Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II
WCECS 2009, October 20-22, 2009, San Francisco, USA

Hierarchical Hybrid Ant Colony Optimization for High Speed Processing

Masaya Yoshikawa, Tomohiro Taguchia

***Abstract*—This paper discusses a new hybrid ant colony optimization algorithm and its characteristics are as follows. (1) A greedy mechanism is combined to ACO in order to reduce calculation time, and new hierarchical constraints are proposed for combining it to ACO. (2) A new pheromone update rule is introduced to consider intensification and diversification. Experimental results using benchmark data prove the validity of the proposed algorithm, in comparison with the conventional ACO, of which the proposed algorithm improves the processing time.**

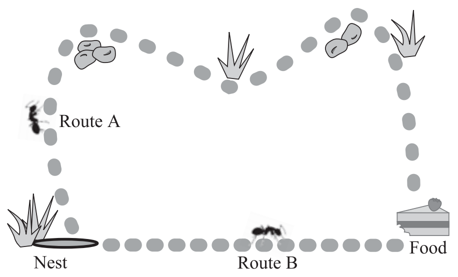
***Index Terms*—Ant Colony Optimization, Hierarchical hybrid approach, High speed processing.**

I. Introduction

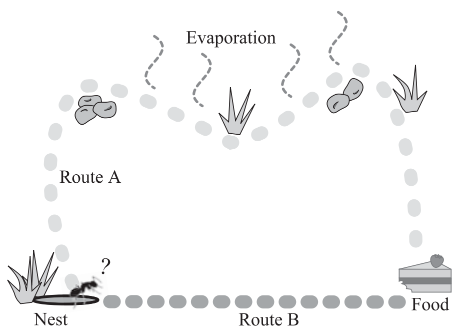
Combinational optimization problem can be applied to various engineering fields. However, most of these problems are classified into non-deterministic polynomial time (NP)-hard. In practical applications of the combinational optimization problem, many cases need semi-optimal solution. Semi-optimal solution is enough accuracy in many cases of practical applications of the combinational optimization problem.

Ant Colony Optimization (ACO) [1],[2] is an algorithm which can find such the semi-optimal solutions efficiently. Especially, the searching performance of ACO is superior to other algorithms, such as Genetic Algorithm (GA) [3]-[5] and Simulated Annealing (SA) [6], if it is applied to Traveling Salesman Problem (TSP) [7]. The search mechanism of ACO is based on positive feedback using pheromone communication among ants. All ants mark their own trails using pheromone, when the ants move. Figure.1 shows an example of placed pheromone on the route. In this example, there are two routes that pheromone was placed. Route B has more pheromone than route A. Because the pheromone amount that is marked by an ant is the same, and the evaporation speed of pheromone is also the same. Therefore, the following ants select route B, and positive feedback works as reinforcement. Thus, ACO realizes to find the shortest route using pheromone communication. However, ACO has the inherent problem of substantial processing time, because it requires a lot of repetitive calculation to obtain the semi-optimal solutions.

This work was supported in part by the Telecommunication Advancement Foundation (TAF) in JAPAN. The authors would like to thank their supports.
Masaya Yoshikawa and Tomohiro Taguchi are with Department of Information Engineering, Faculty of Science and Engineering, Meijo evolution_algorithm@ yahoo.co.jp).



(1) Examples of two routes



(2) Reinforcement by feedback
Fig.1 Example of Ant Colony Optimization

In this paper, we propose a new hybrid ACO algorithm to achieve high-speed processing. The characteristics of the proposed algorithm are as follows: (1) A greedy mechanism is combined to ACO in order to reduce calculation time, and new hierarchical constraints are proposed for combining it to ACO. (2) A new pheromone update rule is introduced to consider intensification (exploitation of the previous solutions) and diversification (exploration of the search space). Experimental results using benchmark data prove effectiveness, in comparison with the conventional ACO, of which the proposed algorithm improves the processing time. This paper is organized as follows. Section 2 briefly surveys ACO and explains the searching mechanism of ACO. The proposed algorithm is discussed in Section 3. Section 4 reports the experimental results. We conclude this study in section5.

II. Related work

ACO is a general term of the algorithm that imitates the behavior of which ants gather of food. Ant System that is proposed by Dorigo [1] is the basic model of these algorithms.

Many ACOs [8]-[14] applied to TSP are based on AS. Ant Colony System (ACS) [2] is the expanded algorithm of AS, and it is reported that ACS is one of the best algorithms when applying to TSP. Therefore, we adopt ACS as a base algorithm, hereafter in this paper, ACO represents ACS. ACO utilizes two kinds of evaluation. One is static evaluation, and the other is dynamic one. The static evaluation depends on the target problem, and usually adopts a reciprocal of distance when applying ACO to TSP. That is, the short distance is evaluated higher than the long distance in the static evaluation.

On the other hand, the dynamic evaluation adopts pheromone amount as an evaluation. ACO has two kinds of pheromone update rules. One is the local update rule, and is applied when ants move. It is defined as follows.

$$\tau(i,j) \leftarrow (1-\psi)\tau(i,j) + \psi\tau_0 \tag{1}$$

Where, ψ is a decay parameter in local update rule, $\tau(i,j)$ is a pheromone amount one the route between city i and city j, $\tau 0$ is the initial value of pheromone. Thus, local update rule adds the pheromone to the selected route between two points, when the ant moves.

The other is the global update rule, and is applied to the shortest tour when all ants complete their tours. It is defined as follows.

$$\tau(i,j) \leftarrow (1-\rho)\tau(i,j) + \rho\Delta\tau(i,j)$$

$$\Delta\tau(i,j) = \begin{cases} 1/L^+ & \text{if } (i,j) \in T^+ \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Where, T^+ is the best tour, and L^+ is the distance of the best tour. Regarding the selection of ant's move, the concretely procedure is as follows. First, the random number q between from 0 to 1 is generated. Next, q is compared with benchmark (parameter) q_0. When q is smaller than q_0, the city that has the largest value of the product is selected. Otherwise, ant k in city i selects the move to city j according to probability p^k and it is defined as follows.

$$p^k(i,j) = \frac{[\tau(i,j)][\eta(i,j)]^\beta}{\sum_{l\in n^k}[\tau(l,j)][\eta(l,j)]^\beta} \tag{3}$$

$\eta(i,j)$ is a reciprocal of the distance between city i and city j, β is a parameter which controls the balance between static evaluation value and dynamic one, and n^k is a set of un-visit cities. Therefore, the selection probability is proportional to the product of the static evaluation and the dynamic one as shown in Fig.2.

III. Hybrid Ant Colony Optimization

The processing of which each ant selects the move requires the most computing time. In the selection procedure of ant's move, the product is calculated as shown in section 2. Here, the static evaluation is constant while optimizing.

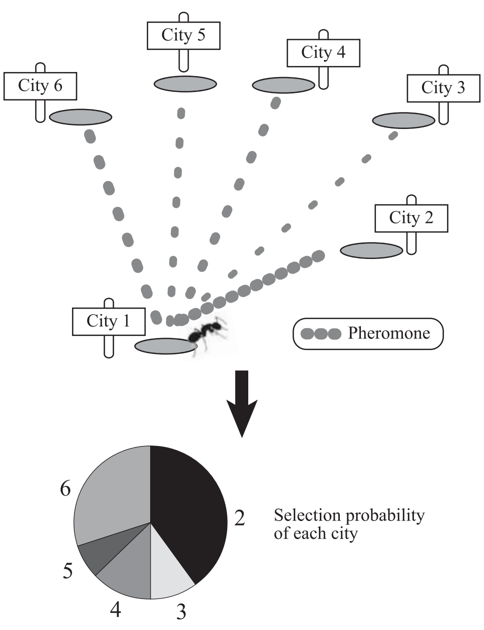


Fig.2 Selection mechanism in ACO

That is, calculations of pheromone amount cause the long computing time. It is necessary to calculate the local update rule and the product of the static evaluation and the dynamic one when each ant selects each city.

To reduce computing time, the proposed algorithm introduces greedy mechanism that utilizes only static evaluation that is constant while optimizing. The greedy mechanism can reduce the number of calculation steps, however, it is easily trapped at local optima.

In order to prevent form trapping at local optima, the proposed algorithm limits a period of which the greedy mechanism is applied. Specifically, the greedy mechanism is introduced at an early phase of optimizing. Figure.3 shows an example of the phase of which the greedy algorithm is introduced. In Fig.2, iteration represents a period that each ant completes each a tour.

Moreover, the greedy mechanism is applied to only the first half of the iteration. Figire.4 shows an example of the procedure to complete a tour. It is important to achieve the well-balanced of the trade-off between intensification and diversification to improve the searching performance. The greedy mechanism functions as intensification.

Regarding the diversification, the proposed algorithm applies the global update rule to the second shorter tour, in addition to the shortest tour. Thus, the proposed algorithm considering intensification and diversification achieves the hybrid optimization with greedy algorithm effectively.

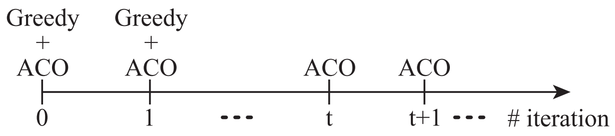


Fig.3 Example of the phase of which the greedy algorithm is introduced

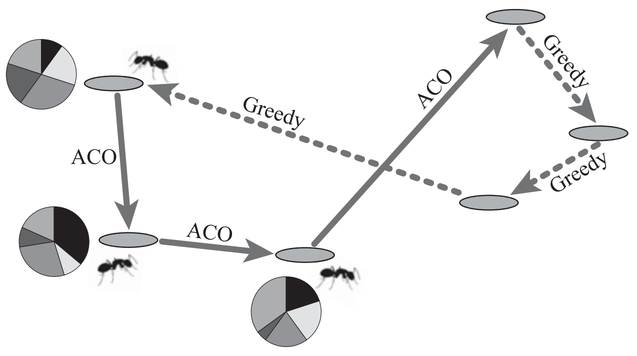


Fig.4 Example of the procedure to complete a tour

IV. Experiments and discussion

In order to evaluate the proposed algorithm, we conduct several experiments using TSP.LIB benchmark data. All experiments are executed ten times.

First, we compare the proposed algorithm with conventional ACO. Experimental results are shown in Tables.1, 2, and 3. In these experiments, parameter q is different. The parameters of Tables.1, 2, and 3 are 0.25, 0.50, and 0.75 respectively. Each value of tour distance and processing time in these tables is calculated as the average value of 10 trials.

In these tables, #iteration represents the timing that it switches ACO without greedy mechanism from ACO with one. That is, 200 in #iteration indicates that ACO is applied until 200 iterations and then Greedy is applied from the 201 iteration. Similarly, 400 in #iteration indicates that ACO is applied until 400 iterations and then Greedy is applied from the 401 iteration. Thus, the application frequency of Greedy increases when the number of #iteration in these tables grows. The proposed algorithm achieved high speed processing compared with conventional ACO as shown in these tables. That is, Greedy reduces calculation cost to obtain a solution effectively. Moreover, the proposed algorithm maintains the quality of solutions.

Next, we evaluate how to the greedy mechanism to combine with ACO. Experimental results are shown in tables, 4, 5, and 6. “Greedy + ACO” represents the approach that the greedy mechanism is introduced in the first half of the iterations. In contrast, “ACO + Greedy” represents the approach that the greedy mechanism is introduced in the latter half of the iterations.

On the other hand, “Greedy => ACO” represents the method that the greedy mechanism is introduced in the first half of composing a tour. In contrast, “ACO => Greedy” represents the method that the greedy mechanism is introduced in the first latter of composing a tour. The greedy mechanism enables to reduce the processing time even it is only included, as shown in these tables. The proposed approach shows the best performance in comparison with the other hybrid approaches.

TABLE.1
RESULTS OF $q = 0.25$

Algorithm	# iteration	Distance	Time (s)
Conventional	N/A	22049	37.84
Proposed	200	22439	37.62
	400	22456	37.52
	600	22609	35.05
	800	22492	35.86

TABLE.2
RESULTS OF $q = 0.50$

Algorithm	# iteration	Distance	Time (s)
Conventional	N/A	21876	38.69
Proposed	200	22005	38.24
	400	21983	36.15
	600	22307	35.38
	800	22365	32.85

TABLE.3
RESULTS OF $q = 0.50$

Algorithm	# iteration	Distance	Time (s)
Conventional	N/A	21768	38.43
Proposed	200	21821	38.48
	400	21810	37.48
	600	21845	35.13
	800	21970	34.27

TABLE.4
COMPARISON OF THE DIFFERENT TECHNIQUES FOR GREEDY MECHANISM
($q = 0.25$)

(1) “ACO + Greedy” and “Greedy => ACO”

# iteration	Distance	Time (s)
200	23008	35.49
400	22798	36.00
600	22733	35.97
800	22641	39.18

(2) “ACO + Greedy” and “ACO => Greedy”

# iteration	Distance	Time (s)
200	22520	33.37
400	22512	37.75
600	22627	38.65
800	22513	38.55

(3) “Greedy + ACO” and “Greedy => ACO”

# iteration	Distance	Time (s)
200	22520	33.37
400	22512	37.75
600	22627	38.65
800	22513	38.55

TABLE.5
COMPARISON OF THE DIFFERENT TECHNIQUES FOR GREEDY MECHANISM
($q = 0.50$)

(1) "ACO + Greedy" and "Greedy => ACO"

# iteration	Distance	Time (s)
200	22473	32.59
400	22303	36.77
600	22169	36.96
800	22128	36.64

(2) "ACO + Greedy" and "ACO => Greedy"

# iteration	Distance	Time (s)
200	22289	36.33
400	22243	36.31
600	21973	35.41
800	21973	36.67

(3) "Greedy + ACO" and "Greedy => ACO"

# iteration	Distance	Time (s)
200	21938	38.64
400	22057	37.54
600	22002	36.79
800	22253	35.78

TABLE.6
COMPARISON OF THE DIFFERENT TECHNIQUES FOR GREEDY MECHANISM
($q = 0.75$)

(1) "ACO + Greedy" and "Greedy => ACO"

# iteration	Distance	Time (s)
200	21998	34.13
400	21924	35.03
600	21853	37.64
800	21818	36.98

(2) "ACO + Greedy" and "ACO => Greedy"

# iteration	Distance	Time (s)
200	21926	36.50
400	21936	37.21
600	21825	37.80
800	21757	38.04

(3) "Greedy + ACO" and "Greedy => ACO"

# iteration	Distance	Time (s)
200	21798	38.29
400	21743	37.74
600	21939	37.49
800	21970	36.85

Lastly, we evaluate the diversification performance of the proposed algorithm. Experimental results are shown in Tables.7, 8, and 9. In these tables, "Best + Better" represents the proposed algorithm without greedy algorithm phase.

TABLE.7
COMPARISON OF DIVERSIFICATION ($q = 0.50$)

Global update rule	Distance	Time (s)
Best	22049	37.84
Best + Better	22353	37.83

TABLE.8
COMPARISON OF DIVERSIFICATION ($q = 0.50$)

Global update rule	Distance	Time (s)
Best	21876	38.69
Best + Better	22109	38.52

TABLE.9
COMPARISON OF DIVERSIFICATION ($q = 0.50$)

Global update rule	Distance	Time (s)
Best	21768	38.43
Best + Better	21709	36.78

That is, it modifies the conventional ACO for applying the global update rule to not only the shortest tour but also the second shorter tour. The proposed algorithm enables not only to reduce the processing time, but also to explore the search space as shown in Table.9.

V. CONCLUSION

In this paper, we proposed a new hybrid ACO algorithm. The proposed algorithm combined the greedy mechanism to ACO in order to reduce calculation time, and new hierarchical constraints were proposed for combining it to ACO. Moreover, a new pheromone update rule enabled to achieve the well-balance between intensification and diversification. Experimental results using benchmark data proved effectiveness, in comparison with the conventional ACO, of which the proposed algorithm improves the processing time.

Regarding future work, experiments using large-scale data are the most important priority. We will also introduce a new hybrid technique for diversification.

REFERENCES

[1] M.Dorigo, V.Maniezzo, A.Colorni, "Ant system: optimization by a colony of cooperating agents", IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol.26, No.1, pp.29-41, 1996.
[2] M.Dorigo, L.M.Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", IEEE Trans. Evolutionary Computation, Vol.1, No.1, pp.53-66, 1997.
[3] Holland, Adaptation in Natural Artificial Systems, the University of Michigan Press (Second edition ; MIT Press)(1992).
[4] Goldberg,D.E, Genetic algorithms in search optimization, and machine learning; Addison Wesley,(1989)
[5] M.Yoshikawa, H.Terai, "Bus-Oriented Floor- planning Technique Using Genetic Algorithm", WSEAS Transactions. on Circuits and System, Issue2, Vol.6, pp.253-258, 2007.
[6] R.A. Rutenbar, "Simulated annealing algorithms: an overview, IEEE Circuits and Devices Magazine", Volume 5, Issue 1, pp.19-26, 1989.
[7] J.Grefenstette et al., Genetic Algorithm for the Traveling Salesman Problem, Proc. of 1st Int. Conf. on Genetic Algorithms and their applications, pp.160-168, 1985.
[8] M.Yoshikawa: Hardware-oriented Ant Colony Optimization Considering Intensification and Diversification, Witold Bednorz (ed.),

Greedy Algorithms, I-Tech Publisher, Vienna, Austria, Chapter 19, pp.359-368, 2008.

[9] M.Yoshikawa, H.Terai, Route Selection Algorithm based on Integer Ant Colony Optimization, Proc. of IEEE International Conference on Information Reuse and Integration, pp.17-21, 2008.

[10] M.Yoshikawa, H.Terai, "A Hybrid Ant Colony Optimization Technique for Job-Shop Scheduling Problems", Proc. of IEEE / ACIS International Conference on Software Engineering Research, Management & Applications, pp.95-100, 2006.

[11] H.M.Rais, Z.A.Othman, A.R.Hamdan, "Improved Dynamic Ant Colony System (DACS) on symmetric Traveling Salesman Problem (TSP)", Proc. of International Conference on Intelligent and Advanced Systems, pp.43-48, 2007.

[12] J.Ouyang, G.R.Yan, "A multi-group ant colony system algorithm for TSP", Proc. of International Conference on of Machine Learning and Cybernetics, pp.117-121. 2004.

[13] Chengming Qi, "An Ant Colony System Hybridized with Randomized Algorithm for TSP", Proc. of Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, vol.3, pp.461-465, 2007.

[14] Z.Cai, "Multi-Direction Searching Ant Colony Optimization for Traveling Salesman Problems", Proc. of International Conference on Computational Intelligence and Security, pp.220 -223, 2008.