Received March 12, 2020, accepted March 29, 2020, date of publication April 3, 2020, date of current version April 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2985498

A Novel Ant Colony Optimization Algorithm With Levy Flight

YAHUI LIU[1] AND BUYANG CAO[2]
[1]School of Software Engineering, Tongji University, Shanghai 200092, China
[2]College of Architecture and Urban Planning, Tongji University, Shanghai 200092, China

Corresponding author: Yahui Liu (liuyahui@tongji.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 41771410.

ABSTRACT Ant Colony Optimization (ACO) is a widely applied meta-heuristic algorithm. Little researches focused on the candidate selection mechanism, which was developed based on the simple uniform distribution. This paper employs the Levy flight mechanism based on Levy distribution to the candidate selection process and takes advantage of Levy flight that not only guarantees the search speed but also extends the searching space to improve the performance of ACO. Levy ACO incorporating with Levy flight developed on the top of Max-min ACO. According to the computational experiments, the performance of Levy ACO is significantly better than the original Max-min ACO and some latest Traveling Salesman Problem (TSP) solvers.

INDEX TERMS Ant colony optimization, Levy flight, Levy distribution, traveling salesman problem.

I. INTRODUCTION

A meta-heuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. Most of them based on a metaphor of some natural or man-made process, i.e, Particle Swarm Optimization (PSO) [1], [2] [3], Simulated Annealing [4]–[6], Whale Optimization Algorithm [7], [8], Moth-flame optimization [9], and Salps algorithm [10].

Invented by Marco Dorigo in 1992 [11], Ant Colony Optimization (ACO) is a meta-heuristic inspired by the behavior of real ant colonies. The first example of such an algorithm is the Ant System which was proposed using as example application of the well-known Traveling Salesman Problem (TSP) [12].

Many researchers conducted in-depth studies and proposed various improved versions of ACO. For instance, the variants of ACO such as Elite Ant Colony algorithm [12], Rank-based Ant Colony algorithm [13], Max-min Ant Colony Optimization algorithm (Max-min ACO) [14] and Ant Colony System algorithm [15] were developed. ACO algorithm was applied not only for solving TSP [15]–[19] but also for other optimization problems such as Vehicle Routing Problem [20]–[25], Quadratic Assignment Problem [26], [27], Job-shop Scheduling Problem [28]–[30], etc. In the latest ACO survey papers [31], [32], the authors stated that the most current research activities in this area were focusing on 1) incorporating ACO with other meta-heuristics such as Simulated Annealing [4]–[6], Genetic Algorithm [33], Tabu Search [34], Particle Swarm Optimization (PSO) [35], [36], and 2) various applications of ACO. Max-min ACO is still one of the state-of-the-art ACOs according to Dorigo and Stützle's survey paper [32] in 2019, and is selected in the benchmark of this manuscript.

ACO is a kind of Reinforcement Learning algorithm [15], [37]. In Figure 1 and 2, the ant, pheromone, road network in ACO are equivalent to the agent, reward, environment in Reinforcement Learning. A critical problem in Reinforcement Learning is the exploration/exploitation dilemma [38], and the proposed Levy ACO is designed to resolve it effectively.

The ants in ACO make probabilistic decisions according to the pheromones and heuristic information when constructing a feasible solution. The distribution of selection probability used to choose the next site is critical when solving a TSP [12]. Levy distribution has a property called fat-tailed or heavy-tailed, which means that their tails are thicker than other distributions. With Levy distribution, the tail portion will have a higher probability to be chosen by Levy ACO that leads to diversified solutions. Furthermore, an increase in the

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

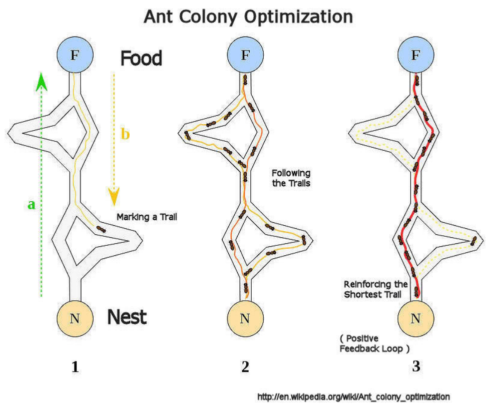
Ant Colony Optimization
F
Food
a
b
Marking a Trail
N
Nest
1
Following the Trails
2
Reinforcing the Shortest Trail
(Positive Feedback Loop)
3
http://en.wikipedia.org/wiki/Ant_colony_optimization

FIGURE 1. Ant colony optimization prototype.

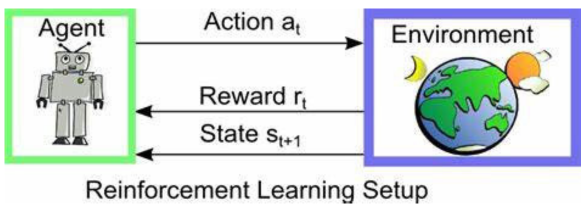

FIGURE 2. Reinforcement learning diagram.

diversity of solutions would be more likely to find the optimal solution.

Levy flight [39] is a type of random walking pattern conforming to Levy distribution. The step length of the Levy flight follows Levy distribution. Levy flight has isotropic random directions when walking in a multidimensional space. Many animals' foraging movements reveal Levy flight features, e.g., they spend most feeding time around a current food source and occasionally travel long-distance to find the next good food source efficiently [40], [41]. Levy flight mechanism has been applied to improve some meta-heuristics such as Particle Swarm Optimization (PSO) [1]–[3], Artificial Bee Colony algorithm [42], Cuckoo Search algorithm [43], [44]. It can be directly employed in spatial search approaches, for instance, PSO and Cuckoo Search algorithm, but cannot be directly applied in ACO without appropriate design.

II. METHODS

A. RESEARCH BACKGROUND

In this paper, the classic TSP is used as the application for the proposed Levy ACO. When solving a TSP, ACO utilizes positive feedback to increase the pheromone to focus on better solutions and negative feedback to decrease the pheromone to reduce the impacts of historical solutions [12]. An efficient ACO algorithm should have the capability of finding solutions as satisfactory as possible within a short period by balancing exploration and exploitation effectively.

In the ACO algorithm, the probability of being chosen for all possible candidate sites are derived from their pheromone and heuristics (attractiveness of candidate edge for TSP) in

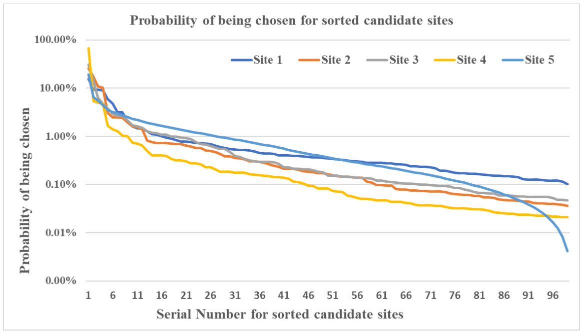

FIGURE 3. Probability of being chosen for sorted candidate sites for an instance with 100 sites.

Formula 1. Ants select a site to visit from candidate sites according to a selection probability which is a random number (uniformly distributed between 0 and 1). The probabilities of being chosen for candidate sites is determined by Formula 1 in which node i is the current site, node j and s are candidate sites, τ is the pheromone, η is the attractiveness, α and β are impact factors for τ and η, and P_{ij} is the probability of being chosen for node j from node i.

$$P_{ij} = \begin{cases} \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum(\tau_{is})^{\alpha}(\eta_{is})^{\beta}}, & j, s \in allowed \\ 0, & otherwise \end{cases} \tag{1}$$

The probabilities of being chosen for candidates are exponentially distributed due to the power function of attractiveness η in Formula 1. The candidates are sorted descending by their probabilities of being chosen. The sorted result is called the candidate list in this paper. Due to the property of the exponential function, the probability of being chosen for a candidate declines quickly and approaches when its position increases in the candidate list. Figure 3 depicts the probabilities of being chosen for the candidate sites in a TSP instance with 100 sites. Since sites are sorted descending by their probabilities of being chosen, the chances to be selected for sites with a smaller index is relatively higher than those with a greater index. For most sites, their probabilities of being chosen are less than 1% which means they almost have no chance to be selected, and this limits the selection scope and compromises the exploration of solution spaces. The proposed Levy ACO aims to improve the exploration strategy and therefore improves the diversity of solutions for achieving a better solution efficiently.

B. LEVY FLIGHT AND LEVY DISTRIBUTION

The Figure 4 illustrates the difference between Levy, normal and Cauchy distributions. Levy distribution is a fat-tailed one, and the possibility in its tail is larger than the probability at the same position in normal and Cauchy distributions. The Figure 5 from the paper [41] depicts the Brownian motions upon uniform distribution and Levy flight following Levy distribution. The search area covered by Levy flight is much larger than the one by Brownian motion within the same

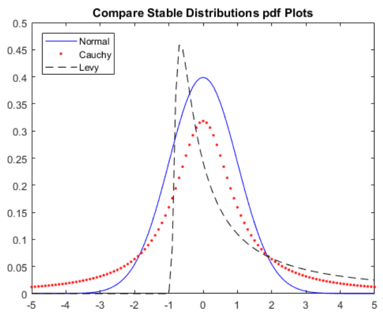

FIGURE 4. Levy distribution.

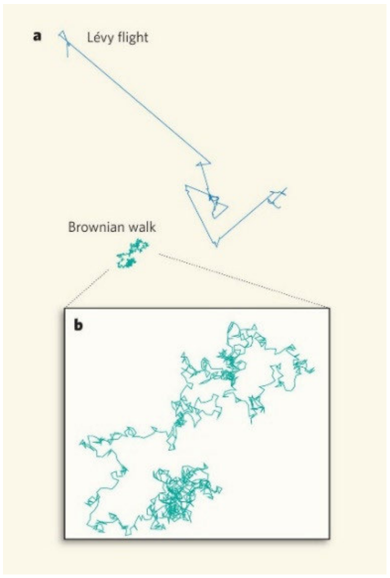

FIGURE 5. Levy flight.

1000 steps. Part b in Figure 5 depicts the zoomed-in trajectory of the corresponding Brownian motions which bounce mainly around the current spot with a step length of 1.

The flying distance is defined as the step length S for Levy flight in this paper and can be greater than 1. The standard Levy distribution is given in Formula 2.

$$S = \begin{cases} \frac{\mu}{|\nu|^{(1/\beta)}}, & \text{if } S > 1 \\ 1, & \text{else}, \end{cases} \quad \mu \sim N(0, \sigma_\mu^2), \nu \sim N(0, \sigma_\nu^2) \tag{2}$$

Formula 2 illustrates how to compute the step length S, which is the most important part of the Levy flight mechanism. Variables μ and ν follow a normal distribution while β is a fixed parameter. Step length S is a non-negative random variable following Levy distribution and associated with a direction uniformly distributed in 2- or 3-dimension depending on particular applications. In our study, no direction needs to be considered when Levy flight is one-dimensional that the step length will be applied as the ratio, altering the selection probability for choosing a candidate in Levy ACO.

Nevertheless, the calculation of a step length S for Levy flight in Formula 2 is complicated and time-consuming due to two normal distributions and one power function. The computational time of Levy ACO will increase significantly if Levy distribution is directly applied, where the Levy flight function must be called repeatedly in the solution procedure. And Formula 2 cannot be directly applied in ACO as candidate selection probability when step length S is greater than or equal to 1. Therefore, the Levy flight mechanism needs some special designs before adopted as an efficient candidate selection mechanism.

C. INTEGRATING LEVY FLIGHT WITH ACO

The solution procedure of the proposed Levy ACO is similar to the classic ACO except for the application of Levy flight to alter the random number used to select the next site. Algorithms 1 and 2 present the candidate selection mechanisms for comparison.

Algorithm 1 The Candidate Selection Mechanism in Levy ACO

```
1: Sort the sites in the candidate list by their probabilities of being selected
2: Generate a uniform random number $P_{now}$ between 0 and 1
3: Generate another uniform random number $P_{levy}$ between 0 and 1
4: if $P_{levy} \geq P_{threshold}$ then
5:     $P_{new} = 1 - A * \frac{1-P_{levy}}{1-Pthreshold} * P_{now}$
6: else
7:     $P_{new} = P_{now}$
8: end if
9: return The next site be selected using $P_{new}$ from the candidate list
```

Algorithm 2 The Candidate Selection Mechanism in Max-Min ACO

```
1: Generate a uniform random number $P_{now}$ between 0 and 1
2: return The next site be selected using $P_{now}$ from the candidate list
```

As we discussed above, sites with smaller indices in the candidate list will be selected more frequently while the rest sites have little chance to be selected when their probability of being chosen are sorted in non-increasing order, which can be observed from Figure 3. This undesired scenario can be improved in Levy ACO by using the proposed Levy flight mechanism.

The original uniformly distributed random number to select the next site will be altered by the step length of Levy flight in Levy ACO. The new random number altered by the step length should still range between 0 and 1 and will encourage Levy ACO to favor the candidates with lower probabilities of being chosen (will be discussed in more detail below). Therefore, Formula 4 is designed to appropriately implement the Levy flight mechanism in Formula 2. Step length S in Formula 2 is replaced by S_{new} in Formula 4 that only uses one single uniformly distributed variable P_{levy} for simplifying the calculation. Formula 5 shows the conversion to obtain the altered random variable P_{new}, which is the final selection probability for candidates. Since the sites in the candidate list are sorted descending by their probability of being chosen, the alteration mechanism presented by Formula 5 will "increase" the original random number so that it inclines to choose those candidates not in the front of the candidate list. As a result, these original "unfavorable" sites will more likely be selected so that diverse solutions will be found.

$$S_{new} = \begin{cases} \frac{1}{A} * \frac{1-P_{threshold}}{1-P_{levy}}, & if\ S_{new} \geq 1 \\ 1, & else \end{cases} \tag{3}$$

$$1 - P_{new} = \frac{1}{S_{new}} * (1 - P_{now}) \tag{4}$$

$$P_{new} = \begin{cases} 1 - A * \frac{1 - P_{levy}}{1-Pthreshold} * (1 - P_{now}), & if\ P_{levy} \geq P_{threshold} \\ P_{now}, & else \end{cases} \tag{5}$$

- S_{new}: New step length for Levy flight, $S_{new} \geq 1$;
- A: Fixed parameter of Levy flight Altering ratio, $A \geq 0$;
- $P_{threshold}$: Fixed parameter of Levy flight threshold, $0 < P_{threshold} < 1$;
- P_{levy}: Probability for turning on/off Levy flight altering, a uniform distribution based variable, $0 < P_{levy} < 1$;
- P_{now}: Original selection probability before Levy flight altering, a uniformly distributed variable, $0 < P_{now} < 1$;
- P_{new}: Final selection probability after Levy flight altering, $0 < P_{new} < 1$;

Formula 4 is designed to ensure that the altered random variable P_{new} will always between 0 and 1 as a selection probability.

In Formula 5, two predefined parameters are needed, namely, $P_{threshold}$ (Levy flight threshold) and A (Levy flight altering ratio). For the Levy flight mechanism, $P_{threshold}$ is defined how often it will be turned on and A decides its altering effect.

The original random variable P_{now} is uniformly distributed in [0, 1] for all candidate sites. The altered random variable P_{new} in Levy ACO decreases the probability of being chosen for the sites with smaller indices while increases those with larger indices in the candidate list to achieve more diversified solutions. The probability distribution of P_{new} with $P_{threshold} = 0$ and $A = 1$ is plotted in Figure 6. Different values of $P_{threshold}$ and A should be tuned to create a more efficient process. This paper is based on our previous paper [45].

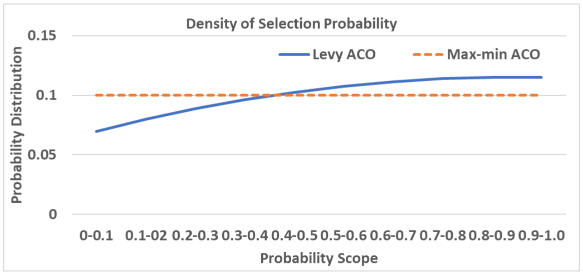

FIGURE 6. Density of selection probability for candidate site.

III. COMPUTATIONAL EXPERIMENTS

The computational experiments were carried out to benchmark the proposed Levy ACO against Max-min ACO and the recent TSP solvers.

A. DATA AND ENVIRONMENT SETTINGS

Levy ACO was implemented on the top of the open-source code (Http://www.aco-metaheuristic.org/aco-code/) developed by Thomas Stützle (the author of Max-min ACO). The source code covers Max-min ACO and some other ACO variants. Though TSP is a classic and well-studied combinatorial problem, it is an NP-hard problem used quite often for ACO performance benchmarks. Furthermore, the Max-min ACO source code used in this paper is merely designed for TSP and supports the standard TSPLIB instances with best-known solutions. Every benchmark ran 100 trials for each TSPLIB instance with the same parameter setting when concerning the stochastic property of ACO. Ten instances include gr202, lin318, gr229, gil262, kroA200, ts225, kroB200, pr226, tsp225 and pr299 from TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html) were selected for the benchmark. For a fair comparison, parameters used both in Max-min and Levy ACO were the same that he parameter vector (pheromone evaporation rate, alpha, beta, population size) was set to be (0.1, 1, 2, 50). The running time for each trial was set to a maximum of 86400 seconds to provide enough running time for the best-known solutions. Source code and running script of Max-min ACO and Levy ACO are both available at https://github.com/akeyliu/levyacotsp/.

Both Max-min ACO and Levy ACO employ a pheromone re-initialization mechanism. It monitors the solution procedure continuously and resets all pheromones to their max value when stagnation occurs, i.e. the solutions converge to a common path. This mechanism attempts to escape from local optima and find the best-known solution within enough iterations and computational time. Thanks to this mechanism, many small- or medium-scale TSPLIB instances can obtain their best-known solutions, including all ones used in this paper. During the benchmark, both Max-min ACO and Levy ACO employed other improvement procedures including nearest neighbor and the 3-opt local search. The iterations when obtaining the best-known solutions were recorded for performance comparisons and included the iterations of the 3-opt local search in Max-min ACO or Levy ACO. It is a more suitable index than the running time for algorithm performance due to the latter is more dependent on the hardware (CPU), programming language, and parallel mechanism, etc.

The computing environment for the benchmark was Windows 10 $\times$ 64, CPU 8 cores at 2.7GHz, Memory 32GB. The programming language for the implementation was C which is used for Max-min ACO.

B. PARAMETERS TUNING FOR LEVY ACO

According to our experience, smaller Levy flight threshold and Levy flight altering ratio will speed up the convergence but easy to fall into local optimal solutions. On the other hand, more exploration will help to search more solution space and yet require prolonged searching time or more iterations. It is critical to find a reasonable parameter setting to balance the exploration/exploitation to find the optima more effectively.

The corresponding computational experiments were conducted for tuning the parameters of Levy ACO, namely Levy flight threshold $P_{threshold}$ and Levy flight altering ratio A in Formula 5. Ten instances were chosen to perform parameter tuning. For each parameter setting, 100 trials were carried out, and a few metrics were applied to determine the quality of a certain parameter setting.

Parameter Levy flight threshold in Formula 5 varying from 0 to 1 with a step length of 0.1 was evaluated and analyzed. As expected, the smaller Levy flight threshold is, the more exploration is. Parameter Levy flight altering ratio in Formula 5 varying from 0 to 10 with a step length of 0.5 was evaluated as well. The result tells that the larger Levy flight altering ratio is, the more exploration is. Please note that Levy flight will be switched off if the Levy flight threshold or altering ratio is set to 0 and Levy ACO degenerates to Max-min ACO.

There were 231 parameter combinations for the Levy flight threshold and altering ratio in the experiment. All trials found the best-known solutions.

To measure the effectiveness of certain parameter setting, a metrics named improvement percentage on iterations was defined as follows:

For the given set of parameters, let $N_{maxmin(i)}$ be the average of iterations to find the best-known solution of the ith instance in 100 trials for Max-min ACO while $N_{levy(i)}$ be the one for Levy ACO, then the average of improvement percentage on iterations over K instances is defined as $\sum_{i=1}^{K}(1 - N_{levy(i)}/N_{maxmin(i)})/K$, and K was 10 in our experiment. It's conceivable that the higher average improvement percentage on iterations is, the better performance of Levy ACO is for the given parameter setting.

In Figure 7, the No.188 parameter combination is the best one where the Levy flight threshold is 0.8 and altering ratio is 9.5, and the following computational experiments employed this parameter setting.

C. BENCHMARKS BETWEEN MAX-MIN ACO AND LEVY ACO

The computational experiments were conducted by employing the ten instances and the results of Levy ACO and Max-min ACO are plotted in Figures 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 which depict the iterations when the best-known solutions of the tested TSP instances are obtained. The average of required iterations to reach the best-known solutions in Levy ACO is lower than in Max-min ACO, and this may imply that Levy ACO can achieve the best-known solutions faster than Max-min ACO does.

Three non-parametric tests called Wilcoxon, rank sums and Mann Whitney U tests were applied to check if Levy ACO and Max-min ACO perform similarly. For each TSP instance, the iterations of these two algorithms to reach the best-known solutions for all 100 trials were input to the function wilcoxon(), ranksums() and mannwhitneyu() in Python package scipy.stat. The outcomes are presented in Table 1. The performances of underlying algorithms are significantly different since the associated P values are less than 0.05 or

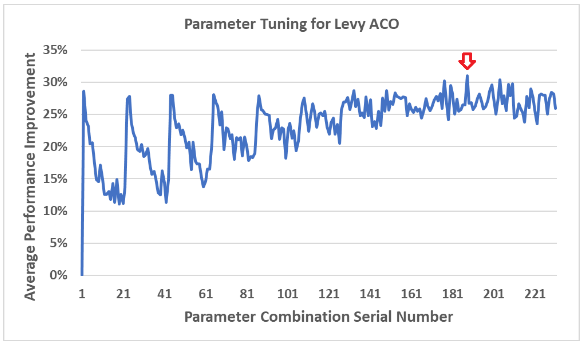

FIGURE 7. Average performance improvement percentage for parameter tuning.

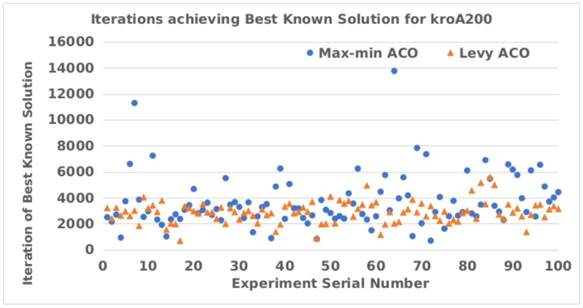

FIGURE 8. Benchmark for instance of korA200.

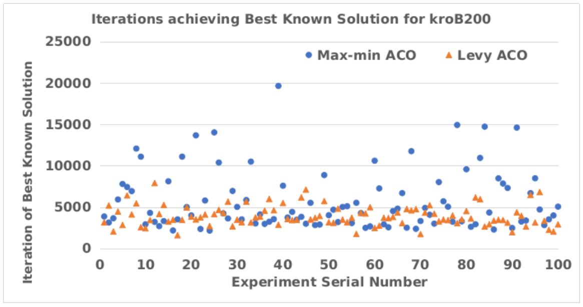

FIGURE 9. Benchmark for instance of korB200.

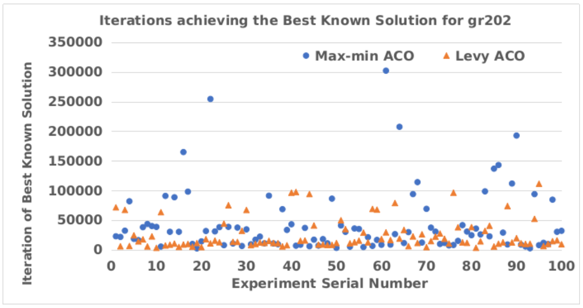

FIGURE 10. Benchmark for instance of gr202.

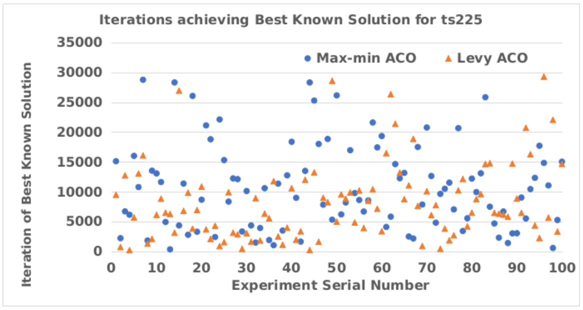

FIGURE 11. Benchmark for instance of ts225.

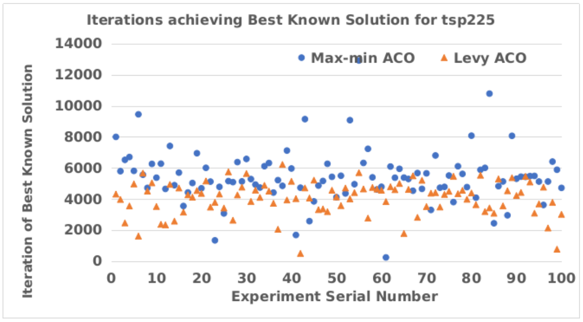

FIGURE 12. Benchmark for instance of tsp225.

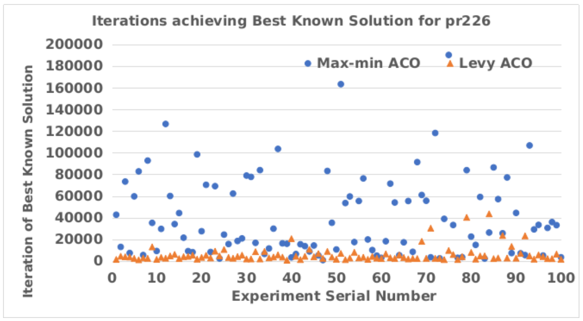

FIGURE 13. Benchmark for instance of pr226.

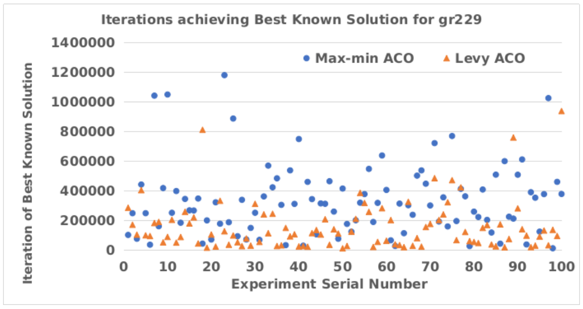

FIGURE 14. Benchmark for instance of gr229.

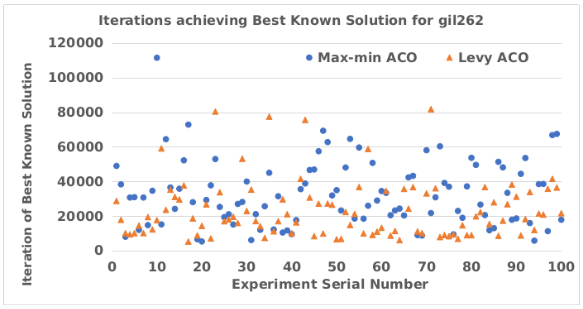

FIGURE 15. Benchmark for instance of gil262.

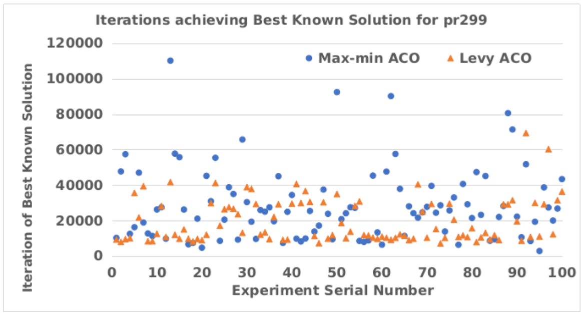

FIGURE 16. Benchmark for instance of pr299.

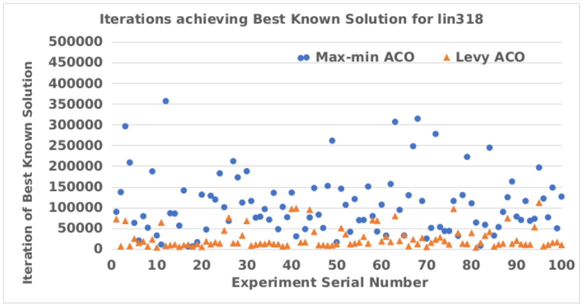

FIGURE 17. Benchmark for instance of lin318.

even 0.001 for the ten instances. Together with the results presented above, we conclude that Levy ACO can achieve best-known solutions faster than Max-min ACO.

Further statistical analyses are conducted upon the achieved results. In Table 2, the computational results are listed for Levy ACO and Max-min ACO. The following metrics are applied to both approaches:

- Average iterations: the average iterations to reach the best-known solution which used to evaluate the speed of the algorithm;

TABLE 1. Statistical testing (P value) for the benchmark analyses.

Instance	Wilcoxon	Rank sums	Mann Whitney U
gr202	1.83E-02 *	1.57E-02 *	7.86E-03 **
lin318	8.25E-25 ***	5.20E-24 ***	2.63E-24 ***
gr229	2.77E-09 ***	1.64E-10 ***	8.26E-11 ***
gil262	1.94E-04 ***	3.96E-05 ***	1.99E-05 ***
kroA200	3.53E-03 **	1.47E-02 *	7.40E-03 **
ts225	6.40E-03 **	7.57E-03 **	3.80E-03 **
kroB200	3.75E-04 ***	4.83E-03 **	2.42E-03 **
pr226	1.85E-14 ***	1.05E-20 ***	5.29E-21 ***
tsp225	7.61E-10 ***	3.89E-13 ***	1.96E-13 ***
pr299	2.71E-04 ***	6.91E-03 **	3.47E-03 **

P value: < 0.05:*; < 0.01: **; < 0.001:***

TABLE 2. Benchmark for max-min ACO and Levy ACO.

Instance	Max-min ACO		Levy ACO		Improvement Percentage	
Name	Average Iterations	Iteration Variance	Average Iterations	Iteration Variance	Average Iterations	Iteration Variance
gr202	42829.13	53355.08	25003.27	25335.15	41.62%	52.52%
lin318	110558.2	81734.86	22946.43	16704.89	79.24%	79.56%
gr229	329414.1	238874.8	155383.4	163463.5	52.83%	31.57%
gil262	32570.7	18911.63	23130.33	16504.16	28.98%	12.73%
kroA200	3683.2	2038.017	2894.11	837.4796	21.42%	58.91%
ts225	10662.05	7193.042	8144.25	6410.879	23.61%	10.87%
kroB200	5646.63	3520.784	3926.5	1234.44	30.46%	64.94%
pr226	38966.48	37236.91	5698.72	7200.127	85.38%	80.66%
tsp225	5446.88	1728.356	4072.85	1066.669	25.23%	38.28%
pr299	28392.65	20383.72	19455.73	12356.31	31.48%	39.38%
Average					42.03%	46.94%

- Iteration variance: the variance of iterations to reach the best-known solution which applied to judge the stabilization of the algorithm;

Table 2 also presents the improvement magnitude of Levy ACO over Max-min ACO for all tested TSP instances regarding the above metrics.

D. BENCHMARKS AGAINST THE LATEST TSP SOLVER

To benchmark the proposed algorithm against other ACO-based state-of-the-art TSP solvers, PSO-ACO-3Opt [35] and PACO-3Opt [36] which most cited recently in Google Scholar were selected. These algorithms hybridize Particle Swarm Optimization, Ant Colony Optimization and 3-opt. The corresponding papers presented computational experiments for a set of TSP instances, in which each instance was solved with 20 trials and within maximal 1000 iterations per trial to evaluate the performance. Specifically, instances berlin52, ch150, eil101, eil51, eil76, kroA100, kroB200, lin105, rat99 and st70 from TSPLIB were employed to perform the corresponding computational experiments.

To conduct a fair comparison, Levy ACO was applied to solve the same 10 instances mentioned above. For each instance, the iterations achieving the best-known solution for the corresponding 20 trials are listed in Table 3, and the best solutions found within 1000 iterations are presented in Table 4.

TABLE 3. Iterations for Levy ACO to achieve the best-known solutions.

Trials	berlin52	ch150	eil51	eil76	eil101	kroA100	kroB200	lin105	rat99	st70
1	**144**	**980**	**409**	**257**	1059	**542**	5684	**187**	**1231**	**141**
2	**155**	**177**	**271**	**519**	1373	**177**	5175	**198**	**458**	**144**
3	**144**	**645**	**134**	**249**	1381	**173**	3291	**196**	**919**	**557**
4	**151**	**492**	**134**	**256**	**455**	**172**	3121	**188**	**313**	**143**
5	**154**	**182**	**128**	**259**	**296**	**187**	6141	**184**	**763**	**146**
6	**145**	**480**	**146**	**638**	**451**	**172**	3355	**217**	1251	**141**
7	**145**	**479**	**141**	**383**	**317**	**343**	3470	**181**	**465**	**141**
8	**146**	**975**	**135**	**256**	1387	**176**	3479	**197**	1399	**135**
9	**126**	1326	**146**	**132**	1530	**184**	1373	**185**	1376	**147**
10	**149**	**972**	**147**	**495**	1484	**175**	4432	**214**	**613**	**139**
11	**134**	**660**	**272**	**629**	**465**	**178**	3706	**217**	**774**	**145**
12	**152**	1719	**270**	**253**	1201	**178**	3537	**201**	**933**	**144**
13	**135**	**824**	**135**	**265**	1062	**164**	3732	**222**	1260	**283**
14	**133**	**959**	**140**	**140**	**459**	**186**	6346	**198**	**308**	**141**
15	**143**	**484**	**135**	**128**	1367	**340**	2681	**189**	**765**	**285**
16	**144**	**513**	**140**	**763**	**158**	**180**	5724	**202**	**754**	**292**
17	**138**	**997**	**139**	**386**	**761**	**168**	4121	**198**	**311**	**139**
18	**147**	1585	**143**	**251**	**777**	**174**	4726	**181**	**787**	**136**
19	**135**	**492**	**144**	**642**	**460**	**176**	3955	**186**	**309**	**291**
20	**131**	1020	**139**	**259**	**149**	**173**	4249	**191**	**930**	**295**
Max iterations	155	1719	409	763	1530	542	6346	222	1399	557
Average iterations	142.55	798.05	172.4	358	829.6	210.9	4114.9	196.6	795.95	199.25

The bolt number means achieve the best-known solution within 1000 iterations.

TABLE 4. Best solutions for Levy ACO in 1000 iterations.

Trials	berlin52	ch150	eil51	eil76	eil101	kroA100	kroB200	lin105	rat99	st70
1	**7542**	**6528**	**426**	**538**	633	**21282**	29530	**14379**	1212	**675**
2	**7542**	**6528**	**426**	**538**	630	**21282**	29641	**14379**	**1211**	**675**
3	**7542**	**6528**	**426**	**538**	630	**21282**	29635	**14379**	**1211**	**675**
4	**7542**	**6528**	**426**	**538**	**629**	**21282**	29481	**14379**	**1211**	**675**
5	**7542**	**6528**	**426**	**538**	**629**	**21282**	29560	**14379**	**1211**	**675**
6	**7542**	**6528**	**426**	**538**	**629**	**21282**	29513	**14379**	1212	**675**
7	**7542**	**6528**	**426**	**538**	**629**	**21282**	29569	**14379**	**1211**	**675**
8	**7542**	**6528**	**426**	**538**	631	**21282**	29563	**14379**	1212	**675**
9	**7542**	6533	**426**	**538**	631	**21282**	29504	**14379**	1212	**675**
10	**7542**	**6528**	**426**	**538**	631	**21282**	29556	**14379**	**1211**	**675**
11	**7542**	**6528**	**426**	**538**	**629**	**21282**	29438	**14379**	**1211**	**675**
12	**7542**	6533	**426**	**538**	630	**21282**	29438	**14379**	**1211**	**675**
13	**7542**	**6528**	**426**	**538**	630	**21282**	29537	**14379**	1212	**675**
14	**7542**	**6528**	**426**	**538**	**629**	**21282**	29517	**14379**	**1211**	**675**
15	**7542**	**6528**	**426**	**538**	630	**21282**	29515	**14379**	**1211**	**675**
16	**7542**	**6528**	**426**	**538**	**629**	**21282**	29695	**14379**	**1211**	**675**
17	**7542**	**6528**	**426**	**538**	**629**	**21282**	29633	**14379**	**1211**	**675**
18	**7542**	6533	**426**	**538**	**629**	**21282**	29721	**14379**	**1211**	**675**
19	**7542**	**6528**	**426**	**538**	**629**	**21282**	29511	**14379**	**1211**	**675**
20	**7542**	6533	**426**	**538**	**629**	**21282**	29563	**14379**	**1211**	**675**
Best	**7542**	**6528**	**426**	**538**	**629**	**21282**	29438	**14379**	**1211**	**675**
Average	7542	6529	426	538	629.75	21282	29556	14379	1211.25	675
Worst	7542	6533	426	538	633	21282	29721	14379	1212	675

The bolt number means the best-known solution.

PSO-ACO-3Opt and PACO-3Opt algorithms could not reach all the best-known solutions within the given 1000 iterations according to the outcomes in paper [35] and [36]. The best solutions obtained by PSO-ACO-3Opt/PACO-3Opt and Levy ACO within the same 1000 iterations are presented in Table 5. With the same iterations, the proposed Levy ACO can achieve better solutions comparing to PSO-ACO-3Opt/PACO-3Opt.

Another latest TSP solver selected for a benchmark is the hybrid Genetic Algorithm (GA) with Multi-agent Reinforcement Learning (MARL), which also applies 2-opt and nearest insertion into the convex hull local search (NICH-LS) [46]. The same instances were solved by the underlying algorithms

TABLE 5. Best solutions obtained in 1000 iterations by compared algorithms.

Instance	Best-known	PSO-ACO-3Opt				PACO-3Opt				Levy ACO			
	Solution	Best	Average	Worst	Error%	Best	Average	Worst	Error%	Best	Average	Worst	Error%
berlin52	**7542**	**7542**	7543.2	7548	0.02%	**7542**	**7542**	**7542**	0.00%	**7542**	**7542**	**7542**	0.00%
ch150	**6528**	6538	6563.95	6622	0.55%	6570	6601.4	6620	1.12%	**6528**	6529	6533	0.02%
eil51	**426**	**426**	426.45	428	0.11%	**426**	426.35	427	0.08%	**426**	**426**	**426**	0.00%
eil76	**538**	**538**	538.3	539	0.06%	**538**	539.85	542	0.34%	**538**	**538**	**538**	0.00%
eil101	**629**	**629**	632.7	638	0.59%	**629**	630.55	639	0.25%	**629**	629.75	633	0.12%
kroA100	**21282**	21301	21445.1	21554	0.77%	**21282**	21326.8	21382	0.21%	**21282**	**21282**	**21282**	0.00%
kroA200	**29368**	29468	29646.1	29957	0.95%	29533	29644.5	29721	0.94%	29438	29556	29721	0.64%
lin105	**14379**	**14379**	14379.2	14381	0.00%	**14379**	14393	14422	0.10%	**14379**	**14379**	**14379**	0.00%
rat99	**1211**	1224	1227.4	1230	1.35%	1213	1217.1	1225	0.50%	**1211**	1211.25	1212	0.02%
st70	**675**	676	678.2	681	0.47%	676	677.85	679	0.42%	**675**	**675**	**675**	0.00%
Average					0.49%				0.40%				0.08%

Error%=(AverageSolution-BestKnownSolution)/BestKnownSolution

TABLE 6. Best solutions obtained in 10000 iterations by compared algorithms.

Instance	Best-known	MARL+NICHLS				GA-MARL+NICHLS				Levy ACO			
	Solution	Best	Average	Worst	Error%	Best	Average	Worst	Error%	Best	Average	Worst	Error%
berlin52	**7542**	**7542**	7557.4	7749	0.20%	**7542**	7550.7	7749	0.12%	**7542**	**7542**	**7542**	0.00%
ch150	**6528**	6543	6593.97	6708	1.01%	**6528**	6547.67	6640	0.30%	**6528**	**6528**	**6528**	0.00%
eil51	**426**	**426**	428.67	437	0.63%	**426**	427.4	432	0.33%	**426**	**426**	**426**	0.00%
eil76	**538**	540	548.23	555	1.90%	**538**	545.3	550	1.36%	**538**	**538**	**538**	0.00%
eil101	**629**	**629**	645.47	655	2.62%	**629**	642.6	653	2.16%	**629**	**629**	**629**	0.00%
kroA100	**21282**	**21282**	21407.47	21569	0.59%	**21282**	21345.4	21498	0.30%	**21282**	**21282**	**21282**	0.00%
kroA200	**29368**	29506	29810.67	30368	1.51%	29435	29662.1	29895	1.00%	**29368**	**29368**	**29368**	0.00%
lin105	**14379**	**14379**	14411.63	14571	0.23%	**14379**	14385.63	14479	0.05%	**14379**	**14379**	**14379**	0.00%
rat99	**1211**	**1211**	1233.57	1256	1.86%	**1211**	1223.3	1235	1.02%	**1211**	**1211**	**1211**	0.00%
st70	**675**	**675**	681.03	690	0.89%	**675**	679.43	687	0.66%	**675**	**675**	**675**	0.00%
Average					1.14%				0.73%				0.00%

Error%=(AverageSolution-BestKnownSolution)/BestKnownSolution

TABLE 7. Best/Average solutions obtained in 500 iterations by compared algorithms.

Instance	Optimal	bWOA-S	bWOA-V	WOA	LevyACO(Best)	LevyACO(Average)
kroA100	**21282**	21345	21647	21989	**21282**	21282.33
kroB100	**22141**	22406	22308	22842	**22141**	22160.47
kroC100	**20749**	21070	21320	21476	**20749**	20749.67
kroD100	**21294**	21596	22013	22083	**21294**	21310.17
kroE100	**22068**	22771	22439	22970	**22068**	22078.57

with the limit of 10,000 iterations, and the results presented in Table 6 confirm the superiority of Levy ACO.

The Whale Optimization Algorithm (WOA) and two improved versions of binary WOAs called bWOA-S and bWOA-V in paper [8] are applied to benchmark with Levy ACO under the same condition (30 trials, 20 agents, and 500 iterations), and the result is listed in Table 7. The Levy ACO performances better than all of them and can deliver the best-known solutions.

IV. CONCLUSION

The proposed Levy ACO was developed on the top of Max-min ACO by applying the Levy flight mechanism. The computational experiments reveal the superiority of the proposed approach. Levy ACO can reach the best-known solutions with fewer iterations comparing to Max-min ACO (an average 42.03% deduction in iterations and 46.94% deduction in iteration variance for all tested TSPLIB instances) and some latest TSP solvers within the given iterations.

Since ACO is a Reinforcement Learning algorithm, the proposed Levy flight mechanism can also be employed to other Reinforcement Learning or Multi-agent Reinforcement learning algorithms for the exploration/exploitation dilemma.

REFERENCES

[1] Y. Hariya, T. Kurihara, T. Shindo, and K. Jin'no, "Lévy flight PSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 2678–2684.
[2] R. Jensi and G. W. Jiji, "An enhanced particle swarm optimization with levy flight for global optimization," *Appl. Soft Comput.*, vol. 43, pp. 248–261, Jun. 2016.
[3] B. Yan, Z. Zhao, Y. Zhou, W. Yuan, J. Li, J. Wu, and D. Cheng, "A particle swarm optimization algorithm with random learning mechanism and levy flight for optimization of atomic clusters," *Comput. Phys. Commun.*, vol. 219, pp. 79–86, Oct. 2017.
[4] D. Azar, K. Fayad, and C. Daoud, "A combined ant colony optimization and simulated annealing algorithm to assess stability and fault-proneness of classes based on internal software quality attributes," *Int. J. Artif. Intell.*, vol. 14, no. 2, pp. 137–156, 2016.
[5] R. Moussi, J. Euchi, A. Yassine, and N. F. Ndiaye, "A hybrid ant colony and simulated annealing algorithm to solve the container stacking problem at seaport terminal," *Int. J. Oper. Res.*, vol. 24, no. 4, p. 399, 2015.
[6] A. M. Mohsen, "Annealing ant colony optimization with mutation operator for solving TSP," *Comput. Intell. Neurosci.*, vol. 2016, 2016.
[7] A. G. Hussien, E. H. Houssein, and A. E. Hassanien, "A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection," in *Proc. 8th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2017, pp. 166–172.

[8] A. G. Hussien, A. E. Hassanien, E. H. Houssein, M. Amin, and A. T. Azar, "New binary whale optimization algorithm for discrete optimization problems," *Eng. Optim.*, pp. 1–15, Jun. 2019.

[9] A. G. Hussien, M. Amin, and M. Abd El Aziz, "A comprehensive review of moth-flame optimisation: Variants, hybrids, and applications," *J. Exp. Theor. Artif. Intell.*, pp. 1–21, 2020.

[10] A. G. Hussien, A. E. Hassanien, and E. H. Houssein, "Swarming behaviour of salps algorithm for predicting chemical compound activities," in *Proc. 8th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2017, pp. 315–320.

[11] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, 1992.

[12] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[13] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system. A computational study," Tech. Rep., 1997.

[14] T. Stützle and H. H. Hoos, "MAX–MIN ant system," *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[15] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[16] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 622–627.

[17] Y. Li and S. Gong, "Dynamic ant colony optimisation for TSP," *Int. J. Adv. Manuf. Technol.*, vol. 22, nos. 7–8, pp. 528–533, Nov. 2003.

[18] H.-F. Wu, X.-Q. Chen, Q.-H. Mao, Q.-N. Zhang, and S.-C. Zhang, "Improved ant colony algorithm based on natural selection strategy for solving TSP problem," *J. China Inst. Commun.*, vol. 34, no. 4, pp. 165–170, 2013.

[19] I. Ariyasingha and T. G. I. Fernando, "Performance analysis of the multi-objective ant colony optimization algorithms for the traveling salesman problem," *Swarm Evol. Comput.*, vol. 23, pp. 11–26, Aug. 2015.

[20] L. M. Gambardella, E. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple colony system for vehicle routing problems with time windows," in *Proc. New Ideas Optim.*, 1999.

[21] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Adv. Eng. Inform.*, vol. 18, no. 1, pp. 41–48, 2004.

[22] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 171–176, 2009.

[23] M. Reed, A. Yiannakou, and R. Evering, "An ant colony algorithm for the multi-compartment vehicle routing problem," *Appl. Soft Comput.*, vol. 15, pp. 169–176, Feb. 2014.

[24] K. V. Narasimha, E. Kivelevitch, B. Sharma, and M. Kumar, "An ant colony optimization technique for solving min–max multi-depot vehicle routing problem," *Swarm Evol. Comput.*, vol. 13, pp. 63–73, Dec. 2013.

[25] M. Schyns, "An ant colony system for responsive dynamic vehicle routing," *Eur. J. Oper. Res.*, vol. 245, no. 3, pp. 704–718, Sep. 2015.

[26] L. M. Gambardella, E. D. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem," *J. Oper. Res. Soc.*, vol. 50, no. 2, pp. 167–176, 1999.

[27] N. Ç. Demirel and M. D. Toksarı, "Optimization of the quadratic assignment problem using an ant colony algorithm," *Appl. Math. Comput.*, vol. 183, no. 1, pp. 427–435, Dec. 2006.

[28] J. Zhang, X. Hu, X. Tan, J. H. Zhong, and Q. Huang, "Implementation of an ant colony optimization technique for job shop scheduling problem," *Trans. Inst. Meas. Control*, vol. 28, no. 1, pp. 93–108, 2006.

[29] J. Heinonen and F. Pettersson, "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem," *Appl. Math. Comput.*, vol. 187, no. 2, pp. 989–998, Apr. 2007.

[30] R.-H. Huang, C.-L. Yang, and W.-C. Cheng, "Flexible job shop scheduling with due window—A two-pheromone ant colony approach," *Int. J. Prod. Econ.*, vol. 141, no. 2, pp. 685–697, 2013.

[31] M. López-Ibáñez, T. Stützle, and M. Dorigo, "Ant colony optimization: A component-wise overview," in *Handbook Heuristics*, pp. 1–37, 2016.

[32] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in *Handbook of Metaheuristics*. Springer, 2019, pp. 311–351.

[33] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing," in *Proc. 7th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, vol. 2, Aug. 2015, pp. 428–431.

[34] Y. Drias, S. Kechid, and G. Pasi, "A novel framework for medical Web information foraging using hybrid ACO and tabu search," *J. Med. Syst.*, vol. 40, no. 1, p. 5, 2016.

[35] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, May 2015.

[36] Ş. Gülcü, M. Mahi, Ö. K. Baykan, and H. Kodaz, "A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem," *Soft Comput.*, vol. 22, no. 5, pp. 1669–1685, Mar. 2018.

[37] L. M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1995, pp. 252–260.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[39] M. F. Shlesinger and J. Klafter, "Lévy walks versus lévy flights," in *On Growth and Form*. Cham, Switzerland: Springer, 1986, pp. 279–283.

[40] G. M. Viswanathan, V. Afanasyev, S. Buldyrev, E. Murphy, P. Prince, and H. E. Stanley, "Lévy flight search patterns of wandering albatrosses," *Nature*, vol. 381, no. 6581, p. 413, 1996.

[41] G. M. Viswanathan, "Fish in Lévy-flight foraging," *Nature*, vol. 465, no. 7301, pp. 1018–1019, Jun. 2010.

[42] İ. Aydoğdu, A. Akın, and M. P. Saka, "Design optimization of real world steel space frames using artificial bee colony algorithm with levy flight distribution," *Adv. Eng. Softw.*, vol. 92, pp. 1–14, Feb. 2016.

[43] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.

[44] X.-S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014.

[45] Y. Liu and B. Cao, "Improving ant colony optimization algorithm with levy flight," in *Proc. MISTA Conf.*, 2017.

[46] M. M. Alipour, S. N. Razavi, M. R. F. Derakhshi, and M. A. Balafar, "A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem," *Neural Comput. Appl.*, vol. 30, no. 9, pp. 2935–2951, Nov. 2018.

YAHUI LIU received the B.S. degree in automation from the Taiyuan University of Science and Technology, Taiyuan, China, in 1995, and the M.S. degree in software engineering from Tongji University, Shanghai, China, in 2009, where he is currently pursuing the Ph.D. degree with the School of Software Engineering.

His research interests include ant colony optimization, reinforcement learning, multi agent reinforcement learning, vehicle routing problem (VRP), rich VRP, and time series forecasting.

BUYANG CAO received the B.S. and M.S. degrees from the University of Shanghai for Science and Technology, Shanghai, China, in 1982 and 1987, respectively, and the Ph.D. degree from the University of the Federal Armed Forces Hamburg, Hamburg, Germany, in 1991.

He is currently a Professor and a Ph.D. Student Supervisor with the College of Architecture and Urban Planning, Tongji University. Before he joined Tongji University, he had been with Esri Inc., the world largest GIS service/product provider, for decades. He led and was involved in multiple national key projects and designing/developing application systems for Fortune 500 enterprises. He possesses multi-year experiences in problem analysis, algorithm design/development, software engineering project management, and so on. His numerous research papers appeared in some top-tier international journals. His current research interests include applying AI, big data, data analytical, and algorithm development technologies to the problems from smart city and logistics arenas.

Dr. Cao was awarded by the Institute for Operations and Management Science (INFORMS) due to excellence in applying optimization techniques for solving complicated decision support problems from the real world.