ACO-Ls	Chaotic-ASPSO-Ls	369	96	0,0080

Table 12 Results and comparisons of the proposed algorithms with state-of-the-art techniques (BKS: best known solution)

Authors	Problem BKS	eil51 426	berlin52 7542	st70 675	eil76 538	rat99 1211	eil101 629	kroA100 21,282	ch150 6528	kroA200 29,368
ACO–2-opt (Jun-man and Yi 2012)	Avg	439.25	7556.58	–	–	–	672.37	23,441.80	–	–
	SD	–	–	–	–	–	–	–	–	–
	Error (%)	3.11	0.19	–	–	–	6.90	10.15	–	–
Hybrid ACO (Junqiang and Aijia 2012)	Avg	431.20	7560.54	–	–	1241.33	–	–	–	–
	SD	2.00	67.48	–	–	9.60	–	–	–	–
	Error (%)	1.22	0.23	–	–	2.5	–	–	–	–
GA-ant system (Dong et al. 2012)	Avg	–	7634.00	–	542.00	–	–	21,437.00	–	29,946.00
	SD	–	–	–	–	–	–	–	–	–
	Error (%)	–	1.22	–	0.74	–	–	0.73	–	1.97
ACO-Tagushi method (Peker et al. 2013)	Avg	435.40	7635.40	–	565.50	–	655.00	21,567.10	–	–
	SD	–	–	–	–	–	–	–	–	–
	Error (%)	2.21	1.24	–	5.11	–	4.13	1.34	–	–
ACO-ABC (Gündüz et al. 2015)	Avg	443.39	7544.37	700.58	557.98	–	683.39	22,435.31	6677.12	–
	SD	5.25	0.00	7.51	4.10	–	6.56	231.34	19.30	–
	Error (%)	4.08	0.03	3.79	3.71	–	8.65	5.42	2.28	–
PSO–ACO–3-opt (α, β) (Mahia et al. 2015)	Avg	426.45	7543.20	678.20	538.30	1227.40	632.70	21,445.10	6563.95	29,646.05
	SD	0.61	2.37	1.47	0.47	1.98	2.12	78.24	27.58	114.71
	Error (%)	0.11	0.02	0.47	0.06	1.35	0.59	0.77	0.55	0.95
AS-PSO 2-opt (Kefi et al. 2016a, b)	Avg	428	7542	678	541.20	1236	632	21,457	6560	29,837
	SD	9.97	202.62	15.92	12.16	31.74	12.29	391.85	171.90	359.28
	Error (%)	0.23	0.001	0.44	0.55	2.08	0.47	0.82	0.49	1.60
SAS-PSO-Ls (Rokbani et al. 2019a, b)	Avg	426.53	7542.42	675.54	543.17	–	645.53	21,305.78	6606.26	29,924.31
	SD	9.64	206.14	20.68	13.14	–	12.13	674,46	176.58	631.58
	Error (%)	0.05	0.05	0.05	0.92937	–	2.5437	0,10,807	1.1949	1.8932
ASFA-Ls (Rokbani et al. 2019a, b)	Avg	428.08	7542.33	675.11	541.03	1213.26	639.19	21,282,09	6571.08	29,532.02
	SD	8.35	–	15.19	11.14	29.14	12.17	437.83	159.15	628.90
	Error (%)	0.6291	0.0	0.0	0.55762	0.16515	1.5898	0	0.6587	0.55843
PACO-3-opt (Gülcü et al. 2018)	Avg	426.35	7542.00	677.85	539.85	1217.10	630.55	21,326.80	–	29,644.50
	SD	0.49	0	0.99	1.09	4.01	2.63	3.72	–	53.43
	Error (%)	0.08	0	0.42	0.3	0.50	0.25	0.21	–	0.94
ACO (Gülcü et al. 2015)	Avg	457.86	7659.31	709.16	561.98	–	693.42	22,880.12	6528.00	–
	SD	4.07	38.7	8.27	3.5	–	6.8	235.18	6.56	–
	Error (%)	–	–	–	–	–	–	–	–	–
ABC (Gülcü et al. 2015)	Avg	590.49	10,390.26	1230.46	931.44	–	1315.95	53,840.03	6702.87	–
	SD	15.79	38.7	41.79	24.86	–	35.28	2198.36	20.73	–
	Error (%)	–	–	–	–	–	–	–	–	–
(Abdulqader 2016)	Avg	426.00	7542.00	–	578.00	–	629.00	21,282.00	6528.00	29,368.00
	SD	0.00	0.00	–	0.00	–	0.00	0.00	0.00	0.00

Table 12 (continued)

Authors	Problem	eil51	berlin52	st70	eil76	rat99	eil101	kroA100	ch150	kroA200
	BKS	426	7542	675	538	1211	629	21,282	6528	29,368
	Error (%)	–		–	–	–	–	–	–	–
ASFPA-Ls	Avg	437.13	7542.38	694.18	566.31	1271.22	661.46	220,034.05	6959.04	29,837.18
	SD	11.18	117.46	21.52	1 3.33	33.34	11.97	594.69	189 .85	432.68
	Error (%)	2.61	0.63	2.84	5.26	4.95	5.16	3.53	6.60	1.59
So-ASPSO-Ls	Avg	426.22	7543.10	675.20	544.66	1212.66	648.22	21,306.41	6606.14	29,934.65
	SD	10.21	106.44	20.86	26.24	29.14	11.13	664,97	166.15	661.55
	Error (%)	0.05	0.014	0.02	1.11	0.13	3.05	0,11	1.19	1.92
Co-ASPSO-Ls	Avg	428.66	7544.10	678.40	543.82	1236.32	632.36	21,458.12	6558.52	29,847.34
	SD	9.897	240.62	15.92	12.16	31.74	12.29	391.85	278.80	269.88
	Error (%)	0.62	0.29	0.51	1.00	2.06	0.53	0.82	0.87	1.63
Chaotic-ASPSO-Ls	Avg	452.80	7542.08	725.51	581.31	1330.78	685.01	22,780.25	7135.10	32,270.58
	SD	10.02	234.01	20.32	12.82	28.99	12.00	593.00	345.53	662.3019
	Error (%)	6.29	0	7.48	8.05	9.89	8.90	7.01	–	9.88

as it can be visible in Table 9.The Wilcoxon test for the Ch150 TSP instance shows that the social-PSO-ACO-Ls and cognitive-PSO-Ls are better than FPA-ACO-Ls while FPA-ACO-Ls is better than the chaotic-ASPSO-Ls for this data set. Table 12 discusses about the comparative analysis of proposed work with existing works.

5 Conclusions and perspectives

This paper introduced a generalized frame of bi-heuristic self-adaptive schema for NP-hard combinatorial optimization problem such as TSP. The proposal includes ASFPA-Ls, So-AS-PSO-Ls and Co-ASPSO-Ls and also ant supervised by chaotic PSO, AS-chaotic-PSO-Ls. Proposed hybridizations attended fair solutions for the TSP problems instances used in the experimental investigations including eil 51, berlin52, St70, eil76, rat 99, eil101, KroA100, Ch150 and kroA200. A good balance performance/time is found with the social ant supervised by PSO with local search, So-ASPSO-Ls. The cognitive ant supervised by PSO with local search, Co-ASPSO-Ls comes in the second position in terms of time effectiveness with close performances followed by AS-chaotic-PSO-Ls and finally AS-FPA-Ls.

PSO-based hybridizations approaches returned low rate errors or BKS for used test benches: berlin52, st70, eil76, rat99, kroA100 and kroA200. ASFPA-Ls gives high quality solution for berlin52 and eil101. In terms of parameters, all propositions showed capacities of self-adaptive behaviors which strengthen the idea of a heuristic schema less depending on parameters variations. The real-time processing constraints of the hybridization schema should also be envisaged for a prospective online implementation is manufacturing and robotics solutions, essentially for path planning and routing systems when the time constraints are about (200 ms) for Berlin 52 related problem. Future investigations will include also the prospective of the proposal to recent heuristics and bio-inspired techniques such as hybrid crow search algorithm (Hassanien et al. 2018) and its chaotic version of Rizk-Allah et al. (2018).

Compliance with ethical standards

Conflict of interest The authors declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work; there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

References

AL-Wagih K (2015) An improved flower pollination algorithm for solving integer programming problems. Appl Math Inf Sci Lett 3(1):31–37

Ariyaratne A, Fernando TGI, Weerakoon S (2016) A self-tuning firefly algorithm to tune the parameters of ant colony system (ACSFA)

Bidar M, Kanan HR (2013) Modified firefly algorithm using fuzzy tuned parameters. 13th Iranian Conference on Fuzzy Systems (IFSC). DOI:https://doi.org/10.1109/IFSC.2013.6675634
Croes GA (1958) A method for solving traveling-salesman problems. Oper Res 6(6):791–812. https://doi.org/10.1287/opre.6.6.791
Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
Dhiman G (2019) ESA: a hybrid bio-inspired metaheuristic optimization approach for engineering problems. Eng Comput, pp 1–31
Dong G, Guo WW, Tickle K (2012) Solving the traveling salesman problem using cooperative genetic ant systems. Expert Syst Appl 39(5):5006–5011
Dorigo M, Birattari M (2007) Swarm intelligence. Scholarpedia 2:1462
Dorigo M, Gambardella LM (1997) Ant colonies for the travelling salesman problem. Biosystems 43(2):73–81
Dorigo M, Stutzle T (2004) Ant Colony Optimization, Massachusetts Institute of Technology
Elloumi W, Rokbani N, Alimi AM (2009) "Ant supervised by PSO". In: Proceedings of International symposium on Computational Intelligence and Intelligent Informatics, pp 161–166
Gülcü Ź, Mahi M, Baykan ÖK, KodazH. (2018) A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem. Soft Comput. 22(5):1669–1685
Gündüz M, Kiran MS, Özceylan E (2015) A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. Turk J Electr Eng Comput Sci 23(1):103–117. https://doi.org/10.3906/elk-1210-147
Hassanien AE, Rizk-Allah RM, Elhoseny M (2018) A hybrid crow search algorithm based on rough searching scheme for solving engineering optimization problems. J Ambient Intell Humaniz Comput, pp 1–25
Helsgaun, k. (2009). "An effective implementation of K-opt moves for the Lin-Kernighan TSP Heuristic", Math Progr Comput, pp 119–163
Jun-man K, Yi Z (2012) Application of an improved ant colony optimization on generalized traveling salesman problem. Energy Procedia 17:319–325
Junqiang W, Aijia O (2012) A hybrid algorithm of ACO and delete-cross method for TSP. In: Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering, pp. 1694–1696, IEEE
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, Technical Report-TR06
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, IEEE, pp 50–53
Kave A, Ghazaan MI (2019) A new VPS-based algorithm for multi-objective optimization problems. Eng Comput, pp 1–12.
Kefi S, Rokbani N, Krömer P, Alimi AM (2016) "Ant supervised by PSO and 2-opt algorithm, AS-PSO-2Opt, applied to traveling salesman problem". IEEE International conference on System Man and Cybernetics SMC
Kefi S, Rokbani N, Alimi AM (2016) Impact of ant size on ant supervised by PSO, AS-PSO, performances. In: International Conference on Hybrid Intelligent Systems, pp 567–577. Springer, Cham
Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp 1942–1948
Kora P, Rama Krishna KS (2016) Hybrid firefly and particle swarm optimization algorithm for the detection of bundle branch block. Int J Cardiovasc Acad 2(1):44–48. https://doi.org/10.1016/j.ijcac.2015.12.001
Kumbharana SN, Pandey PGM (2013) Solving travelling salesman problem using firefly algorithm. Int J Res Sci Adv Technol 2:53–57
Mahi M, Baykan ÖK, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem. Appl Soft Comput 30(Supplement C):484–490
Matai R, Singh S, Mittal ML (2010) Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches (D. Davendra Ed.): Traveling Salesman Problem, Theory and Applications
MATLAB Statistics Toolbox User's Guide (2014). The MathWorksInc. http:www.mathworks.com/help/pdf_doc/stats/stats.pdf
Mirjalili S, Mirjalili SM, Yang XS (2014) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681
Mohsen AM (2016) Annealing ant colony optimization with mutation operator for solving TSP. Comput Intell Neurosci. https://doi.org/10.1155/2016/8932896
Nabil E (2016) A modified flower pollination algorithm for global optimization. Expert Syst Appl 57:192–203
Nekouie N, Yaghoobi M (2015) MFASA: A new memetic firefly algorithm based on simulated annealing. Int J Mech Electr Comput Technol 5:2347–2354
Olief I, Farisi R, Setiyono B, Danandjojo RI (2016) A Hybrid firefly algorithm–ant colony optimization for traveling salesman problem open journal systems, p 7
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
Pedersen MEH, Chipperfield AJ (2010) Simplifying particle swarm optimization. Appl Soft Comput 10:618–628
Peker M, Şen B, Kumru PY (2013) An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method. Turk J Electr Eng Comput Sci 21(1):2015–2036
Reinelt G (1991) TSPLIB—A traveling salesman problem library. ORSA J Comput. https://doi.org/10.1287/ijoc.3.4.376
Rizk-Allah RM, Hassanien AE (2018) New binary bat algorithm for solving 0–1 knapsack problem. Complex Intell Syste 4(1):31–53
Rizk-Allah RM, Zaki EM, El-Sawy AA (2013) Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. Appl Math Comput 224:473–483
Rizk-Allah RM, El-Sehiemy RA, Deb S, Wang GG (2017) A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. J Supercomput 73(3):1235–1256
Rizk-Allah RM, Hassanien AE, Bhattacharyya S (2018) Chaotic crow search algorithm for fractional optimization problems. Appl Soft Comput 71:1161–1175
Rokbani N, Abraham A, Alimi AM (2013) Fuzzy ant supervised by PSO and simplified ant supervised PSO applied to TSP". In: The 13th International conference on hybrid intelligent systems (HIS), pp 251–255
Rokbani N, Momasso AL, Alimi AM (2013) ''AS-PSO ant supervised by PSO meta-heuristic with application to TSP. Proc Eng Technol 4:148–152
Rokbani N, Casals A, Alimi AM (2015) IK-FA, a New heuristic inverse kinematics solver using firefly algorithm. In: Azar AT, Vaidyanathan S (eds) Computational intelligence applications in modeling and control. Springer International Publishing, Cham, pp 369–395
Rokbani N, Abraham A, Twir I, Haqiq A (2019) Solving the travelling salesman problem using fuzzy and simplified variants of ant supervised by PSO with local search policy, FAS-PSO-LS, SAS-PSO-LS. Int J Hybrid Intell Syst 15(1):17–26

Rokbani N, Kromer P, Twir I, Alimi AM (2019) A new hybrid gravitational particle swarm optimisation-ACO with local search mechanism, PSOGSA-ACO-Ls for TSP. Int J Intell Eng Inf 7(4):384–398

Saji Y, Riffi ME (2016) A novel discrete bat algorithm for solving the travelling salesman problem. Neural Comput Appl 27(7):1853–1866

Saraei M, Analouei R, Mansouri P (2015) Solving of travelling salesman problem using firefly algorithm with greedy approach. In: Proceeding of the international conference on non-linear system and optimization in computer and electrical engineering

Taengtang T, Sitthivet W, Paithoonwattanakij K (2013)"Fermicidae swarm system". In: Proceedings of the 2013 international conference on information technology and electrical engineering (ICITEE), pp 124–126

Tsai C-F, Tsai C-W, Tseng C-C (2004) A new hybrid heuristic approach for solving large traveling salesman problem. Inf Sci 166(1):67–81

Twir I, Rokbani N, Alimi A (2018) Ant supervised by firefly algorithm with a local search mechanism, ASFA-2Opt. In: 2018 International conference on control, automation and diagnosis (ICCAD), IEEE, pp 1–5

Valenzuela L, Valdez F, Melin P (2017) Flower pollination algorithm with fuzzy approach for solving optimization problems. In: Melin P, Castillo O, Kacprzyk J (eds) Nature-inspired design of hybrid intelligent systems. Springer, Berlin, pp 357–369

Wang M-b, Fu Q, Tong N, Li M, Zhao Y (2016) An improved firefly algorithm for traveling salesman problems. In: Proceeding of the 4th national conference on electrical, electronics and computer engineering (NCEECE 2015)

Yang XS (2010) Firefly algorithm, Levy flights and global optimization. Research and development in intelligent systems XXVI. Springer, London, pp 209–218

Yang XS (2013) Flower pollination algorithm: A novel approach for multi- objective optimization. Eng Optim. https://doi.org/10.1080/0305215X.2013.832237

Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Engineering computations

Yang XS, Deb S, Loomes M, Karamanoglu M (2013) A framework for self-tuning optimization algorithm. Neural Comput Appl 23(7–8):2051–2057

Zhang M, Dai J, Zheng J, Zhang G (2016) 'An improved flower pollination algorithm, In; Proceedings of the 2016 13th Web information systems and applications conference (WISA) pp 179–183, IEEE

Affiliations

Nizar Rokbani[1,4] · Raghvendra Kumar[3] · Ajith Abraham[2] · Adel M. Alimi[4] · Hoang Viet Long[5,6] · Ishaani Priyadarshini[7] · Le Hoang Son[8,9]

1 MIR-Labs, Machine Intelligence Research Labs, Auburn, USA

2 Research Groups in Intelligent Machines (REGIM-Lab.), LR11ES48, National Engineering School of Sfax, 3038 Sfax, Tunisia

3 Department of Computer Science and Engineering, GIET University, Gunupur, India

4 University of Sousse ISSAT Sousse, Taffela City, 4003 Sousse, Tunisia

5 Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam

6 Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam

7 Department of Electrical Engineering, University of Delaware, Newark, USA

8 Institute of Research and Development, Duy Tan University, Da Nang, Vietnam

9 VNU Information Technology Institute, Vietnam National University, Hanoi, Vietnam

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com