Learning-Based Neural Ant Colony Optimization

Yi Liu
Fudan University
Shanghai, China
liuyi_@fudan.edu.cn

Jiang Qiu
Fudan University
Shanghai, China
21210860074@m.fudan.edu.cn

Emma Hart
Edinburgh Napier University
Edinburgh, UK
e.hart@napier.ac.uk

Yilan Yu
Fudan University
Shanghai, China
ylyu22@m.fudan.edu.cn

Zhongxue Gan
Fudan University
Shanghai, China
ganzhongxue@fudan.edu.cn

Wei Li
Fudan University
Shanghai, China
fd_liwei@fudan.edu.cn

ABSTRACT

In this paper, we propose a new ant colony optimization algorithm, called *learning-based neural ant colony optimization* (LN-ACO), which incorporates an "intelligent ant". This intelligent ant contains a convolutional neural network pre-trained on a large set of instances which is able to predict the selection probabilities of the set of possible choices at each step of the algorithm. The intelligent ant is capable of generating a solution based on knowledge learned during training, but also guides other 'traditional' ants in improving their choices during the search. As the search progresses, the intelligent ant is also influenced by the pheromones accumulated by the colony, leading to better solutions. The key idea is that if tasks or instances share common features either in terms of their search landscape or solutions, then information learned by solving one instance can be applied to substantially accelerate the search on another. We evaluate the proposed algorithm on two public datasets and one real-world test set in the path planning domain. The results demonstrate that LN-ACO is competitive in its search capability compared to other ACO methods, with a significant improvement in convergence speed.

CCS CONCEPTS

• **Computing methodologies** → *Search methodologies.*

KEYWORDS

Ant colony optimization, swarm intelligence, intelligent ant, deep learning

ACM Reference Format:
Yi Liu, Jiang Qiu, Emma Hart, Yilan Yu, Zhongxue Gan, and Wei Li. 2023. Learning-Based Neural Ant Colony Optimization. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590483

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590483

1 INTRODUCTION

Ant colony optimization (ACO) algorithms have proved successful in numerous optimization tasks [2, 27] such as travelling salesman problems [9], vehicle routing [30] and internet routing, as well as allocation and scheduling [6, 11]. Numerous variations of the basic ACO method [4, 10] have been proposed. For example, Elite AS methods [5] use information recorded from the best journeys found so far to modify the pheromone update step in order to accelerate convergence. MAX-MIN approaches [22] try to avoid premature stagnation by limiting the pheromone concentration to a fixed range, reinitializing the pheromone if the search enters a dormant state, and also modifying the update step of the pheromone matrix. In recent years, several new ACO algorithms have been proposed. PF3SACO [31] dynamically adapts the system parameters while FACO [21] attempts to reduce the difference between a newly constructed solution and a previous one, resulting in a more focused search. GSACO [15], based on an adaptive greedy strategy, continuously adjusts and changes the algorithm's parameters to increase the diversity of the population. Additionally, an enhanced heuristic ant colony optimization (EH-ACO) [13] algorithm has also been proposed. Other methods (e.g. [18, 19]) have attempted to create a pre-initialized pheromone distribution matrix. However, these methods fail to incorporate experiences from previous problem instances or modify the route choices made in subsequent iterations. Despite this range of modifications, many ant colony algorithms generally still suffer from slow convergence and issues with solution accuracy. Furthermore, like many other population-based search algorithms, ant colony approaches are typically initialized randomly each time a new instance needs to be solved and all information learned during the search to solve the instance is discarded once a solution is found. Given that many instances may contain shared features — for example, the shape and size of obstacles — there is a missed opportunity for a colony solving an instance to employ information learned from previously solving other instances. There have been some attempts in this direction. For example, [24] uses a machine learning (ML) method to predict the termination point for an unseen instance using a model trained on landscape features derived from a set of training instances. Other methods use ML to improve the initial search pheromone matrix [14, 23]. More specifically, the former approach concentrates on the orienteering problem, while the latter aims to address personalized journey route planning on multimodal public transport networks.

Our proposed method combine ML with ACO to reduce the computational time of the ant colony search and find high-quality

solutions by introducing an *intelligent ant* into a 'traditional colony'. We utilize a deep learning approach to learn a brain module for the intelligent ant: the brain consists of a convolutional neural network which is pre-trained on a large set of instances with data describing environmental information and accumulated pheromone information of a colony. The trained brain module, which effectively integrates global and local distance information, guides ant towards the most promising direction by predicting the selection probabilities of potential choices during the search process. In contrast to state-of-the-art methods that focus on pheromone optimization, the proposed approach results in an improvement of several orders of magnitude in the time taken to find a solution in some cases. We term the ant colony algorithm mixed with intelligent ant(s) *learning-based neural ant colony optimization* (LN-ACO).

The contributions of this paper are as follows:

- The first attempt to construct an intelligent ant with the ability to predict the selection probabilities of feasible nodes from a given state for a new instance. The intelligent ant contains a brain module consisting of a trained convolutional neural network that captures the historical experience of an ant colony solving many previous instances.
- A hybrid ant colony model that combines 'traditional' and 'intelligent' ants, benefiting from the exploration capabilities of the 'traditional' ants and the exploitation capability of the 'intelligent' ant.
- A comprehensive evaluation demonstrates that LN-ACO is able to obtain similar solutions to existing methods using only half the number of ants and iterations.

The paper is structured as follows. In Section 2, we introduce relevant literature. Next, in Section 3, we describe the classical ACO algorithm. Then, in Section 4, we present the framework and training process of LN-ACO. Additionally, in Section 5, we compare LN-ACO to other ant colony optimization algorithms and present findings from an ablation study. Finally, we conclude the paper in Section 6 and discuss potential directions for future research.

2 RELATED WORK

Within the field of ACO, significant effort has been devoted to enhancing the core components of ACO algorithms. This has led to a series of improved ACO algorithms, such as modifying hyperparameters, updating pheromone strategies, and improving ants' local search [20] and action transfer probability strategies [16, 17]. Other researchers have focused on directly optimizing the information in the global pheromone matrix, e.g. by restricting the range of values [22] or by initializing the pheromone matrix before search [18]. Luo *et al.* [18] suggested an unequal allocation of initial pheromones based on the relative distance of nodes and optimizing the state transition rule to solve local optima and slow convergence. In addition, they utilized the optimal and worst solutions to improve the method for updating global pheromones. Other approaches aim to integrate ACO with other traditional methods [8, 20, 25]. However, in all of these works, information learned from solving one instance cannot be re-used on new instances. The ants thus fail to learn from the colony's historical experience and from previously encountered features in the environment.

Sun *et al.* [23] proposed a machine learning model to initialize the initial pheromone matrix for meta-heuristic ACO to boost its performance. Instances are solved using a generic exact solver (CPLEX) to generate a training set. An ML model is then trained to classify edges either as part of optimal route. Similarly, He *et al.* [14] proposed using a machine learning based Max-Min Ant System (ML-MMAS) to learn a pheromone function that directly predicts global pheromone trails for the initial pheromone matrix to solve a multi-criteria journey planning problem.

In contrast to previous work, we attempt to improve the search capability of an individual, through predicting the selection probability of feasible nodes based on different environmental states, and utilizing this intelligent individual to further enhance the group's search. In comparison to the strategy of only initializing the pheromones of the colony, an intelligent ant participates in the entire search phase of the colony, forming a hybrid colony with traditional ants. The intelligent ant accelerates convergence by efficiently locating good solutions. Meanwhile, in order to avoid falling into local optima, traditional ants explore a broader space. The intelligent ant further improves its performance based on the accumulated pheromone information as the search progresses. Thus, we propose that the hybrid colony will be more effective and efficient by combining *exploration* of traditional ants with *exploitation* of intelligent ants.

3 PRELIMINARIES

The basic ACO description is given as this is used to create datasets to train the intelligent ant (see Section 4.3). In the first iteration, ants randomly choose the next step from the set of feasible actions, each of which has the same probability. The pheromone concentration is updated when all the ants have finished their search at the end of each iteration. The pheromone update formula is given by Equation (1):

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k},\ 0<\rho<1 \tag{1}$$

where the maximum number of ants is m, ρ is the volatility factor of the pheromone, τ_{ij} is the pheromone between node i to j and $\Delta\tau_{ij}^{k}$ denotes the amount of pheromone left by each of ants on route $i \rightarrow j$.

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k, & if\ ant\ k\ visited\ edge(i,j)\ in\ its\ tour \\ 0, & otherwise \end{cases} \tag{2}$$

where Q is a hyperparameter and L_k is the path length conducted by ant k.

In the following iterations, the ants choose the search direction based on a selection mechanism influenced by the pheromones left by the previous ants. Assuming that when ant k is located at node i, the set of feasible nodes of node i is $allowed_k$ denoting the nodes around node i that can be visited by ant k, the probability of going to node j is given by:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{l\in allowed_k}\tau_{il}^{\alpha}\cdot\eta_{il}^{\beta}}, & if\ j\in allowed_k \\ 0, & otherwise \end{cases} \tag{3}$$

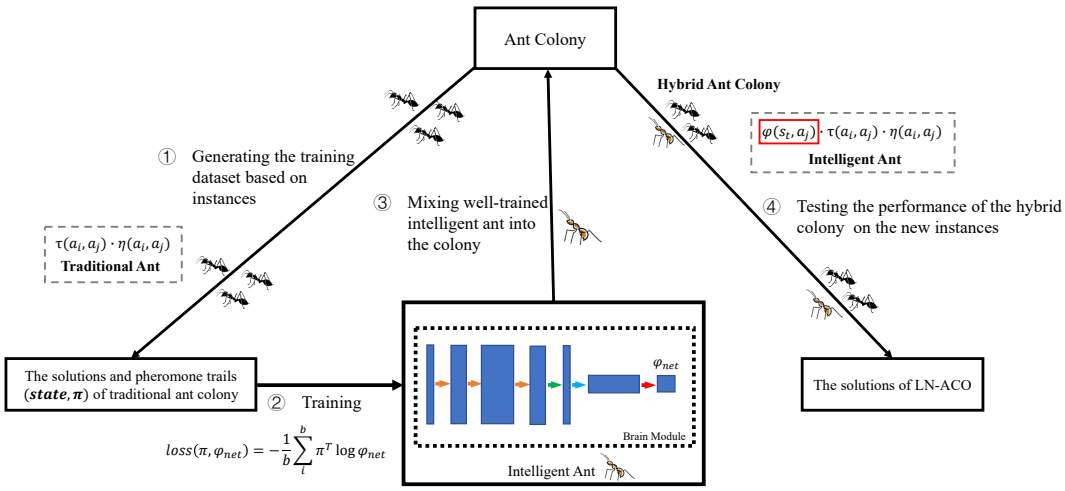


Figure 1: The framework of the LN-ACO algorithm. ① The traditional ant colony solves instances and produces data in the format $(state, \pi)$. $state$ represents environmental information and provides input to the brain module. π is the pheromone distribution of the colony. ② The training process is performed through cross-entropy loss of the predicted selection probability φ and the pheromone distribution π, as shown in Equation (6). The intelligent ant learns the solution $(state, \pi)$ and evolves the ability to predict the selection probability. ③ Following training, a hybrid ant colony consisting of intelligent ant(s) with a trained brain module and traditional ants is formed . ④ LN-ACO solves the new instances (detailed in Section 4.1).

Algorithm 1: LN-ACO algorithm.

Input : Problem instances.
Output: The near-optimal path.

```
Initialization:The problem environment, problem instances
  and the ant colony;
for iter ← 1 to MaxIteration do
    for k ← 1 to MaxNumberAnt do
        if IntelligentAnt then
            if allow_k ≠ ∅ ∧ actual_node ≠ destination then
                Well-trained intelligent ant (in Section 4.4)
                  ant_k predicts selection probability of the
                  next movements;
                Select the next move according to
                  Equation (5);
            end
        else
            if actual_node ≠ destination then
                ant_k selects the next move based on the
                  result of Equation (3);
            end
        end
        Record the path of ant_k;
    end
    Update pheromones on each iteration according to
      Equation (1);
end
```

where α and β are hyperparameters to control the relationship between the pheromone and the heuristic function; η is a heuristic function, expressed as the reciprocal of the distance between node i to node j, which is given by:

$$\eta_{ij} = 1/d_{ij} \tag{4}$$

d_{ij} means the distance of node i and node j.

4 LN-ACO

Figure 1 presents the framework of LN-ACO, which consists of four steps. Firstly, a training dataset to train the brain module of the intelligent ant is defined using solutions obtained from solving a large set of instances using traditional ACO (as described in Section 3). As previously noted, ants select the next move in the traditional method based on pheromone concentration and distance between nodes a_i and a_j, as shown in Equation (3). Secondly, using the dataset described in step (1), the brain module of the intelligent ant is trained to predict the probability distribution of pheromone trails for a given instance. With training, the brain module becomes more discriminative, and consequently, the intelligent ant predicts the probability of node selection more accurately. Thirdly, after training, the trained intelligent ant is combined with a traditional ant colony to establish a hybrid ant colony. Finally, the performance of the hybrid ant colony is evaluated on new instances that were not included in the training dataset. The intelligent ant selects its next move based on its predicted selection probability, the concentration of pheromone accumulated by the colony, and the distance between nodes, as shown in Equation (5).

The significant difference between LN-ACO and traditional ACO is that the ants in traditional ACO move based on the pheromone concentration and distance between nodes a_i and a_j, while LN-ACO adds trained intelligent ants that can predict the selection probability of nodes, therefore making a more informed decision.

In the initial stages of LN-ACO, the well-trained intelligent ant obtains better quality solutions than the traditional ant, which can accelerate the convergence speed of the colony in these early stages. To avoid the problem of falling into local optima, LN-ACO retains the functionality of traditional ants to improve the exploration ability of the hybrid ant colony. As the iterations progress, the intelligent ant and the traditional ants collectively promote the convergence of pheromones and eventually find a near-optimal solution to the problem.

The inference process of the LN-ACO algorithm is described in Algorithm 1. The hybrid ant colony of LN-ACO combines an intelligent ant and traditional ants. The intelligent ant explores the solution space based on the prediction results and previously accumulated pheromones (lines 3-7). Traditional ants follow their search process as outlined in lines 9-11. Ants's results are used to update the pheromone matrix (line 15). The intelligent ant guides traditional ants to find near-optimal solutions faster, while traditional ants explore more extensively than the intelligent ant.

4.1 Intelligent ant with brain module

The transfer probability of intelligent ant from node i to node j is:

$$a_j \sim p_{ij}^k = \begin{cases} \frac{\varphi_{ij} \cdot \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in allowed_k} \varphi_{il} \cdot \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & if\ j \in allowed_k \\ 0 & otherwise, \end{cases} \tag{5}$$

where α and β are hyperparameters; j is one of the feasible nodes of node i; k is the number of ant; p_{ij} is the transfer probability from node i to node j; τ_{ij} is the pheromone concentration accumulated by the colony; φ_{ij} is the selection probability of feasible nodes predicted by intelligent ant ant_k; and η_{ij} is the reciprocal of the distance from node i to node j.

The value of the initialized pheromone is set as 1. For cases when the number of ants and search iterations in LN-ACO are both 1, the algorithm is referred to as LN-ACO-1-1. In the LN-ACO-1-1 algorithm, the single intelligent ant calculates the transfer probability denoted by p_{ij}. It is based on the predicted selection probability by the brain module and the distance between nodes. By default, all experiments using LN-ACO have one intelligent ant in the colony. The intelligent ant combines pheromones left by the other ants during the search process. This combination aims to further optimise the predicted results, which in turn guides the colony to find a better solution. If required, this method can be generalised to include multiple intelligent ants. Specifically, during the initial search stages, the initialized pheromone provides little guidance with respect to the best search direction. The traditional ants search randomly, while the intelligent ant predicts the selection probability of the feasible choices. As a result, the intelligent ant guides the colony to find the solution more efficiently. The intelligent ant has more influence on the colony during the early part of the search. As the number of iteration increases and pheromone accumulates, traditional ants continue to search a broader space, while the intelligent ants modify their predicted results using accumulated pheromone trails. Eventually, under both traditional and intelligent ants, the colony finds a near-optimal solution. The search process of hybrid ants can be interpreted as the emergence of heterogeneous swarm intelligence.

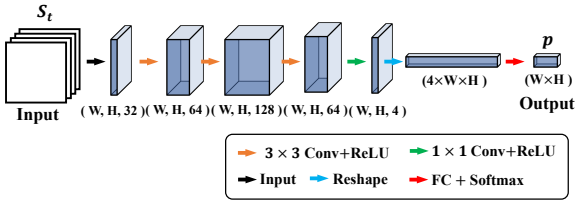


Figure 2: The architecture of the brain module.

4.2 Brain module architecture

The architecture of the brain module is illustrated in the "LN-ACO" framework of Figure 2. S_t is the environment information and the input of the brain module. p is the output of the module, which represents the selection probability. The brain module follows the typical architecture of a convolutional network. It consists of three 3×3 convolutions, a 1×1 convolution, each followed by a rectified linear unit (ReLU), and a fully connected layer. The brain module takes an input $\mathbf{S} \in \mathbb{R}^{W \times H \times 4}$ (detailed in Section 4.3), and outputs a vector $\mathbf{p} \in \mathbb{R}^{Z \times 1}$, where Z equals $W \times H$. Specifically, the input feature $\mathbf{S}$ travels through three convolutions to extract the local features in $\mathbf{S}$. In this process, the features are transformed into a higher dimensional space to discover more expressive features. Subsequently, the channel number is gradually reduced and reshaped to the expected input size of the fully connected layer. Lastly, a fully connected layer is used to fuse the global information.

It is worth noting that the brain module is more lightweight than other potential network models such as the Transformer [12] and DeepLab [7] networks which are commonly used for visual tasks. As the input feature map of our task is more structured than the typical input feature map used in visual tasks, a lightweight model is shown to be sufficient.

Algorithm 2: Training process of brain module.

Input : Training dataset (Q) of instances.
Output: Brain module.

1 Initialization: Initialize the parameters of brain module f_θ;
  for $j \leftarrow 1$ **to** ζ **do**
2 $\quad \left\{Q^i = (s_i, \pi_i) \mid i = 1, \ldots, b\right\} \leftarrow$ Sample-data (Q);
3 $\quad \varphi \leftarrow$ BrainModule (s);
4 $\quad loss \leftarrow$ LossFounction (π, φ);
5 $\quad f'_\theta \leftarrow$ TrainingBrain ($loss$);
6 **end**

4.3 Training data

We evaluate the performance of LN-ACO on the path planning problem [1, 18, 26, 29]. An environment is represented as a grid in which it is possible to move in eight directions from a given point. The distance from each node to nearby nodes is either 1 or $\sqrt{2}$ depending on the direction of a move. We use AS (as described in Section 3) to obtain training solutions and the accumulated true pheromone concentration in the $W \times H$ environment map. The resulting near-optimal paths are converted to$(\mathbf{S}, \pi)$ format and form part of the

ground truth (training) dataset. $(\mathbf{S}, \pi)$ is constructed from the state environment $\mathbf{S}$ and the information from accumulated pheromone π. The input state consists of four channels, $\mathbf{S} \in \mathbb{R}^{W \times H \times 4}$, $s_{ij} \in \{0, 1\}$. The first channel represents the origin's location of the instance; the second represents the destination; the third represents the current location of the ant and the location of the nodes on the historical route; the fourth represents the location of all obstacles on the map. The ant with the brain module (intelligent ant) predicts the selection probability of nearby nodes based on the input $\mathbf{S}$. In the training stage, the ant's predicted selection probability φ is compared with the true distribution of pheromone concentration π using cross-entropy loss calculation. The details of training process are given in Section 4.4.

4.4 Training process

The training process of an intelligent ant is shown in Algorithm 2. ζ is the iterations of training process (line 1). A mini-batch of data (s_i, π_i) is sampled from the training dataset $\mathbf{Q}$ (line 2). s_i is the input to the brain module, and π_i is the ground truth label of state i. The brain module outputs the prediction φ based on the input s_i (line 3). The loss of φ and π is calculated according to the loss function Equation (6) (line 4). Finally, the parameters θ of the brain module are updated (line 5). The training proceeds until the end condition is reached (line 6).

Cross-entropy is chosen as the loss function:

$$loss = -\frac{1}{b} \sum_{i=1}^{b} {\pi_i}^T \log \varphi_i \tag{6}$$

where b is the number of samples. The network parameters θ are adjusted based on the loss of Equation (6) to maximise the similarity between the predicted φ_i and the ground truth π_i.

5 EXPERIMENTS

5.1 Datasets

5.1.1 Various Obstacles (VO) Dataset. We created ten 10×10 environment maps with different layouts for training the LN-ACO algorithm, denoted as the VO (various obstacles) dataset. The VO dataset includes one obstacle-free map, five maps representing five different obstacles and four mixed obstacle maps. The training and validation sets are generated from the first eight environment maps, while the last two mixed obstacle maps were used for the test set. We further increased the diversity of the maps using data augmentation methods, such as horizontal and mirror rotation. In detail, maps are rotated by angles of 90, 180 and 270, and flipped horizontally and vertically during the training process. As a result, the dataset contains 80,000 training data samples, 10,000 validation and 10,000 test data samples.

5.1.2 Motion Planning (MP) Dataset. This collection of grid-world environments was proposed by [3, 28]. It consists of 24 maps. We generated the training and validation sets from the first 18 maps, and the test set from the last six maps. We resized these maps to 20×20 to complete the experiment in a reasonable amount of time. We also increased the diversity of these maps using the same data augmentation methods as on the VO dataset. As a result, 320,000 training data examples, 40,000 validation and 40,000 test data samples were generated. Following usual protocols from machine learning, the datasets are created such that the data in the training and test sets have different layouts, and the beginning and end of instances in the same maps are different, so the training and test sets do not overlap.

5.2 Experimental setups

We employed PyTorch to code all algorithms in Python 3.7 and trained them on a high performance computing server with two GeForce RTX 3090 GPUs and two 4-core dual-threaded CPUs 3.80 GHz. The inference phase of LN-ACO and the other experiments were performed on a personal computer with an Intel Core i7-8700 @ 3.20GHz CPU and an RTX 2080 SUPER GPU.

We used AS, described in Section 3, to generate the data for training the brain module. The parameters of AS on the VO dataset are set as follows: the number of ants is 30, and the number of search iterations is 20 (denoted "AS-30-20"). We denoted the different ant colony algorithms by "algorithm name - population size - number of search iterations". The parameters of AS on the MP dataset are set as follows: the number of ants is 50, and the number of search iterations is 50 ("AS-50-50"). α is 0.1, and β is 3. The pheromone evaporation rate is set to $[0.1, 0.4)$, and the hyperparameter Q is 20. We parallelised the process of collecting traditional ACO data to accelerate the training process. After 6 hours of training on the VO dataset, the loss of LN-ACO converged.

To assess the performance of the LN-ACO algorithm on the path planning problem, we used four ant colony algorithms as baselines, including AS, Elite AS, MMAS and the PPACO (as described in [18]). The performance of LN-ACO is evaluated by comparing the average path length (AP), average time consumption for each instance (Time), and the percentage of successfully solved instances (Success) in a group of 100 random instances. A smaller average path length reflects a better solution quality of the algorithm. The average time consumption is defined as the average of the time ratio of the time taken by the two algorithms across n instances, revealing the difference in the time taken by the algorithms. We also analysed the standard deviation of path lengths (SD-P) and time consumption (SD-T). We employed the Mann Whitney U test as a significance test to determine the mean difference between the experimental results for algorithms. The significance level is set to 0.05.

5.3 Results and discussion

5.3.1 Training results. We utilized LN-ACO-1-1 to evaluate the performance of the intelligent ant in the training process. We randomly selected a test group of 100 instances from each validation and test set. We evaluated the training results of the LN-ACO by four metrics. The loss quantifies the error produced by the model. A high loss value usually means that the model's output differs significantly from the true label, while a low loss value indicates that the model produces a result with a small error compared with the ground truth. The success ratio is the number of instances in which an algorithm is able to find a reasonable solution. The path ratio is the mean ratio of LN-ACO-1-1'paths versus the AS'paths. The time ratio is the average of the ratios of the LN-ACO-1-1'time consumption to the AS'time consumption for n instances. Smaller

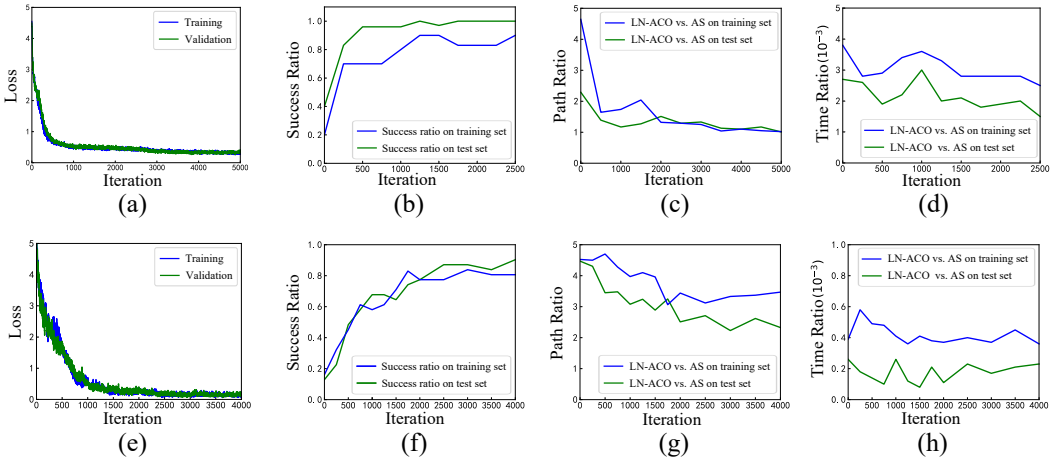


Figure 3: Training results of LN-ACO on the VO and MP datasets. Figures *a, b, c* and *d* show the variation of the brain module's loss value during the training process (Loss), the percentage of successfully solved instances among total instances (Success Ratio), the path length of LN-ACO-1-1 in comparison to the AS algorithm (Path Ratio) and the average time consumption ratios of LN-ACO and AS for 100 instances (Time Ratio) during the training iterations on the VO dataset. Similarly, figures *e, f, g* and *h* present the variation of the loss, success ratio, path ratio and time ratio on the MP dataset.

values of the path and time Ratio indicate that LN-ACO outperforms AS.

Figures 3(a), 3(b), 3(c) and 3(d) show the training results of LN-ACO on the VO dataset. Figure 3(a) shows the loss value of LN-ACO on the training and validation sets. The loss values decrease significantly in the first 500 iterations, and then start to converge. Figure 3(b) shows the increase of the success ratio of LN-ACO on the training and test sets. The success ratio increases as the number of training iterations increases. The success ratio reaches 1.0 on the training set, and 0.9 on the test set. Figure 3(c) shows the path ratio on the training and test sets. The path ratio on the training and test sets gradually converges to 1. This shows that the length of LN-ACO is almost the same as that of AS after the training process. Figure 3(d) shows the variation of the average time ratio. This result shows that LN-ACO takes much less time than AS — by a factor of almost one thousand. Figures 3(e), 3(f), 3(g) and 3(h) show the training results of LN-ACO on the MP dataset. Figure 3(e) shows the loss value of LN-ACO on the training and validation sets. The loss values decrease significantly in the first 1000 iterations before converging. Figure 3(f) shows the growth of the success ratio of LN-ACO on the training and test sets. The success ratio for the training set reaches 0.93; and the success ratio for the test set is 0.80. Figure 3(g) shows the path ratio of LN-ACO in the training and test sets. The intelligent ant's path length at the beginning of training is about 5 times longer than AS. Finally, it gradually converges to 2.32 on the training set and converges to 3.47 on the test set. Figure 3(h) shows the time ratio of LN-ACO. The time consumption of LN-ACO is much less than that of AS: in fact, it is almost one ten-thousandth of AS.

These experimental results on the VO and MP datasets show that the training process of LN-ACO-1-1 is effective, and its performance significantly improves. The initial LN-ACO-1-1 solves instances with low success ratios and poor solution quality. In Figures 3(b) and 3(f), the initial success ratios of LN-ACO-1-1 are typically less than 0.2, which means that the initial LN-ACO cannot solve most of the instances. However, its success ratios and path ratios improve dramatically after training. In Figures 3(b) and 3(f), its solution length before training is about five times longer than AS, and eventually shrinks to the same as that of AS. Figures 3(a) and 3(e) indicate that both the training and the validation loss are reduced and stabilized. Futhermore, the time ratios before and after training (Figures 3(d) and 3(h)) indicate that the search process of LN-ACO-1-1 is significantly faster than traditional ACO.

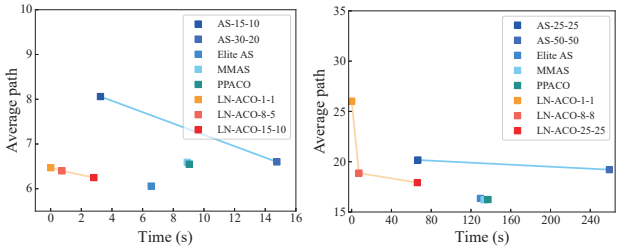
5.3.2 Comparisons with the baselines. Table 1 shows the results of LN-ACO on the VO and MP datasets. One hundred instances with different starting and destinations are randomly selected as a test group. This experiment compares the performance of LN-ACO with other competitor algorithms in terms of the average path length (AP), average time (Time) of each instance, percentage of successfully solved instances (Success), the standard deviation of path length (SD-P) and time (SD-T), and significance results (p-value).

On the VO dataset, amongst the competitor methods, PPACO-30-20 has the best AP (6.55), followed by Elite AS (AP = 6.56). Their APs have little difference from the other algorithms with the same number of populations and iterations. AS-30-20 had the longest average time consumption for each instance, taking 14.76*s*. In contrast, Elite AS has the lowest time consumption, taking only 6.56*s*. This is likely due to the fact that Elite AS records the best journey found so far to modify the update steps of the pheromone, which accelerates the convergence. Unlike other methods, LN-ACO balances path length and time consumption well. The average path length of LN-ACO-1-1 is 6.47, which is less than that of Elite AS. Additionally, the average time consumption for each instance of LN-ACO-1-1 is only 6*ms*: in contrast, Elite ACO consumes nearly

Table 1: Results of algorithms on the VO and MP datasets, including the average path length (AP) of each algorithm, average time consumption (Time) of each instance, percentage of successfully solved instances (Success (%)), the standard deviation of path length (SD-P) and time (SD-T), and significance results (p-value). The best APs are highlighted in bold.

VO Dataset						
	AP	*Time (s)*	*Success (%)*	*SD-P*	*SD-T*	*p-value*
AS-15-10	8.06	3.257	100	4.30	2.91	0.001
AS-30-20	6.60	14.768	100	3.20	10.73	**0.222**
Elite AS-30-20	6.56	6.273	100	3.13	6.95	**0.254**
MMAS-30-20	6.59	8.914	100	3.17	7.93	**0.233**
PPACO-30-20	6.55	9.038	100	3.12	9.92	**0.258**
LN-ACO-1-1	6.47	0.017	92	6.43	0.31	0.002
LN-ACO-8-5	6.40	0.754	100	3.75	0.53	**0.331**
LN-ACO-15-10	***6.25***	2.836	100	3.52	1.59	-
MP Dataset						
	AP	*Time (s)*	*Success (%)*	*SD-P*	*SD-T*	*p-value*
AS-25-25	20.17	66.156	100	12.52	40.28	0.008
AS-50-50	19.21	259.243	100	12.76	155.48	0.017
Elite AS-50-50	16.21	129.537	100	10.37	87.85	**0.493**
MMAS-50-50	16.35	133.436	100	10.42	91.62	**0.470**
PPACO-50-50	***16.24***	137.117	100	10.35	193.90	-
LN-ACO-1-1	26.02	0.045	87	54.51	0.38	0.000
LN-ACO-8-8	18.87	7.328	100	11.54	4.57	0.045
LN-ACO-25-25	17.52	66.125	100	10.75	39.11	**0.396**

1,000 times more than LN-ACO-1-1, and AS takes nearly 540 times longer. LN-ACO-15-10 has lower average path length and average time consumption than other traditional ACO algorithms with 30 ants and 20 iterations. Furthermore, the AP and average time of LN-ACO-15-10 are better than those of AS-15-10. In addition, the SD-P of LN-ACO-15-10 is similar to that of other traditional ACO algorithms with 30 ants and 20 iterations, while the SD-T of LN-ACO-15-10 is better than theirs. On the MP dataset, LN-ACO-25-25 has a significantly lower average time consumption than the baselines with 50 ants and 50 iterations. Among the baseline methods, PPACO-50-50 has the best average path length of 16.24. Note PPACO is specifically designed for path planning problems and uses various methods, including adaptive adjustment of the ratio of deterministic and random choices and introducing optimal and worst solutions to improve global pheromone updates. The average time for each instance of PPACO-50-50 is 137.117s. The average time for each instance of LN-ACO-1-1 is 45ms, while Elite ACO consumes nearly 3,000 times more than LN-ACO: the average path length of LN-ACO-1-1 is 26.02, which is only 1.60 times that of Elite AS. The average path length of LN-ACO-8-8 is lower than that of AS-50-50, and its time consumption is only 2.8% of AS-50-50. Further, LN-ACO-25-25 produces a better solution than LN-ACO-8-8 and AS-50-50. The AP of LN-ACO-25-25 is comparable to that of



(a) **(b)**

Figure 4: (a) represents the AP versus time (s) on VO dataset. (b) represents the AP versus time (s) on MP dataset.

PPACO-50-50, but its time value is half of PPACO-50-50. The SD-P of LN-ACO-25-25 is generally consistent with that of traditional ACO algorithms with 50 ants and 50 iterations. These results show that LN-ACO can achieve comparable or superior performance to the traditional ACO using a smaller population size and fewer iterations. However, since the intelligent ant can visit each node in the map only once, the success ratio of LN-ACO-1-1 is lower than that of other LN-ACO algorithms. Mann Whitney U tests were performed to obtain the results between the algorithm with the best AP (shown in bold italics) and each baseline. In Table 1, we marked the results in bold whenever two results were not significantly different with a significance level of 0.05. On the VO dataset, LN-ACO-15-10 is determined to be the best method. The results of the significance test show that there is no significant difference between AS-30-20, Elite AS-30-20, MMAS-30-20, PPACO-30-20 and LN-ACO-8-5. Similarly, on the MP dataset, PPACO-50-50 is the best method, while there is no significant difference between PPACO-50-50, Elite AS-50-50, MMAS-50-50, and LN-ACO-25-25. On both the VO and MP datasets, LN-ACO performed comparably to traditional ACOs while using half the number of ants and iterations. The experiments show that LN-ACO is more efficient in its search than other algorithms and can achieve the same results as other algorithms with only half the number of ants and iterations.

As shown in Figures 4a and 4b, the traditional ACO algorithms are distributed in the lower right corner, which means that the traditional ACO trades off solution quality against time. LN-ACO focuses on a speed/accuracy trade-off which is distributed in the lower left corner, particularly LN-ACO-15-10 on the VO dataset and the LN-ACO-8-8 and LN-ACO-25-25 on the MP dataset. Overall, LN-ACO outperforms the baselines in terms of time consumption and path quality and improves the trade-off between path optimality and search efficiency.

5.3.3 Test on real-world environment maps. This section further verifies the generality of the LN-ACO algorithm by randomly selecting real-world maps containing different types of obstacles from Google Maps. As shown in Figure 5, we compare the planning results and the convergence speed between LN-ACO, AS, Elite AS, MMAS, PPACO [18] and LN-PPACO. PPACO algorithm focuses on the path planning problem. It optimizes the results from several aspects based on domain knowledge, while LN-ACO only optimizes the AS algorithm using intelligent ants to improve the search process. Therefore, we constructed LN-PPACO for further comparison

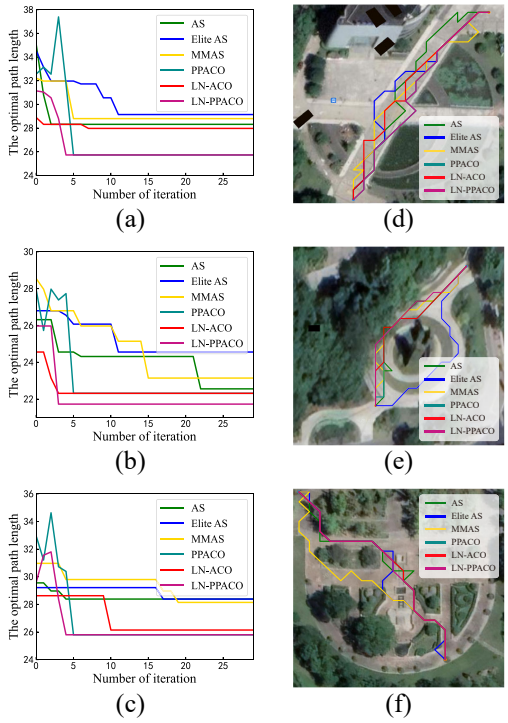


Figure 5: The convergence curves (Figures (a), (b) and (c)) and path planning results (Figures (d), (e) and (f)) of five algorithms run on a real world environment maps.

with PPACO. LN-PPACO is an improved PPACO algorithm. Its ant colony consists of one intelligent ant and traditional ants. The search of these ants follows the process of PPACO. Its ant colony consists of 30 ants, and its search iteration is 30. Figures 5(a), 5(b) and 5(c) show the convergence curves. Figures 5(e), 5(f) and 5(g) show the path planning results. Figure 5 (a) represents the length of the optimal paths searched by each ant colony algorithm during their search iterations on the map (Figure 5(d)) with the starting (19, 5) and the destination (0, 18), and the path planning results are shown in Figure 5(d). Figures 5(b), 5(e) and Figures 5(c), 5(f) show the instances from(0, 18) to (16, 7) and from (0, 0) to (17, 14), respectively. As shown in Figures 5(a), 5(b), and 5(c), the ant colony of LN-ACO obtained shorter paths than the baselines after the first iteration. Moreover, LN-ACO converges earlier than AS, Elite AS and MMAS, while the length of the optimal path of LN-ACO is the best among them. The paths generated by the LN-ACO algorithm have fewer turning points, and appear more reasonable compared to those of the baselines, as shown in Figures 5(d), 5(e) and 5(f). Moreover, the path of LN-PPACO can achieve identical performance as the PPACO, and the path length also has less fluctuation during operation. This experiment demonstrates that the convergence speed and the global optimal search capability of LN-ACO have greatly improved compared to the original algorithm and are better than Elites AS and MMAS. Moreover, the comparison of LN-PPACO to PPACO indicates that the intelligent ant promotes the colony to search more purposefully and makes the colony converge faster.

Table 2: Ablation study. Performance comparison of the intelligent ant working in different roles in ant colony on the MP dataset, including the average path length (AP) of each algorithm, the average time consumption (Time) of each instance, the percentage of successfully solved instances (Success (%)), the standard deviation of path length (SD-P) and time (SD-T).

	AP	*Time (s)*	SD-P	SD-T	*Success (%)*
LN-ACO-8-8	21.83	7.52	10.42	2.75	100
IP LN-ACO-8-8	27.13	7.04	17.11	2.85	100
LN-ACO-25-25	19.38	65.75	8.91	22.11	100
IP LN-ACO-25-25	24.13	67.46	14.70	24.01	100

5.3.4 Ablation study. We performed an ablation study to compare the performance of two configurations: LN-ACO and IP LN-ACO. The 'IP' LN-ACO is configured to use the intelligent ant only to generate the initial pheromone (IP) matrix. This intelligent ant's purpose is to generate a reasonably initialized pheromone concentration distribution that can guide future iterations. On the other hand, LN-ACO utilizes the intelligent ant in every iteration during the search process. Experiments were conducted on ten instances randomly selected from the MP dataset. We showed the results of the experiments in Table 2. Although LN-ACO and IP LN-ACO use nearly the same amount of time to solve the instances, LN-ACO has superior results in terms of AP, SD-P and SD-T. These results indicate that LN-ACO produces better and more stable results than IP LN-ACO. This approach's effectiveness is attributable to the intelligent ant's decision-making process, which is influenced by the pheromones accumulated by the colony. As a result, this improved method produces superior results overall.

6 CONCLUSION

This paper proposed a novel ant colony algorithm, LN-ACO, which integrates an intelligent ant trained on extensive datasets to predict selection probabilities for potential solutions accurately. The experimental results indicate that the proposed LN-ACO approach outperforms other ACO methods in terms of both convergence speed and solution quality. It demonstrates that the colony is capable of learning from historical information gleaned from solving past instances, and that this information can be exploited by the intelligent ant to improve convergence speed. The proposed approach contributes to a growing trend in optimization to develop methods that are capable of learning and reusing past information. Future research will be directed toward exploring the integration of the proposed brain module into other swarm intelligence algorithms.

ACKNOWLEDGMENTS

This research was supported in part by Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103), and in part by Scientific Research Development Center in Higher Education Institutions by the Ministry of Education, China (No.2021ITA10013), and in part by Engineering Research Center for Intelligent Robotics, Ji Hua Laboratory, Foshan, China (No.X190011TB190). Emma Hart is partially funded by EPSRC EP/VO26534/1.

REFERENCES

[1] Khaled Akka and Farid Khaber. 2018. Mobile robot path planning using an improved ant colony optimization. *International Journal of Advanced Robotic Systems* 15, 3 (2018), 1729881418774673.

[2] Anoushka Alavilli and Mai Vu. 2022. PLAN: a leafcutter ant colony optimization algorithm for ride-hailing services. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 4–12.

[3] Mohak Bhardwaj, Sanjiban Choudhury, and Sebastian Scherer. 2017. Learning heuristic search via imitation. In *Conference on Robot Learning*. PMLR, PMLR, 271–280.

[4] Christian Blum. 2005. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews* 2, 4 (2005), 353–373.

[5] Bernd Bullnheimer, Richard F Hartl, and Christine Strauss. 1997. A new rank based version of the Ant System. A computational study. 1 (1997).

[6] Lizhi Chen, Wei-Li Liu, and Jinghui Zhong. 2022. An efficient multi-objective ant colony optimization for task allocation of heterogeneous unmanned aerial vehicles. *Journal of Computational Science* 58 (2022), 101545.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 834–848.

[8] Xuezhen Cheng, Jiming Li, Caiyun Zheng, Jianhui Zhang, and Meng Zhao. 2021. An Improved PSO-GWO Algorithm With Chaos and Adaptive Inertial Weight for Robot Path Planning. *Frontiers in Neurorobotics* 15 (2021).

[9] Quoc Trung Dinh, Duc Dong Do, and Minh Hoàng Hà. 2021. Ants can solve the parallel drone scheduling traveling salesman problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 14–21.

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26, 1 (1996), 29–41.

[11] Marco Dorigo and Thomas Stützle. 2019. Ant colony optimization: overview and recent advances. *Handbook of metaheuristics* (2019), 311–351.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[13] Wenxiang Gao, Qing Tang, Beifa Ye, Yaru Yang, and Jin Yao. 2020. An enhanced heuristic ant colony optimization for mobile robot path planning. *Soft Computing* 24 (2020), 6139–6150.

[14] Peilan He, Guiyuan Jiang, Siew-Kei Lam, and Yidan Sun. 2022. ML-MMAS: Self-learning ant colony optimization for multi-criteria journey planning. *Information Sciences* 609 (2022), 1052–1074.

[15] Wei Li, Le Xia, Ying Huang, and Soroosh Mahmoodi. 2022. An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems. *Journal of Ambient Intelligence and Humanized Computing* (2022), 1–15.

[16] Chao Liu, Lei Wu, Xiaodong Huang, and Wensheng Xiao. 2022. Improved dynamic adaptive ant colony optimization algorithm to solve pipe routing design. *Knowledge-Based Systems* 237 (2022), 107846.

[17] Jianhua Liu, Jianguo Yang, Huaping Liu, Xingjun Tian, and Meng Gao. 2017. An improved ant colony algorithm for robot path planning. *Soft computing* 21, 19 (2017), 5829–5839.

[18] Qiang Luo, Haibao Wang, Yan Zheng, and Jingchang He. 2020. Research on path planning of mobile robot based on improved ant colony algorithm. *Neural Computing and Applications* 32, 6 (2020), 1555–1566.

[19] Teng Ren, Tianyu Luo, Binbin Jia, Bihao Yang, Ling Wang, and Lining Xing. 2023. Improved ant colony optimization for the vehicle routing problem with split pickup and split delivery. *Swarm and Evolutionary Computation* 77 (2023), 101228.

[20] Samia Sammoud and Inès Alaya. 2022. A new Ant colony optimization metaheuristic based on pheromone guided local search instead of constructive approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 13–21.

[21] Rafał Skinderowicz. 2022. Improving Ant Colony Optimization efficiency for solving large TSP instances. *Applied Soft Computing* 120 (2022), 108653.

[22] Thomas Stützle and Holger H Hoos. 2000. MAX–MIN ant system. *Future generation computer systems* 16, 8 (2000), 889–914.

[23] Yuan Sun, Sheng Wang, Yunzhuang Shen, Xiaodong Li, Andreas T Ernst, and Michael Kirley. 2022. Boosting ant colony optimization via solution prediction and machine learning. *Computers & Operations Research* 143 (2022), 105769.

[24] Marco Veluscek, Tatiana Kalganova, and Peter Broomhead. 2015. Improving ant colony optimization performance through prediction of best termination condition. In *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2394–2402.

[25] Lina Wang, Hejing Wang, Xin Yang, Yanfeng Gao, Xiaohong Cui, and Binrui Wang. 2022. Research on smooth path planning method based on improved ant colony algorithm optimized by Floyd algorithm. *Frontiers in Neurorobotics* 16 (2022).

[26] Tian Xue, Liu Li, Liu Shuang, Du Zhiping, and Pang Ming. 2021. Path planning of mobile robot based on improved ant colony algorithm for logistics. *Mathematical Biosciences and Engineering* 18, 4 (2021), 3034–3045.

[27] Xinhua Yang, Yufan Zhou, Ailing Shen, Juan Lin, and Yiwen Zhong. 2021. A hybrid ant colony optimization algorithm for the knapsack problem with a single continuous variable. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 57–65.

[28] Ryo Yonetani, Tatsunori Taniai, Mohammadamin Barekatain, Mai Nishimura, and Asako Kanezaki. 2021. Path planning using neural a* search. In *International conference on machine learning*. PMLR, PMLR, 12029–12039.

[29] Mohd Nayab Zafar and JC Mohanta. 2018. Methodology for path planning and optimization of mobile robots: A review. *Procedia computer science* 133 (2018), 141–152.

[30] Huizhen Zhang, Qinwan Zhang, Liang Ma, Ziying Zhang, and Yun Liu. 2019. A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows. *Information Sciences* 490 (2019), 166–190.

[31] Xiangbing Zhou, Hongjiang Ma, Jianggang Gu, Huiling Chen, and Wu Deng. 2022. Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. *Engineering Applications of Artificial Intelligence* 114 (2022), 105139.