A Hybrid Algorithm Based on Particle Swarm Optimization and Ant Colony Optimization Algorithm

Junliang Lu[1,2], Wei Hu[1,2](✉), Yonghao Wang[3], Lin Li[1,2], Peng Ke[1,2], and Kai Zhang[1,2]

[1] College of Computer Science and Technology,
Wuhan University of Science and Technology, Wuhan, China
ljllujunliang@foxmail.com,
{huwei,lilin,ke_peng,zhangkai}@wust.edu.cn

[2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan, China

[3] The DMT Lab, Birmingham City University, Birmingham, UK
yonghao.wang@bcu.ac.uk

Abstract. Particle swarm optimization (PSO) and Ant Colony Optimization (ACO) are two important methods of stochastic global optimization. PSO has fast global search capability with fast initial speed. But when it is close to the optimal solution, its convergence speed is slow and easy to fall into the local optimal solution. ACO can converge to the optimal path through the accumulation and update of the information with the distributed parallel global search ability. But it has slow solving speed for the lack of initial pheromone at the beginning. In this paper, the hybrid algorithm is proposed in order to use the advantages of both of the two algorithm. PSO is first used to search the global solution. When it maybe fall in local one, ACO is used to complete the search for the optimal solution according to the specific conditions. The experimental results show that the hybrid algorithm has achieved the design target with fast and accurate search.

Keywords: Particle swarm optimization · Ant colony optimization · Optimal solution

1 Introduction

Many engineering problems are proven to be NP complete or NP hard. In recent years, the intelligent heuristic optimization algorithms become more and more attractive for they can be used to search for the optimal solutions to solve NP problem partially [1]. However, due to the particularity and complexity of the various problems, each algorithm shows its advantages and disadvantages. The key challenge is the tradeoff between the time performance and optimization effect. If the algorithm is designed to find the optimal solution in limited time, it has to compromise in the optimization effectiveness and vice versa. One of the promising approach is to adopt the ideas of different algorithms and find a mixed way to be a better solution.

Particle swarm optimization (PSO) and ant colony optimization (ACO) are two important algorithms to find optimal solutions for the NP problems. PSO was proposed as an evolutionary calculation method based swarm intelligence [2]. It is an optimization method based on iteration, which is similar to genetic algorithm. PSO algorithm has adopted the concepts of group and evolution, which is derived from the research on the behaviors of the birds' feed [3]. It operates mainly based on individual adaptive values. This algorithm has simple design concept and it is easy to implement. It has strong global search ability with less experienced parameters. However, it also has obvious disadvantage that this algorithm is easy to fall into local optimal solution though it has fast global search capability with fast initial speed.

ACO is a different type of intelligent optimization algorithm. This algorithm was derived from the study of path finding behaviors of ants' activity of looking for food [4]. It also adopts the concepts of group and evolution and is based on iterative optimization. It uses a positive feedback mechanism. This algorithm could converge to the optimal solution through the pheromone that updating continuously. However, it has a slow convergence speed due to the lack of pheromone at the beginning [5].

PSO and ACO are popular algorithms. They are used to solve many problems. These two algorithms can be used individually or jointly [6–10]. Many researches focused on the performance and convergence of the two algorithms [11–14] that showing both of PSO and ACO have their advantages and disadvantages. In this paper, our design is to adopt their concepts to obtain the optimal solutions. PSO has a better performance to search the optimal solutions. This algorithm is used as the initial processing. When premature convergence is emerging, ACO is used to complete the rest of the optimization process. The key is how to identify the premature convergence. Our design focuses on the aggregation of the particles. It helps our approach to switch PSO to ACO.

This paper is organized as the follows. Section 2 provides the PSO based optimization. Section 3 describes the ACO based optimization. Section 4 depicts the hybrid algorithm. The experiments and result analysis are discussed in Sect. 5. And at last, we give the conclusions in Sect. 6.

2 Basic Principle of PSO and Its Optimization

PSO simulates birds' feeding behaviors. Imagine that a flock of birds search for food in the area randomly, in which there is only a piece of food. All the birds do not know the location of the food, but they know the distances between their own current positions to the food. The simplest and most effective method is to search the area, in which a bird is the nearest to the food. PSO algorithm was inspired from this kind of thought. It is used to solve the optimization of the problem. Each bird in this search space may be the solution for the problem. The bird can be considered as an idealized particle without quality and volume in this space. Each particle has an adaptive value that is determined by the optimization function, and there is a velocity that determines the direction and distance of the flight. Located in a S dimension of the target search space, there are m particles to form a group, where the current position of the particle i is represented as $X_i(x_{i1}, x_{i2},\ldots, x_{iS})$, current flight velocity $Vi(v_{i1}, v_{i2},\ldots, vi_S)$ and the position of $P_i(p_{i1}, p_{i2}, \ldots, p_{iS})$ in which the best position P_{pbest}(that is, with the best fitness value of the position, which is an individual extreme value). The current space of all particles have experienced the best position P_{gbest}(global extreme value). In each iteration, the particle updates itself by tracking the two extreme values. Particle i in S dimensional space update its velocity and position according to the following computation:

$$v_{is}(t+1) = w * v_{is}(t) + c_1 r_1 \left(P_{pbest}(t) - x_{is}(t)\right) + c_2 r_2 \left(P_{gbest}(t) - x_{is}(t)\right) \quad (1)$$

$$x_{is}(t+1) = x_{is}(t) + v_{is}(t+1) \quad (2)$$

Among them, $i \in [1, m]$ and $s \in [1, S]$; inertia weight w is non-negative number to control the influence from the previous velocity on the current velocity, which has very big effect on balancing the global search ability and local search ability of the algorithm. When w is small, the previous velocity has little effect on the local search ability of PSO algorithm. When the w is large, the previous velocity has great influence on the global search ability of PSO algorithm. c_1 and c_2 are learning factors, which are non-negative values. r_1 and r_2 are independent pseudo-random numbers, which obey the uniform distribution on [0, 1]. $v_{is} \in [-v_{max}, v_{max}]$, and vmax is constant. In the process of updating, the maximum velocity in each dimension of a particle is restrained as vmax, and the coordinates in each dimension of a particle is also restrained in the permitted range. At the same time, P_{pbest} and P_{gbest} are constantly updated in the iterative process. The final output is P_{gbest}, which is the optimal solution output by the algorithm.

Standard particle swarm algorithm completes the search for the optimal solution through the individual extremum and global extremum. The operations are simple with fast convergence. However, when the number of iterations increases, the particles become similar with the population convergence. It may result into local optimal solution. Such approach to track the particles' positions is replaced by other methods. The optimal solution is searched through the crossover of the individual extremum and the global extremum, or the mutation of the particles.

3 Principle of ACO Algorithm and Its Model

3.1 Optimization Principle of ACO

Ants have the ability to find the shortest path from their nest to the food without any cues. They can avoid the obstacles appropriately according to the terrain and be adapt to search a new path as the different choice. The nature of such phenomenon is that the ants will release a special kind of secretion called pheromone. Pheromone will disappear gradually over time. The remains can represent the distance of the path. And then, the ants can change their paths according the concentration of the rest pheromone. If the probability of choosing the path also high, more ants will choose this path. They will release more pheromone. When a path is chosen by more ants, it will keep more pheromone. This forms a positive feedback mechanism. Through such positive feedback mechanism, the ants can find its nest to food source of the shortest path finally. In particular, when there is an obstacle between the ants and food source, the ants can not only pass around the obstacles, and they can find the shortest path after a period of time of the positive feedback through the changes of the pheromone in different paths.

3.2 Elitist Ant System (EAS)

EAS is a relatively well global optimization of ACO algorithm [15]. Assumes that a path *e(i, j)* has the $\tau_{ij}(t)$ as the concentration of the pheromone track at time *t*. At the initial moment, the different paths have the same pheromone. Ant *k (k = 1, 2, 3, …, m)* determines its direction during the movement according to the concentration of the pheromone in each path. $p_{ij}^{k}(t)$ represents the probability that the ant *k* shifts from position *i* to position *j* at *t* time. Then the probability can be obtained as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t) * \eta_{ij}^{\beta}(t)}{\sum_{s \in allowed_k} \tau_{is}^{\alpha}(t) * \eta_{is}^{\beta}(t)}, & j \in allowed_k \\ 0, & others \end{cases} \tag{3}$$

In (3), $allowed_k = \{0, 1, \ldots, n-1\}$. In addition, $tabu_k$ is defined as a taboo list and indicates the positions that ant *k* can choose in the next step. $tabu_k(k = 1,2,3,\ldots,m)$ is used to record the current position of ant *k*. η_{ij} represents the visibility of a path *e(i, j)*, which is general set as $\eta_{ij} = \frac{1}{d_{ij}}$. d_{ij} represents the distance between position *i* and position *j*. α represents the relative importance of the tracks; β indicates the relative importance of visibility; ρ represents the persistent of the track; $1\text{–}\rho$ is the disappear degree of the information. After the ants complete a cycle, the amount of the pheromone in each path are adjusted according to the follows:

$$\tau_{ij}(t+n) = \rho\tau_{ij}(\mathrm{t}) + \sum\nolimits_{k=1}^{m} \Delta\tau_{ij}^{k}(\mathrm{t}) \tag{4}$$

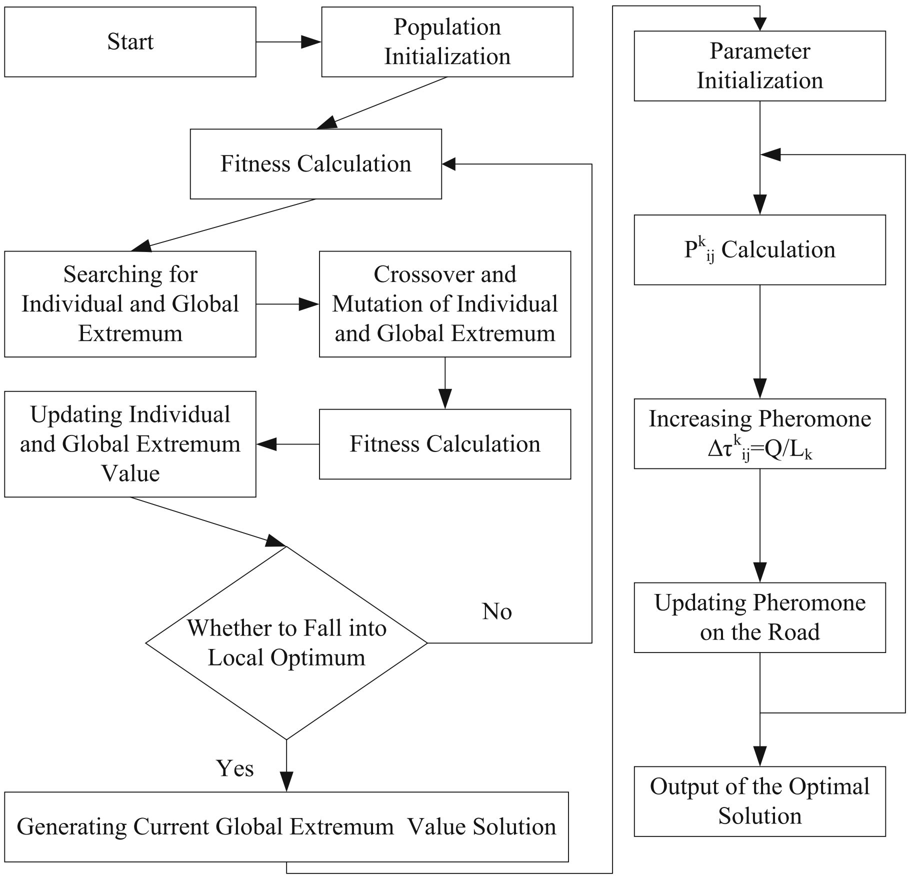
3.3 Max-Min Ant System

MMAS (Max-Min Ant System) was proposed as a general-purpose ant colony algorithm [16]. It improved ant colony algorithm in the following areas. Firstly, only the ants in the shortest path could update the pheromone after an iteration. The update methods was same to EAS. Secondly, the concentration of the pheromone was set in the range $[\tau_{min}, \tau_{max}]$ to improve the efficiency. Any values beyond this range would be forced to set in this range as τmin or τmax. Thirdly, the initial values of the pheromone in each path were set as the τmax for better search. Furthermore, a smaller evaporation coefficient was set in order to find more search paths.

4 HOA: Hybrid Optimization Algorithm

4.1 HOA Architecture

The basic idea of HOA is derived from the advantages of PSO and ACO without their defects. The process of HOA has two different stages. The former is to use PSO for the efficiency, the global search ability and the fast convergence. When the premature is emerging, ACO is used to replace PSO. PSO will output the basic information of the paths. They are the initial parameters for ACO, which means ACO can obtain a better initialization compared with the original one. In the process of the algorithm, ant colony algorithm is used for the positive feedback. Its overall framework is shown in Fig. 1.



Start
Population Initialization
Fitness Calculation
Searching for Individual and Global Extremum
Crossover and Mutation of Individual and Global Extremum
Fitness Calculation
Updating Individual and Global Extremum Value
Whether to Fall into Local Optimum
No
Yes
Generating Current Global Extremum Value Solution
Parameter Initialization
P^k_{ij} Calculation
Increasing Pheromone $\Delta\tau^k_{ij}=Q/L_k$
Updating Pheromone on the Road
Output of the Optimal Solution

Fig. 1. HOA framework

4.2 Premature Phenomena and Early Maturity of PSO

PSO is similar to many heuristic optimization algorithms just like the other heuristic algorithms. This algorithm has no operations such as selection, crossover and mutant. It means this algorithm is relatively simple with faster convergence. However, if a particle finds a current optimal node, the other particles will quickly move closer to this particle during the processing. If this node is the local optimal node, the particles will fall into the local optimal solution without escape. And the search cannot be restart. The output is a local optimal solution, which is called premature convergence phenomenon.

The existing works showed that particles in PSO will aggregate whether or not the algorithm is premature convergence or global convergence. The particles will aggregate at a special position or several special positions, which is determined by the problem itself and the selection of the fitness function. In our design, the heuristic starts from the normal distribution and fitness variance is adopted as the judgment condition to the premature convergence.

If particle swarm particle number is n, F_i represents the fitness of the *i-th* particle, F_{avg} represents the average fitness of the PSO, fitness variance σ^2 can be obtained as the follows:

$$\sigma^2 = \sum_{i=1}^{n} \left(\frac{F_i - F_{avg}}{F} \right)^2 \tag{5}$$

In (5), F is the normalized calibration factor and it is used as the limit to the size of the σ^2 variance. F can be obtained as the follows:

$$\mathrm{F} = \begin{cases} \max\left|F_i - F_{avg}\right|, \max\left|F_i - F_{avg}\right| > 1, & among\,\mathrm{i} \in [1, \mathrm{n}] \\ 1, & others \end{cases} \tag{6}$$

The variance gives the degree of aggregation of the particles in the particle swarm. The smaller the σ^2 variance is, the aggregation degree of particle swarm is big. If the algorithm does not meet the end condition and the aggregation is too large, the particle swarm algorithm falls into the so-called premature phenomenon. So when $\sigma^2 \leq h$ (h for a given constant), this algorithm will process using ACO.

5 Experiments and Results Analysis

5.1 Implementation

In order to overcome the problem that PSO is easy to fall into local optimal solution and ACO is lack of initial pheromone, HOA is designed to take use of the advantages of both PSO and ACO. The steps of HOA are as follows. First, PSO is chosen as the initial algorithm. Its parameters are set as the follows. The total number of particles is 100. The parameter h, which is used to judge whether PSO falls into a local optimal solution, is set to 15 as a constant. And then when the PSO falls into a local optimal solution, the output of PSO algorithm will be recorded as bestPath. And then ACO is initialized with the 5 times of bestPath as the initial pheromone concentration. And then ACO starts with 4 as the number of ants. At last ACO gives the output.

5.2 Experimental Results and Analysis

TSP (Travelling Salesman Problem) is an important combinatorial optimization problem, which has been proved to be NP complete. In order to verify HOA, TSP is adopted as the target problem.

Taking the urban plan as an example, the theoretical optimal value is 423.7406. The parameters of PSO algorithm are set as the follows: the total number of particles is 100; the number of cities is 30; The parameter h, which is used to judge whether PSO falls into a local optimal solution, is set to 15 as a constant. The parameters of ACO is set as the follows. α and β are set random values in [1, 2] (here $\alpha = \beta = 2$ as usual); $\rho = 0.9$ (usually from [0, 1]); the number of ants is 4.

The experimental results are shown in Table 1. PSO still contributes most iterations in HOA. PSO will compact the search space as an intermediate one and provide enough initial information for PSO. It also means that when PSO may fall into the local optimal solution during the process. At that moment, the premature is detected and PSO is switched to ACO. ACO only provides relatively less iterations. ACO can complete the whole algorithm in limited iterations. When h is suitable, HOA can complete the process quickly.

Table 1. Experimental results of HOA

α	β	h	HOA (PSO + ACO)	
			Shortest path length	Iterative times
2	2	8	431.9335	165 + 2
2	2	5	425.2667	312 + 0
2	2	10	425.9887	149 + 38
2	2	4	427.8107	167 + 16
2	2	4	425.5095	598 + 3
2	2	12	430.8101	167 + 0
2	2	7	423.7406	267 + 11
2	2	7	429.3803	153 + 5
1	2	8	427.1752	302 + 205
1	2	8	425.6807	164 + 0
1	2	7	448.0349	162 + 44
1	2	7	423.7406	285 + 6
1.5	2	7	423.7406	269 + 4

Table 2 shows the experimental results of the HOA for the length of the shortest paths and the iterations. HOA has less iterations. Furthermore, HOA can iterate repeatedly. And it can also avoid the local optimal solution trap with higher accuracy. The experimental results also show that this algorithm can complete the processing in 30 s, while the average execution time of the hybrid particle swarm algorithm exceeds 100 s. In a word, HOA has better optimization performance on the accuracy and efficiency.

Table 2. Experimental results of PSO

α	β	PSO	
		The shortest path length	Iterative times
2	2	434.8224	598
2	2	429.3803	588
2	2	424.6918	965
2	2	430.1988	867
2	2	446.2989	610
2	2	439.3706	107
2	2	445.9569	588
2	2	445.1756	783
1	2	448.6918	608
1	2	443.7406	253
1.5	2	453.1411	998
1.5	2	434.4903	615
2	2	434.8224	598

Figure 2 shows the average value comparison of HOA and the hybrid particle swarm algorithm after running 20 times. When PSO is running, HOA and hybrid PSO almost have the same curve. It means they almost have the same convergence speed. However, when the convergence is going to the end, HOA is faster. ACO provides more accurate solution.

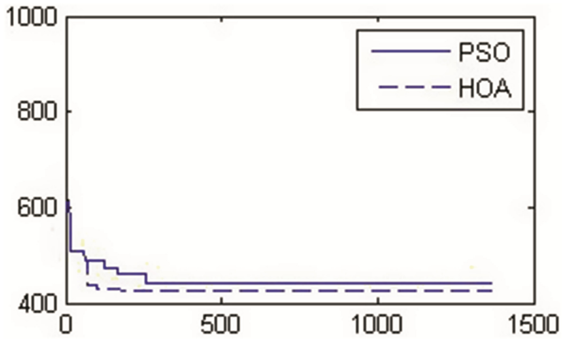


Fig. 2. Comparison of the iterative process of PSO and HOA

6 Conclusions

This paper presents the HOA algorithm based on the particle swarm algorithm and ant colony algorithm. This algorithm is designed to take use of the advantages of both PSO and ACO whereas to avoid their disadvantages. PSO improves the convergence speed and provides the input to ACO for further processing. ACO is used to avoid the premature convergence. The switch time is determined by the fitness variance. ACO helps HOA to provide the accuracy of processing. Our algorithm is verified through the experiments. TSP is used as the target problem. The experimental results shows that our algorithm can provide faster convergence speed and higher accuracy.

References

1. Amudhavel, J., Kumar, K.P., Monica, A., Bhuvaneshwari, B., Jaiganesh, S., Kumar, S.S.: A hybrid ACO-PSO based clustering protocol in VANET. In: The 2015 International Conference on Advanced Research in Computer Science Engineering & Technology. ACM Press, New York (2015). Articles 25
2. Lam, H.T., Nicolaevna, P.N., Quan, N.T.M.: A heuristic particle swarm optimization. In: The 9th Annual Conference on Genetic and Evolutionary Computation, p. 174. ACM Press, New York (2007)
3. Snyman, J.A., Kok, S.: A strongly interacting dynamic particle swarm optimizational method. In: The 9th Annual Conference on Genetic and Evolutionary Computation, p. 183. ACM Press, New York (2007)
4. Wu, C., Zhang, C., Wang, C.: Topology optimization of structures using ant colony optimization. In: The First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 601–608. ACM Press, New York (2009)
5. Chen, Y., Wong, M.L.: Optimizing stacking ensemble by an ant colony optimization approach. In: The 13th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 7–8. ACM Press, New York (2011)
6. Al-Rifaie, M.M., Bishop, M.J., Blackwell, T.: An investigation into the merger of stochastic diffusion search and particle swarm optimization. In: The 13th Annual Conference on Genetic and Evolutionary Computation, pp. 37–44. ACM Press, New York (2011)
7. Khosla, A.: Particle swarm optimization for fuzzy models. In: The 9th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 3283–3296. ACM Press, New York (2007)
8. Sinnott-Armstrong, N.A., Greene, C.S., Moore, J.H.: Fast genome-wide epistasis analysis using ant colony optimization for multifactor dimensionality reduction analysis on graphics processing units. In: The 12th Annual Conference on Genetic and Evolutionary Computation, pp. 215–216. ACM Press, New York (2010)
9. Cao, S., Qin, Y., Liu, J., Lu, R.: An ACO-Based user community preference clustering system for customized content service in broadband new media platforms. In: The 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 591–595. IEEE Press, Washington, DC (2008)
10. Rajini, A., David, V.K.: Swarm optimization and Flexible Neural Tree for microarray data classification. In: The Second International Conference on Computational Science, Engineering and Information Technology, pp. 261–268. ACM Press, New York (2012)
11. Chen, S., Montgomery, J.: A simple strategy to maintain diversity and reduce crowding in particle swarm optimization. In: The 13th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 811–812. ACM Press, New York (2011)
12. Ugolotti, R., Cagnoni, S.: Automatic tuning of standard PSO versions. In: The Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1501–1502. ACM Press, New York (2015)

13. Abdelbar, A.M.: Is there a computational advantage to representing evaporation rate in ant colony optimization as a gaussian random variable? In: The 14th Annual Conference on Genetic and Evolutionary Computation, pp. 1–8. ACM Press, New York (2012)
14. Chira, C., Pintea,C.M., Crisan, G.C., Dumitrescu, D.: Solving the linear ordering problem using ant models. In: The 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1803–1804. ACM Press, New York (2009)
15. Hemmatiana, H., Fereidoona, A., Sadollahb, A., Bahreininejad, A.: Optimization of laminate stacking sequence for minimizing weight and cost using elitist ant system optimization. Adv. Eng. Softw. **57**, 8–18 (2013)
16. Wang, G., Gong, W., Kastner, R.: Instruction scheduling using MAX-MIN ant system optimization. In: The 15th ACM Great Lakes symposium on VLSI, pp. 44–49. ACM Press, New York (2005)