ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.3, March 2007, pp.617−624
DOI: 10.1360/jos180617

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

并行蚁群算法中的自适应交流策略*

陈 崚[1,2+], 章春芳[1]

[1](扬州大学 计算机科学与工程系,江苏 扬州 225009)

[2](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

Adaptive Exchanging Strategies in Parallel Ant Colony Algorithm

CHEN Ling[1,2+], ZHANG Chun-Fang[1]

[1](Department of Computer Science, Yangzhou University, Yangzhou 225009, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

+ Corresponding author: Phn: +86-514-7978307, Fax: +86-514-7887937, E-mail: lchen@yzcn.net

Chen L, Zhang CF. Adaptive exchanging strategies in parallel ant colony algorithm. *Journal of Software*, 2007,18(3):617−624. http://www.jos.org.cn/1000-9825/18/617.htm

Abstract: Two strategies for information exchange between processors in parallel ant colony algorithm are presented. Theses strategies can make each processor choose other processors to communicate and to update the pheromone adaptively. A strategy is also presented to adjust the time interval of information exchange adaptively according to the distribution of the solutions so as to keep balance between the convergence speed and the diversity of the solutions. The adaptive parallel ant colony algorithm (APACA) based on these strategies adaptively updates the pheromone according to the equilibrium of the pheromone distribution in each information exchange so as to avoid the precocity and local convergence. These strategies are applied to the traveling salesman problem on the massive parallel processors (MPP) Dawn 2000. Experimental results show that the algorithm has higher convergence speed, speedup and efficiency than other parallel ant algorithms.

Key words: ant colony algorithm; parallel computation; adaptive strategy; information exchange

摘 要: 提出了并行蚁群算法中处理机间信息交流的两种策略,使得各处理机能够自适应地选择其他处理机以进行信息交换和相应信息素的全局更新.还提出了一种确定处理机之间进行信息交流的时间的策略,可以根据解的分布情况自适应地确定信息交流的时间,以取得全局收敛速度和解的多样性之间的平衡.在算法每一次信息交换后,采用自适应的更新策略,根据信息素的均匀度进行信息素的更新,从而避免了早熟和局部收敛.在 MPP 处理机曙光 2000 上对 TSP 问题的实验结果,表明了基于该自适应信息交换策略的并行蚁群算法比其他算法具有更好的收敛性、更高的加速比和效率.

关键词: 蚁群算法;并行计算;自适应策略;信息交流

中图法分类号: TP18 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60673060 (国家自然科学基金); the Chinese National Foundation for Science and Technology Development under Grant No.2003BA614A-14 (国家科技攻关项目); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2005047 (江苏省自然科学基金)

Received 2005-05-02; Accepted 2005-12-13

1 Introduction

Ant colony algorithm (ACA) is a meta-heuristic algorithm simulating the collective behavior of real ants. Inspired by the collective effect of ant colony, Dorigo, *et al*. first advanced an ant colony system and the ant colony algorithm (ACA)[1,2] to solve several discrete optimization problems. In ACA, artificial ants are created to emulate the real ants in the process of seeking food and exchanging information. The successful simulation has been applied to TSP problem[3], quadratic assignment problem[4], mesh-partition problem[5], protein folding problem[6], web page classification[7] and other combinational optimization problems[8−13]. ACA can usually converge to an almost optimal result in a certain number of iterations, but it becomes more difficult to speedup the algorithm when the complexity of the problem increases. To parallelize the ACA into a parallel ant colony algorithm, the structure of ACA must be modified to fit the parallel computational model. To obtain an efficient parallel ant algorithm, three key aspects must be considered: how to divide the single ant colony of sequential ACA into several mutually independent sub colonies so as to assign them into the multiprocessors, how to control and to manage the information exchange between the sub colonies, and when the information exchange between the sub-ant colonies should take place. Different methods of colony dividing and information exchanging produce different parallel ant colony algorithms.

Some results on parallel ant colony algorithms have been reported recently. Buklnheimer[14] proposed two synchronous and asynchronous parallelization strategies. In the synchronous strategy, every processor exchanges information after every iteration, while in the asynchronous strategy, every processor exchanges information after a certain time interval regularly. Talbi[15] presented a synchronous fine grained parallel ant colony algorithm in master/servant fashion combined with local tabu search, and applied this algorithm to solve quadratic assignment problem (QAP). Piriyakumar DAL[16] introduced an asynchronous parallel Max-Min ant colony algorithm associated with the local search strategy. Their algorithm was tested on the TSP using the parallel computer Cray-T3E. Randal M[17] introduced a synchronous parallel strategy which assigns only one ant on each processor. Dorigo M[18] advanced a parallel ant colony algorithm on the hyper-cube architecture by modifying the rule of updating the pheromone so as to limit the pheromone values within the range of [0,1]. This new approach enhances the ability of the ant colony algorithm to deal with complicated objective functions theoretically and practically. By modifying the classical ACA, Merkle D[19] proposed a parallel ant colony algorithm on reconfigurable processor arrays. The running time of the algorithm is quasi-linear with the problem size n and the number of ants on a reconfigurable mesh with n^2 processors.

The pattern and the time interval of information exchange between the processors are the most important factors to be considered in parallelizing the ant colony algorithm. These factors affect not only the speed of convergence, but also the optimization ability of the algorithm. In the algorithms of Ref.[14−17], the globally best solution is computed and broadcasted to all the processors in information exchange. Then every processor updates the pheromone matrix according to the global best solution. Besides the method of broadcasting the globally best solution, Shu-Chuan Chu[20] presented other six strategies for communication between ant colonies. All these methods of information exchange could not take the characteristic of each colony into account, and may probably create some similar solutions in different processors, which cause large amount of pheromone on some trails. These trails could cause local convergence, and will reduce the searching ability of the processors. In the algorithm of Ref.[21], each sub-colony updates the pheromone according to its neighbor's pheromone matrix. Since the logical neighbor of each sub-colony is determined by a randomly constructed virtual ring structure, and the weight of each sub-colony's pheromone is fixed, the algorithm does not consider the degree of the evolution of the sub-colonies. In the algorithms of Ref.[14−17,20,21], the processors exchange information in a constant time interval. Since this

constant time interval information exchange does not take the distribution of the solutions into account, it may influence the diversity of the solutions and the convergence speed. Although Ref.[14] acknowledged that this constant time interval of information exchange could affect the optimization speed, diversity and convergence of the algorithm, the detailed analysis of the effect and the method to reduce it have not been provided.

Due to the overhead caused by synchronous and communication, the parallel algorithms mentioned above are not efficient, their speed of convergence and performance could be improved. In this paper, two strategies for information exchange between processors in parallel ant colony algorithm are presented. Theses strategies can make each processor choose the partner to communicate and to update the pheromone adaptively. In order to increase the ability of optimization and to avoid early convergence, we also present a strategy to adjust the time interval of information exchange adaptively. The adaptive parallel ant colony algorithm (APACA) based on our strategies adaptively updates the pheromone according to the equilibrium of the pheromone distribution in each iteration so as to avoid the precocity and local convergence. These strategies are applied to the traveling salesman problem on the massive parallel processors (MPP) Dawn 2000. Experimental results show that the algorithm has higher convergence speed, speedup and efficiency than other parallel ant algorithms.

2 Adaptive Strategies for Choosing Processors to Exchange Information

Suppose P processors are used in APACA, the ants are divided into P sub-colonies. Each processor is assigned one sub-colony of ants which can make its local optimization independently in its own processor, and all the optimization processes of the sub-colonies can be carried out in parallel. After several generations of local optimization, the solutions of a sub-colony could become stagnant and probably no better solutions could be generated. To prevent such local convergence, the sub-colonies exchange information and update pheromone matrix by message passing between processors. Instead of choosing a neighboring or a random processor[22], we propose two strategies for information exchange which enable each processor to choose some other processors to exchange information adaptively according to their fitness or dissimilarities.

2.1 The strategy based on fitness

Let $f_{ave}(i)$ be the average fitness of solutions by processor i on current iteration. In every information exchange step, $f_{ave}(i)$ of all the processors are sorted in descent order. Suppose the indices of the processors after such sorting are $rank_1$, $rank_2$,…,$rank_P$. Let $q=\log_2 p$, the binary form of i be $(i_{q-1},i_{q-2},\ldots,i_0)_2$ and $i(l)=(i_{q-1},\ldots,\bar{i}_l,\ldots,i_0)_2$, for l=0,1,…,q−1, then the processor with index $rank_i$ ($i\in[1,2,\ldots,P]$) chooses the processors with indices $rank_{i(l)}$ (l=0,1,…,q−1) to exchange information.

In the information exchange, after the processor with index $rank_i$ ($i\in[1,2,\ldots,P]$) communicates with processors with indices $rank_{i(l)}$ for l=0,1,…,q−1, the elements $\tau(s,k)$ (s,k=1,2,…,n) of its pheromone matrix are updated as follows:

$$\tau_{rank_i}(s,k)=(1-\rho)\cdot\tau_{rank_i}(s,k)+\frac{\rho}{q^2}\sum_{l=0}^{q-1}(q-rank_{i(l)})\psi_{rank_{i(l)}}\tau_{rank_{i(l)}}(s,k) \tag{1}$$

where $\rho\in(0,1)$ is the evaporation coefficient of the pheromone in processor $rank_i$, and $\psi_{rank_{i(l)}}$ is defined as:

$$\psi_{rank_{i(l)}}=\begin{cases}1, & \text{The trail } (s,k) \text{ is included in the best tour of processor } rank_{i(l)}\\ 0, & \text{Otherwise}\end{cases} \tag{2}$$

This strategy enables the solutions of processors $rank_{i(l)}$ (l=0,1,…,q−1) to have different influence on the pheromone updating in processor $rank_i$. If the average fitness of a processor is relatively high, it will have greater influence on the pheromone updating. This will enable processor $rank_i$ to make full use of the information from the

processors with high average fitness so as to increase the pheromone on the best trail which is helpful to accelerate the speed of optimization in processor $rank_i$.

By adjusting the pheromone adaptively, the ants can avoid falling into the local optimum on the process of optimization. Meanwhile, the processor of low average fitness can improve their searching speed effectively and enhance the optimization ability by combining with the information coming from the processors of higher average fitness.

2.2 The strategy based on dissimilarity

We use $diss(i,j)$ to measure the dissimilarity of the pheromone in processor i and processor j:

$$diss(i,j)=\frac{2}{n(n-1)}\sum_{l=1}^{n}\sum_{k=l}^{n}\left|\tau_i(k,l)-\tau_j(k,l)\right| \tag{3}$$

In the strategy based on dissimilarity, each processor chooses the q processors with the most different pheromone, namely, processor i receives pheromone information from the q processors with the most dissimilarity. Suppose the indexes of those processors are $i_1,i_2,\ldots,i_q$, processor i updates the pheromone matrix with the information coming from those processors:

$$\tau_i(s,k)=(1-\rho)\cdot\tau_i(s,k)+\frac{\rho}{q^2}\sum_{l=1}^{q}(q-i)\psi_{i_l}\tau_{i_l}(s,k) \tag{4}$$

where $\rho\in(0,1)$ is also the evaporation coefficient of the pheromone in processor i, and $\psi_{rank_{i(l)}}$ is defined in Eq.(2).

From Eq.(4), we can see that the processors with larger dissimilarity will have greater influence on the pheromone updating. Since the strategy of information exchange based on dissimilarity enables each processor to choose its partner by their fitness and dissimilarity, each processor will exchange information with the processors which have the most dissimilarity and get the information of the highest fitness solution. Since the processors can update its pheromone with reference to the information on the trail of the best solutions of its partners, the processors can evolve towards the optimum solution and maintain the diversity of pheromone distribution to avoid local convergence and precocity.

3 Strategy of Adaptively Adjusting the Time Interval of Information Exchange

The purpose of information exchange is to propagate the information of the high quality solutions to other processors. When the optimization process of one processor tends to converge, it can get rid of local optimum solution by the information absorbed from other processors. If the pheromone of the processors lacks of diversity, the time interval of information exchange should be reduced so that the diversity can be increased and the searching ability can be enhanced by information exchange. If the processors have diversified pheromone, such time interval should be increased so that the overhead of communication can be reduced and each processor can continue searching in its own environment of evolution. To adjust the time interval of information exchange adaptively according to the diversity of pheromone, we denote the average and maximum dissimilarity of the pheromone in processors as *diss_ave* and *diss_max* respectively, i.e.

$$diss\_ave=\frac{2}{P(P-1)}\sum_{i=1}^{P}\sum_{j=i}^{P}diss(i,j),\quad diss\_\max=\max_{\substack{1\le i\le P\\ 1\le j\le P}}\{diss(i,j)\}.$$

The time interval of information exchange *period* is determined as follows:

$$period=\max\left\{k\cdot\frac{diss\_ave}{diss\_\max},\ 1\right\} \tag{5}$$

where k is a positive constant. Since *diss_ave* is the average dissimilarity of the pheromone in processors, $\frac{diss\_ave}{diss\_\max}$ reflects the global diversity in the processors. When its value increases, pheromone of the processors becomes well diversified, the time interval of information exchange can be increased appropriately in order to reduce the overhead of communication. When it becomes smaller, pheromone of the processors in the whole system lacks of diversity, so the time interval of information exchange should be reduced in order to interchange the information of the best solutions more frequently within the processors to prevent local convergence.

4 Experimental Results and Analysis

We test our parallel ant algorithm APACA based on the adaptive strategies mentioned above on the massive parallel processors Dawn 2000 using MPI (C bounding). The TSP benchmarks from the library TSPLIB[23] are used in the experiment. The parameters in the test are set as follows: ρ=0.4, α=1, β=2, k=16, the number of ants is equal to the number of cities, the number of processors used is 6. We perform 50 trials on each problem and 2000 iterations on each trial. Table 1 shows the experimental results on five TSP problems. In the table, Bit-PACA represents the algorithm of adapting the 4th communication strategy in Ref.[20], which enables the sub-colony i to choose the colony j differs by one bit. Fit-APACA stands for APACA which determines the time interval of information exchange based on fitness, while Dis-APACA is based on dissimilarity. The number in the name of a TPS problem is just the number of the cities in the problem, for instance, there are 318 cities in problem lin318.

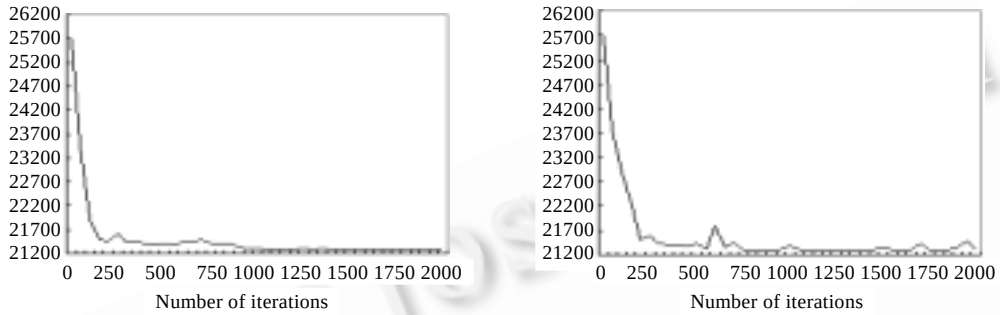
Table 1 Experimental results of Bit-PACA, Fit-APACA and Dis-APACA

Problem	Algorithm	Best value	Average value	The number of trials reaching the best solution	Time (s)
eil51	Bit-PACA	426.21	433.38	48	12.53
	Fit-APACA	426.21	426.21	50	10.07
	Dis-APACA	426.21	426.21	50	10.54
eil76	Bit-PACA	538.37	549.37	47	22.25
	Fit-APACA	538.37	539.98	49	15.51
	Dis-APACA	538.37	538.37	50	16.49
d198	Bit-PACA	15 793.92	15 937.62	42	28.29
	Fit-APACA	15 780.03	15 783.69	46	20.34
	Dis-APACA	15 780.03	15 781.48	48	22.01
lin318	Bit-PACA	42 036.72	42 108.25	43	44.71
	Fit-APACA	42 029.14	42 033.71	47	34.58
	Dis-APACA	42 029.14	42 032.48	47	36.54
att532	Bit-PACA	27 701.61	27 798.16	42	117.58
	Fit-APACA	27 686.45	27 717.75	45	104.39
	Dis-APACA	27 686.45	27 705.32	47	109.41

From Table 1, we can see that Fit-APACA and Dis-APACA have more trials getting the theoretical optimums and their average length of the shortest paths in the trails is much smaller than that of Bit-PACA. This indicates that our parallel algorithm has higher optimization ability and its computation time is reduced due to its high convergence speed. The reason for APACA's high optimization ability is that it can accelerate the convergence and avoid premature by determining the partners and the time interval of information exchange according to the diversity and the quality of the solutions. APACA uses two strategies to choose the communication partner adaptively, while in the algorithm of Bit-PACA all the sub-colonies could choose only the other sub-colony with the different codes to communicate. But since the sub-colonies are encoded randomly without any heuristic information, Bit-PACA has lower optimization ability than APACA.

Figure 1 shows the evolutionary process of the best solution in the test on kroA100 problem to compare the performance of our Dis-APACA with that of Bit-PACA. From Fig.1, we can see that the evolutionary process of our

algorithm converges after only 1200 iterations. After reaching the best solution, the curve is flat and constrained within a narrow scope around the best solution. But the curve of Bit-PACA fluctuates even after 1800 iterations. This shows our algorithm is more stable and effective than Bit-PACA. The high performance and efficiency of APACA is the result of its adaptive strategies of information exchange which enable the ant colonies to make full use of the information coming from the other processors to update the pheromone matrix and to guide the further search along the direction towards the best solution.



(a) Solutions obtained by "Dis-APACA" (b) Solutions obtained by "Bit-PACA"

Fig.1 The evolutionary process of the best solution for kroA100 problem

The results of APACA on the TSP problems using different numbers of processors are listed in Table 2. From the table we can see that when the number of processors is increased, the computing time can be reduced due to the fewer ants assigned on each processor. However, since the overhead of communication increases the total time of the algorithm, the speedup of our algorithm can not increase linearly with the increasing of the processors exactly. This is in conformity with the Amdahl's Law.

Table 2 Comparison of the results using different processors

Problem	Number of processors	Best value	Average value	Number of trials reaching the best solution	Time (s)
eil51	3	426.21	431.05	45	18.49
	6	426.21	426.21	50	10.54
	10	426.21	432.92	44	6.27
kroA100	3	21 282.44	21 335.81	37	31.52
	6	21 282.44	21 285.16	48	17.35
	10	21 282.44	21 305.91	41	10.17
lin318	3	42 029.14	42 092.73	35	68.25
	6	42 029.14	42 032.48	47	36.54
	10	42 029.14	42 048.51	40	22.41
att532	3	27 686.45	27 761.85	34	198.35
	6	27 686.45	27 705.32	47	109.41
	10	27 686.45	27 746.61	42	67.29

In the experiment, we also compare the results of APACA with that of the parallel ant algorithms with fixed time interval of information exchange. We test the problems using different time intervals of information exchange and the results are shown in Table 3. It can be seen in Table 3 that the quality of the solutions by APACA is higher than that of the algorithms with fixed time interval of information exchange. If fixed time interval is adopted, longer time interval requires less communication overhead and hence reduces the time cost. But on the other hand, since the information of the best solutions cannot be exchanged frequently, this could affect the search ability of the algorithm and the quality of the solution. Using the adaptive strategy to adjust the time interval of information exchange, APACA can keep balance between the computation time and the quality of the solution so as to get higher quality solutions.

Table 3 Comparison of the results by different time intervals of information exchange

Problem	Criterion of evaluation	The time interval of information exchange			
		4	6	8	Fit-APACA
eil51	Best value	427.34	429.75	434.48	426.21
	time (s)	9.53	9.39	9.26	10.07
kroA100	Best value	21 287.56	21 294.18	21 301.34	21 282.44
	time (s)	15.58	15.34	15.21	16.56
d198	Best value	15 791.81	15 802.17	15 814.92	15 780.03
	time (s)	19.51	19.39	19.28	20.34
pcb442	Best value	50 796.81	50 815.46	50 829.35	50 778.13
	time (s)	73.44	72.03	71.56	73.54

5 Conclusion

To improve the efficiency of the parallel ant colony algorithm, we present two strategies for information exchange between processors. Theses strategies can make each processor choose other processors to communicate and update the pheromone adaptively. We also present a strategy to adjust the time interval of information exchange adaptively according to the distribution of the solutions so as to keep balance between the convergence speed and the diversity of the solutions. Our adaptive parallel ant colony algorithm (APACA) based on these strategies adaptively updates the pheromone according to the equilibrium of the pheromone distribution in each information exchange so as to avoid the precocity and local convergence. Experimental results based on the traveling salesman problem on the massive parallel processors (MPP) Dawn 2000 confirm the high convergence speed, speedup and efficiency of algorithm APACA and the effectiveness of our adaptive strategies.

References:

[1] Dorigo M, Maniezzo V, Colomi A. The ant system: Optimization by a colony of cooperating agents. IEEE Trans. on Systems, Man, and Cybernetics (Part B), 1996,26(1):29−41.

[2] Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, 1997,1(1):53−66.

[3] Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem. BioSystems, 1997,43(2):73−81.

[4] Gambardella LM, Taillard E, Dorigo M. Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society, 1999,50:167−176.

[5] Korosec P, Silc J, Robic B. Solving the mesh-partitioning problem with an ant-colony algorithm. Parallel Computing, 2004,30: 785−801.

[6] Shmygelska A, Hoos HH. An improved ant colony optimization algorithm for the 2D HP protein folding problem. In: Yang X, Chaib-Draa B, eds. Proc. of the 16th Canadian Conf. on Artificial Intelligence. LNCS 2671, Springer-Verlag, 2003. 400−417.

[7] Holden N, Freitas AA. Web page classification with an ant colony algorithm. In: Yao X, *et al*., eds. Proc. of the 8th.Int'l Conf. on Parallel Problem Solving from Nature. Birmingham: Springer-Verlag, 2004. 18−22.

[8] Moss JD, Johnson CG. An ant colony algorithm for multiple sequence alignment in bioinformatics. In: Pearson DW, *et al*., eds. Artificial Neural Networks and Genetic Algorithms. Springer-Verlag, 2003. 182−186.

[9] Ando S, Iba H. Ant algorithm for construction of evolutionary tree. In: Langdon WB, ed. Proc. of the Genetic and Evolutionary Computation Conf. New York, 2002. 1552−1557.

[10] Maniezzo V, Carbonaro A. An ANTS heuristic for the frequency assignment problem. Future Generation Computer Systems, 2000, 16(6):927−935.

[11] Di Caro G, Dorigo M. AntNet: A mobile agents approach for adaptive routing. Technical Report, IRIDIA/97-12, Belgium: Universit Libre de Bruxelles, 1997.

[12] Schoonderwoerd R, Holland O, Bruten J. Ant-Like agents for load balancing in telecommunications networks. In: Johnson WL, *et al*., eds. Proc. of the 1st Int'l Conf. on Autonomous Agents. Marina del Rey: ACM Press, 1997. 209−216.

[13] Holland OE, Melhuish C. Stigmergy, Self-organization, and Sorting in Collective Robotics. Artificial Life 5, 1999. 173−202.

[14] Bullnheimer B, Kotsis G, Steauss C. Parallelization strategies for the ant system. High Performance and Algorithms and Software in Nonlinear Optimization, Applied Optimization, 1998,24:87−100.

[15] Talbi EG, Roux O, Fonlupt C, Robilard D. Parallel ant colonies for the quadratic assignment problem. Future Generation Computer Systems, 2001,17:441−449.

[16] Piriyakumar D, Levi P. A new approach to exploiting parallelism in ant colony algorithm. In: Stone P, ed. Proc. of the Int'l Symp. on Micromechatronics and Human Science. 2002. 237−243.

[17] Randall M, Lewis A. A parallel implementation of ant colony algorithm. Parallel and Distributed Computing, 2002,62:1421−1432.

[18] Blum C, Dorigo M. The hyper-cube framework for ant colony algorithm. IEEE Trans. on SMC, 2004,34(2):1161−1172.

[19] Merkle D, Middendorf M. Fast ant colony optimization on runtime Reconfigurable processor arrays. Genetic Programming and Evolvable Machine, 2002,3(4):345−361.

[20] Chu SC, Roddick JF, Pan JS. Ant colony system with communication strategies. Information Science, 2004,167:63−76.

[21] Tsai CF, Tsai CW, Tseng CC. A new hybrid heuristic approach for solving large traveling salesman problem. Information Science, 2004,166(1-4):67−81.

[22] Middendorf M, Reischle F, Schmeck F. Information exchange in multi colony ant algorithms. In: Rolim JDP, ed. Proc. of the Int'l Parallel and Distributed Processing Symp. Springer-Verlag, 2000. 645−652.

[23] TSPLIB WebPage. http://www.iwr.uni-heidelberg.de/groups/comopt/software/ TSPLIB95/tsp/

CHEN Ling was born in 1951. He is a professor and doctoral supervisor at the Department of Computer Science, Yangzhou University. His research areas are bioinformatics and parallel computation.

ZHANG Chun-Fang was born in 1982. She is a master candidate at the Department of Computer Science, Yangzhou University. Her research areas are optimization of algorithm and parallel computation.