2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)

Ant Colony System with Sparse Pheromone

Mengfan Jin , Guangtao Fu , Tianhao Fa , Zhibin Huang*
School of Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
Email: mengfan_jin@163.com; scottfu@bupt.edu.cn; shraem@163.com; huangzb@bupt.edu.cn

Zhiqiang Chu
Information Center
State Administration for Market Regulation
Beijing, China
Email: chuzhiqiang@samr.gov.cn

***Abstract*—Ant colony optimization algorithm is a typical meta-heuristic algorithm, which is widely used in various combinatorial optimization problems, but its high space complexity, which has become one of the main constraints affecting the application of ACO algorithm. The pheromone matrix is one of the major storage overheads. This paper based on the analysis of typical ant colony optimization algorithms, it is observed that the pheromone matrix of ACS has very strong sparseness. Therefore, SACS is proposed, whose pheromone matrix uses triplet sparse storage. In order to solve the problem of how to deal with the number of items of the initial allocated pheromone triplet and the new non-default pheromone update, this paper proposed the method based on the small fixed storage space with different replacement policies, those are, SACS-Max, SACS-Min, SACS-Rand. Many experiments show that this method basically eliminating the storage bottleneck of the pheromone matrix.**

Keywords-ACO; Pheromone Matrix; Sparse Matrix; Storage Overhead

I. INTRODUCTION

ACO has achieved good performance in many application fields, with a large number of wonderful emerging applications, such as virtual machine placement and scheduling[1], resource allocation[2], network intrusion detection[3], solving clustering problems[4], etc. With the further application of ACO, the scale of problems dealt with by ACO is becoming larger and larger. This paper focuses on the pheromone storage overhead of ACO. Through analyzing and observing the changes of the pheromone matrix during each iteration of the ACS algorithm, it is observed that pheromone matrix shows good sparsity, so an Ant Colony System for Sparse Storage (SACS) of ant pheromone is proposed to change the support of ant colony optimization algorithm from dense linear algebra mechanism to sparse linear algebra, so that the ant colony optimization algorithm can be applied to larger scale problems. The SACS algorithm proposed in this paper is based on the triplet realization of pheromone sparse storage, and the new non-default pheromone update, and proposed three replacement policies. Many experiments show that this method pheromones could be efficiently stored, and the acceptable optimized solution could still be obtained efficiently, basically eliminating the storage bottleneck of the pheromone matrix.

II. ANT COLONY SYSTEM AND TSP

Ant Colony Optimization (ACO) is a famous metaheuristic mechanism, which shows great potential in solving combinatorial optimization problems. The ACO is inspired by the behavior of biological ants as they forage for food. In the process of foraging, the ants will release a pheromone as a path marker for other ants to refer to when making path-seeking decisions, which makes the pheromone become an indirect communication medium within the ant population, and the ant's group behavior will eventually obtain the optimal path to reach the destination. Since Dorigo designed the first-generation Ant System in 1991, some influential Ant colony systems have emerged through continuous improvement, such as Elitist AS[5], Ant-Q[6], MMAS[7], ACS[8], Rank-based AS, population-based ACO[9], Beam-ACO, etc.

The TSP is the problem of finding a minimum length Hamiltonian cycle of the graph, where an Hamiltonian cycle is a closed tour visiting exactly once each of the $n = |\mathrm{N}|$ vertices of G. In fact, (published) tests of most ACO variants have been done on the TSP, which again confirms the central role of this problem in ACO research[10]. By assigning $1.4 \times \mathrm{N} \sim 1.6 \times \mathrm{N}$ items (N is the number of cities) to the pheromone triplet, SACS can reduce the space requirement of the pheromone matrix by more than 95%, although the solution error is almost the same as that of the basic ACS, basically eliminating the storage bottleneck of the pheromone matrix.

III. ANT COLONY SYSTEM WITH SPARSE PHEROMONE

The storage overhead of the ACO algorithm severely restricts its application. The ACO algorithm will generate a pheromone matrix, a heuristic information matrix and some intermediate calculation results. The heuristic information matrix can be obtained by calculation based on the original data (such as the distance matrix between cities in the TSP problem) and the pheromone matrix. Therefore, the pheromone matrix is the main storage overhead. But the pheromone matrix of ACS algorithm is sparse. In this paper, proposes that the pheromones are stored in a sparse manner instead of the matrix. In this scheme, a pheromone is stored by a triple (city S , city E , pheromone), i.e. a sparse pheromone in a manner similar to Coordinate (COO). There firstly, the mechanism of sparse pheromone storage will be introduced. Then the mechanism and policy to further reduce the storage overhead will be described, and thirdly the process of SACS algorithm will be described based on the main steps of ACS algorithm.

A. The mechanism of sparse pheromone storage

During the process of storing, it is also defaulted that the pheromones from city S to city E equal to the

2473-3636/22/$31.00 ©2022 IEEE
DOI 10.1109/DCABES57229.2022.00018

pheromones from city E to city S (the pheromone matrix is a symmetric matrix) for symmetric TSP problems. A single pheromone is represented by a triple (city S, city E, pheromone value). These triples are stored in a fixed storage space. If there are N cities, the maximum number of triples is N^2. Ideally, when all ants have the same path, N triples are needed to store pheromones to represent the path. But before all the ants agree on a path, there may be more pheromones. In the experiment of this scheme, it is found that when $10 \times N$ triples are allocated, all pheromone updates are basically saved and less pheromone updates are discarded. This also indirectly indicates that there are obvious sparse features in ACS.

Although storing pheromones in triples greatly reduces the storage overhead of pheromones, searching and addressing pheromones becomes one of the key factors affecting the performance of SACS because the reading and the update of pheromones are often random. Therefore, a hash table structure is used in SACS as an addressing mechanism for triples.

Every time we input city number S and city number E, and the additional pheromones, we first sort S and E. If $S \le E$, we keep S and E unchanged. If $S > E$, we swap the values of S and E, so that $S \le E$. Then, S and E are used to generate the hash address according to $\text{Hash\_Address} = (S \times N + E)\%M$,where $\%$ is the modular operation, M is the maximum number of items in the triplet array, and N is the number of cities.

Thus, the triplet array stores the non-default elements of the upper triangle of the original pheromone matrix. The generated hash address serves as an index to indicate access to the corresponding triplet, and by comparing the values of S and E, determines whether the data item is really required. When inserting data, if the corresponding triplet item is empty, the corresponding data is directly filled in. If it is not empty, a conflict occurs, then look for items at $(\text{Hash\_Address} + 1)$ and $(\text{Hash\_Addr ess -1})$. If there are still conflicts, continue to add 1 and subtract 1 in order to generate hash addresses, and then compare to find any free item. Discard the pheromone item until the maximum number of conflicts specified by the parameter maxHashConflicts is reached. When reading data, do something similar. If it is not found, then return the default value.

B. Mechanisms and polices to reduce storage overhead

For different TSP problems, it is impossible to predict and evaluate how many non-default pheromones there are, so what happens if there are new pheromone updates when all of the fixed assigned triples are exhausted?

SACS algorithm provides three policies to deal with this problem: maximum replacement policy (SACS-Max), minimum replacement policy (SACS-Min), and random replacement policy (SACS-Rand).

Maximum replacement policy (SACS-Max): In order to avoid falling into local optimization, when the pheromone sparse storage space is full, the element item with maximum pheromone needs to be found and removed. If the pheromone value of a path is too large, the ants always choose this path when selecting the city, which may cause the ant colony system to fall into local optimization and fail to converge quickly. Replacing the maximum gives the ants the chance to try more different paths. If a path improves the quality of the solution, it will appear in the global optimal path, and after the iteration is completed, it will be recorded when the global update is made. Therefore, it can be considered that the maximum value replacement can delete some pheromone values that are too large without missing a really useful path.

Minimum value replacement policy (SACS-Min): In order to remove useless information in the pheromone sparse storage, the minimum value of pheromone will be found and removed. Since the smallest pheromone value maybe belongs to a path that is little visited, such a path may not contribute to the optimal solution. By replacing the smallest pheromone value, these storages can be used to store the more valuable path pheromone.

Random replacement policy (SACS-Rand): As a comparison strategy, this is a replacement policy that finds any pheromone randomly in the sparse storage and removes it. This policy compares the experimental results of the previous two groups, and through random selection, it can be observed whether the maximum replacement and the minimum replacement have an effect on the quality of the SACS solution and whether it can speed up the convergence of the algorithm.

C. Operation of SACS

The process of SACS is similar to ACS. It is divided into initialization, construction of a feasible solution, obtaining a more optimized solution based on local search, and updating global pheromone. The last three subprocesses will be iterated to improve the quality of feasible solutions. Suppose there are m ants and n cities in the system. The pheromone on the path from city r to city s is represented by $\tau(r,s)$, and the city set which the ant k didn't visit at current city r is represented by $J_k(r)$.

The first step, initialization, requires allocating memory and assigning values to variables based on parameters. The initial value of the pheromone is $\tau_0 = \frac{1}{n \cdot L_0}$, where n is the size of the problem, and L_0 is the total length of a complete path obtained by selecting the nearest neighbor city by greedy algorithm. Randomly determine an initial city for each ant and delete the initial city from the never visited city set.

The SACS algorithm initializes a triplet array structure that stores pheromones.

The second step is to choose a path from the current city r_k to the next city s_k for each ant k, and remove the selected next city from the set of never visited cities. The method of selecting a city is based on the following Equation (1):

$$s = \begin{cases} argmax_{u \in J_k(r)} \{[\tau(r,u)] \cdot [\eta(r,u)]^{\beta}\} \\ \qquad ,if \;\; q < q_0 (exploitation) \\ S, otherwise (bised \quad exploration) \end{cases} \tag{1}$$

Where $q(0 \le q \le 1)$ is a random number and $q_0(0 \le q \le 1)$ is a parameter set by the system, which represents the probability of choosing the optimal path of pheromone concentration.

When $q < q_0$, through exploitation, we choose the best path from the current city to the unvisited city.

When $q > q_0$, we choose the next city by biased exploration and according to the probability $\mathrm{p}_k(\mathrm{r},\mathrm{s})$ of each city obtained by Equation (2).

$$\mathrm{p}_k(\mathrm{r},\mathrm{s}) = \begin{cases} \dfrac{[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\beta}}, & if \ \ s \in J_k(r) \\ 0, & otherwise \end{cases} \tag{2}$$

Where τ is a pheromone, $\eta = \dfrac{1}{\delta}$ is the reciprocal of the distance between cities, δ（r,s）is the city to be visited after ant k arrives at city r (in order to obtain a feasible solution), and $\beta(\beta > 0)$ is a parameter that determines the importance of the pheromone compared to the distance.

In SACS, because the pheromone is sparsely stored, the calculation of the heuristic information first needs to retrieve the triples according to the index numbers of the two cities. If there is no corresponding entry, the default value is directly used in the calculation. The element stored by a triplet array can be accessed directly. This is achieved by computing the hashed addressing. When a path is determined, each ant locally updates the pheromone $\tau(r_k, s_k)$ of the path it travels, as shown in Equation (3):

$$\tau(r_k, s_k) \leftarrow (1 - \rho)\cdot\tau(r_k, s_k) + \rho\cdot\tau_0 \tag{3}$$

ρ is a parameter representing the evaporation rate.

In SACS, the pheromone of the unrecorded path is $\tau(r_k, s_k) = \tau_0$, so according to Equation (3), its pheromone is unchanged before and after the update, and both are τ_0. Therefore, in the local update, it is first found whether the path pheromone is recorded in the triplet array. Only the recorded path pheromone needs to be updated, and no additional triple record needs to be added.

The second step selects next path in the loop iteration until each ant obtains a feasible solution.

The third step is to adjust the feasible solution obtained through 3-OPT local search. The local search attempts to perform different permutations of the three paths of the feasible solution, and selects an optimal permutation that makes the local path the shortest. Through local search, a better solution is obtained. Of course, other local search algorithms also can be used, such as 2-opt, 2.5-opt or Simulated Annealing, Tabu search, etc.

These operations do not involve updating and reading the pheromone.

The fourth step is to compute the complete path length of each ant's solution. If there exists one path length, which is less than the historical optimal solution, then update the historical optimal solution according to this complete path. Otherwise, the historically optimal solution remains unchanged. Globally update the pheromone on the path traveled by the historical optimal solution, as shown in Equation (4) and Equation (5):

$$\tau(r,s) \leftarrow \tau(r,s) + \alpha\cdot\Delta\tau(r,s) \tag{4}$$

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1}, if \ (r,s) \in gloable - best - tour \\ 0, \quad otherwise \end{cases} \tag{5}$$

α is a parameter expressing pheromone decay, and L_{gb} is the length of the globally optimal solution.

Pheromone global update is based on L_{gb} . Since the pheromone value is different from the original value after the global update of SACS, the value of the pheromone needs to be updated or inserted into the triplet array according to Equation (4). If the triplet array still has free space, then it is allocated directly and filled in by the new value. When the number of triples reaches the set limit, some of the original items need to be removed to make room for the new pheromone. The SACS algorithm provides three replacement policies.

IV. Experimental Evaluation

In the experiments of this paper, the common parameters related to the ACS algorithm are shown in Table 1.

Table 1. Main Common Parameters Used In ACS And SACS

Symbol	value	Description
q_0	$(N-20)/N$	Probability of choosing the optimal pheromone concentration path, N is the total number of cities.
β	3.0	The index of the path length, which determines the importance parameter of the pheromone compared to the distance.
α	0.2	Decay parameter for global pheromone update
ρ	0.01	Evaporation rate for local pheromone update
$iteration_{\max}$	100	The maximum value of iterations
ll_{nn}	32	Number of static candidate list items
maxHashConflicts	8	Maximum number of collisions when a hash collision occurs

In order to evaluate the effect of sparse storage of pheromone, the number of the pheromone triples is changed from $\lfloor 1.0\times \mathrm{N} \rfloor$, $\lfloor 1.2\times \mathrm{N} \rfloor$, $\lfloor 1.4\times \mathrm{N} \rfloor$, $\lfloor 1.6\times \mathrm{N} \rfloor$, $\lfloor 1.8\times \mathrm{N} \rfloor$ to $\lfloor 2.0\times \mathrm{N} \rfloor$, and collect the best solution of 100 iterations, and the convergence, the execution time, the replacement of the triples array, etc., and then evaluate experimental results.

First, compare the solution quality of ACS and SACS through experiments, as shown in Figure.1 SACS implement three replacement strategies of Max, Min and Rand. In the experiment, the number of ants was 100, 50 and 20.

As shown in Figure.1, when the number of ants is equal to 100 for 15 TSP problems, compared with the ACS, the quality of the SACS-Max solution is 2.2% lower than the best solution by an average; SACS-Min decreased by an average of 2.0%; SACS-Rand decreased by an average of 1.8%. When the number of ants is equal to 50, for 15 TSP problems, compared with the ACS, the quality of the

solutions of the SACS series decreased by 1.7%, 1.4%, 1.2% on average. When the number of ants is equal to 20, the average reduction in the quality of the solutions of the SACS series is 1.0%, 0.7%, 0.6%.

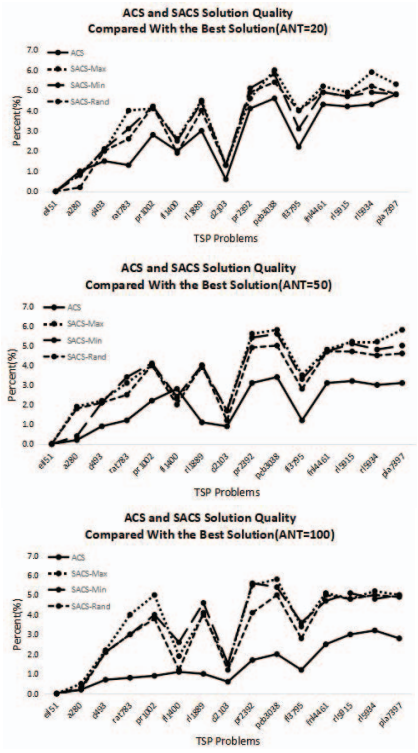

Figure 1. Comparison between optimal solutions achieved by ACS and SACS (the number of iteration steps is equal to 100, and the number of ants are 100,50 and 20 respectively. The number of pheromone triplets in SACS is equal to $2 \times N$).

It can be seen from the data that, since the pheromone is stored in a sparse triplet manner, the storage overhead of the pheromone is greatly reduced. For the SACS, the number of pheromone triplets is only $2 \times N$ (the number of pheromone matrix entries in the ACS is N^2 entries).

From the quality of the solution, when the number of ants is small, the solution quality of the SACS decreases by a smaller amount than when the number of ants is large. When the number of ants is large, the pheromone update operation is more. The discarded pheromone item will be more, which will cause the quality of the solution to decrease more.

For different numbers of ants, the reduction of the solution quality of the SACS is less than 2.5%. Considering the influence of random numbers, it can be considered that the quality of its solution is basically close to the quality of the original ACS solution. The quality of some TSP problems exceeds that of ACS. This shows that the pheromone is effectively stored in a sparse triplet array structure. It not only resolves the memory access conflict caused by local pheromone update, but also alleviates the impact of pheromone update caused by poor quality paths in the path search process. Through the replacement algorithm, the influence combination of heuristic information and pheromone information is increased, thereby greatly reducing the pheromone storage overhead.

But SACS performance is reduced compared with ACS. The main reason is: when storing sparse pheromones based on triplets, it is necessary to calculate the hash addresses and resolve possible conflicts in order to quickly find the corresponding triplets by direct hash addressing.

Although some additional calculations are appended, sparse storage overhead of pheromones reduces more than 95%, so that it can support the solution of larger-scale problems. From this point of view, SACS is still very valuable.

V. CONCLUSION

This paper proposes a new ACS algorithm, SACS, By analyzing the pheromone matrix of typical ACO algorithms, it is found that it has a certain degree of sparsity. Therefore, based on the original ACS algorithm, a SACS algorithm based on pheromone sparse storage is proposed, it is analyzed based on TSP problems. For different TSP problems, the number of required non-default pheromone triplets cannot be predicted and evaluated. When the number of fixed allocated triplets is exhausted, three policies are proposed for processing of newly added pheromone triplets: Maximum replacement policy (SACS-Max), minimum replacement policy (SACS-Min) and random replacement policy (SACS-Rand) Experiments show that the storage space requirement of pheromone is reduced by more than 95%.

REFERENCES

[1] A.Aryania, H. S. Aghdasi, L. M. Khanli, Energy-aware virtual machine consolidation algorithm based on ant colony system, Journal of Grid Computing 16 (3) (2018) 477–491.

[2] M. D. Rezende, B. S. P. De Lima, S. Guimarães, A greedy ant colony system for defensive resource assignment problems, Applied Artificial Intelligence 32 (2) (2018) 138–152.

[3] H. M. Rais, T. Mehmood, Dynamic ant colony system with three level update feature selection for intrusion detection., IJ Network Security 20 (1) (2018) 184–192.

[4] K. M. Salama, A. A. Freitas, Classification with cluster-based bayesian multi-nets using ant colony optimisation, Swarm and Evolutionary Computation 18 (2014) 54–70.

[5] M. Dorigo, V. Maniezzo, A. Colorni, The Ant System : An Autocatalytic Optimizing Process. European Conference on Artificial Life 1991.

[6] L. M. Gambardella, M. Dorigo, Ant-q: A reinforcement learning approach to the traveling salesman problem, in: Machine Learning Proceedings 1995, Elsevier, 1995, pp. 252–260.

[7] T. Stützle, H. H. Hoos, Max − min ant system, Future generation computer systems 16 (8) (2000) 889–914.

[8] M. Dorigo, L. M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on evolutionary computation 1 (1) (1997) 53–66.

[9] M. Guntsch, M. Middendorf, A population based approach for aco, in: Workshops on Applications of Evolutionary Computation, Springer, 2002, pp. 72–81.

[10] M. Dorigo, T. Stützle, Ant colony optimization: overview and recent advances, in: Handbook of metaheuristics, Springer, 2019, pp. 311–351.