July 2004, Vol.19, No.4, pp.472–478 J. Comput. Sci. & Technol.

Binary-Coding-Based Ant Colony Optimization and Its Convergence

Tian-Ming Bu[1], Song-Nian Yu[1], and Hui-Wei Guan[2]

[1]*School of Computer Engineering and Science, Shanghai University, Shanhai 200072, P.R. China*

[2]*Department of Computer Science, North Shore Community College, MA 01923, USA*

E-mail: tmbu@acm.shu.edu.cn; yusn@mail.shu.edu.cn; hguan@nscc.mass.edu

Received October 16, 2002; revised July 3, 2003.

Abstract Ant colony optimization (ACO for short) is a meta-heuristics for hard combinatorial optimization problems. It is a population-based approach that uses exploitation of positive feedback as well as greedy search. In this paper, genetic algorithm's (GA for short) ideas are introduced into ACO to present a new binary-coding based ant colony optimization. Compared with the typical ACO, the algorithm is intended to replace the problem's parameter-space with coding-space, which links ACO with GA so that the fruits of GA can be applied to ACO directly. Furthermore, it can not only solve general combinatorial optimization problems, but also other problems such as function optimization. Based on the algorithm, it is proved that if the pheromone remainder factor ρ is under the condition of $\rho \geqslant 1$, the algorithm can promise to converge at the optimal, whereas if $0 < \rho < 1$, it does not.

Keywords ant colony optimization, genetic algorithm, binary-coding, convergence, heuristic, function optimization

1 Introduction

Ant Colony Optimization[1,2] (ACO for short) was first proposed by Dorigo and his colleagues in the 90s last century as a brand-new approach to hard combinatorial optimization problems like traveling salesman problem (TSP). Taking inspiration from the behavior of real ant colonies, in particular, from their foraging behavior, the algorithm has become one of the most successful examples of swarm intelligence systems[3,4] now. Its main idea is the indirect communication among the individuals of a colony of agents, called *ants*, based on an analogy with trails of a chemical substance, called *pheromone*, which real ants use for communication. The pheromone trails are a kind of distributed numeric information which is modified by the ants to reflect their experience accumulated while they are solving a particular problem.

Although the meta-heuristics which was presented about only 10 years ago has taken on competitive performance in various applications, we have not yet recognized its essence clearly. Both the analysis of convergence and the design for operators and parameters are very immature. A great many constructions and improvements of ACO just rest on the experimental stage. There is still a wide gap between the theoretical researches and ACO's expansive practical applications.

Meanwhile, genetic algorithm[5,6] (GA for short) which originated like ACO from bionics, for about 40-year research, has already been developed completely in most fields. So, in this article, we introduce a new binary-coding-based ant colony optimization (BBACO for short), which connects ACO with GA as a bridge. Thus, by BBACO, a lot of significant results and advanced technologies of GA can even be applied directly to ACO. Furthermore, since the model is simple and consistent with ACO without modifying the essential parts, it is a well-suited tool to analyze ACO theoretically. In this paper, we present a mathematical convergence result based on the algorithm. In addition, the algorithm widens ACO's application scope to include function optimization.

The remaining part of this paper is organized as follows. In Section 2, we introduce ACO. Section 3 describes BBACO in detail. In Sections 4 and 5, we demonstrate the similarities and differences between ACO and GA. Section 6 gives the quantitative analysis of the algorithm's convergence. Finally, we indicate the directions for further research on the basis of our final considerations.

*Correspondence

This work is supported by the Science Foundation of Shanghai Municipal Commission of Science and Technology under Grant No.00JC14052.

2 Ant Colony Optimization

It is known that ant colonies are able to find the shortest path from a food source to their nest in their natural environment by relying on a biological mechanism[7,8]:

While going from the nest to the food source and vice versa, ants deposit pheromone on the ground. When arriving at a decision point, they make a probabilistic choice biased by the amount of pheromone they smell on all the branches. At the beginning of the experiment there is no pheromone on any branches and therefore ants going from the nest to the food source choose any one of the branches with equal probability. Due to different branch length, the ants choosing the shortest branch will be the first to reach the food source. At the decision point along their paths back to the nest, they will find some pheromone trail on the shorter path, which they released during the forward travel, and they will choose the path with higher probability than the longer ones. New pheromone will be released on the chosen path, making it even more attractive for the subsequent ants. While the process iterates, pheromone on the shorter path is deposited at a higher rate than on the longer ones, making the shorter path be selected again and again until all ants end up using it.

The collective behavior that emerges is a form of *autocatalytic* behavior. The process is thus characterized by a *positive feedback* loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path.

According to the above fact, in ACO, there are following corresponding analogies: artificial ants searching the solution space correspond to real ants searching their environment for food, the objective values are equivalent to the total length of the path between the nest and the food source.

In ACO, each individual ant constructs candidate solutions by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. At every point each ant has to decide which solution component to be added to its current partial solution according to a *probabilistic state transition rule*[9]. After the solution construction is completed, the ants give feedback to the solutions they have constructed by depositing pheromone on the solution components which they have used in their solution according to a *global pheromone updating rule*[9].

The probabilistic state transition rule used by an ant is given by (1), which gives the probability with which ant k in the vertex r chooses to move to the vertex s.

$$p_k(r,s)=\begin{cases}\dfrac{[\tau(r,s)]\cdot[\eta(r,s)]^{\alpha}}{\sum\limits_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^{\alpha}}, & \text{if } s\in J_k(r)\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

where τ is the pheromone trail, η is local heuristic information, and α determines the relative importance of τ to η. $J_k(r)$ is the set of vertices that remain to be visited by ant k positioned on vertex r (to make the solution feasible).

The global updating rule is implemented as follows:

$$\tau(r)\leftarrow\rho\cdot\tau(r)+\sum_{k=1}^{m}\Delta\tau_k(r) \tag{2}$$

where τ is the pheromone trail, $\Delta\tau_k(r)$ is the pheromone ant k deposits on the solution component r, m is the number of ants, and ρ is the pheromone remainder parameter.

3 Binary-Coding-Based Ant Colony Optimization

The binary-coding-based ant colony optimization uses *coding-space* instead of problem's *parameter-space*, and operates the colony on individual *bit-string* by selection and pheromone updating mechanism to establish an iteration process based on the evaluation of fitness function and evolution of coding colony. During the process, by reforming the bit-string stochastically, the new generation's bit-strings will be better than the previous and close to the optimal solution gradually.

The algorithm consists of five components: problem's parameter encoding, design of fitness function, selection, pheromone updating and control parameter setting.

The running of BBACO is typically a iterative process. The fundamental steps are as follows:

Step 1. Choose the encoding strategy, and convert the parameter set X into the bit-string space S.

Definition 1. *An l-bit-string s denotes that the length of s is l. e_{k0} denotes that the value of the k-th bit of s is 0 and e_{k1} the k-th bit of s is 1 contrarily.*

Step 2. Define the fitness function $f(s)$.

Definition 2. *If S denotes the bit-string space, the fitness function on S can be represented by $f(\cdot)$:*

$S \to \mathbb{R}^+$. In general, $f(\cdot)$ should be relative to the optimal problem $opt(s)$, namely, f is the function of $opt(s)$.

For example, if $opt(s)$ represents the length of the cycle in TSP, then f can have the following representations[9–11]:

(a) $f = \dfrac{1}{opt(s)}$

(b) $f = \begin{cases} \dfrac{1}{opt(s)}, & s \text{ belongs to the } \textit{global} \\ & \textit{best} \text{ or } \textit{local best} \text{ solution} \\ 0, & \text{else.} \end{cases}$

Step 3. Confirm the strategy of choosing the individual bit-string.

Virtually, the selection of an individual consists of the choice of the sequential bits. Thus, if every bit has its own information of pheromone trail and greedy heuristic, the probability of choosing bit-strings is the following product according to the foregoing probabilistic state transition rule:

$$P(s) = \prod_{k=1}^{l} \frac{\tau(e_{ks_k}) \cdot \eta^{\alpha}(e_{ks_k})}{\tau(e_{k0}) \cdot \eta^{\alpha}(e_{k0}) + \tau(e_{k1}) \cdot \eta^{\alpha}(e_{k1})} \quad (3)$$

where s_k denotes the state, 0 or 1, of the k-th bit of s.

Step 4. Confirm the strategy of pheromone trial updating.

According to the above definition, generally, we can adopt the following equation:

$$\tau(n, e_{ki}) = \rho \cdot \tau(n-1, e_{ki}) + \sum_{s \in A} f(n-1, s) \quad \rho > 0 \quad (4)$$

where n represents the n-th generation, ρ is a pheromone reminder factor and $A = \{x | e_{ki}$ belongs to the bit-string $x\}$.

Step 5. Set the parameters, including the proportion parameter α, the size N of colony (the number of ants), the initial pheromone c, the pheromone reminder parameter ρ and so on.

During the running, there exists a set of parameters which has a strong impact on the performance. In the phases of initialization and running, the parameters should be set and controlled reasonably to make the solution reach the best along the optimal track.

The larger the size of the colony is, the higher the diversity of the colony is, and the smaller the chance that the algorithm traps into the local optimal is. But the load of computing follows the increase of the colony's size. In some papers[9,12], the authors discussed the relation among the parameters in detail.

In the following section, it is also shown that if $0 < \rho < 1$, the solution generated by BBACO cannot guarantee the convergence to the optimal.

Step 6. Initialize the pheromone trial.

Step 7. Generate the next colony according to Step 3.

Step 8. Compute the fitness function $f(s)$ of each bit-string in the colony and update the pheromone trial.

Step 9. Update the global optimal solutions.

Step 10. Judge whether the performance of colony reaches the target or the algorithm reaches the maximum iteration $NC_{\max}$. If not, back to Step 6.

Setting of the maximum number of the iterations is used in most cases. Easy as the method is, it is not precise. So the running should be controlled by the degree of colony's diversity, namely the degree of convergence of the colony. And the running can also be stopped by the variability of the best solution getting from the beginning.

4 Computational Instance

We will employ a representative function optimization[13] $\max f(x) = x \cdot \sin(10\pi \cdot x) + 2.0$, $x \in [-1, 2]$ as an example to introduce BBACO further, including its idea and computational process. And we can understand the similarity and difference between ACO and GA by sensibility.

It is easy to know that in the function there are 15 peak values within the interval and the maximum value is a bit bigger than 3.85.

We are now in a position to solve the function optimization problem by BBACO:

1) Encoding: we use the binary-coding strategy. The length $l = 22$ and the conversion function from a bit-string to a special parameter is:

$$S = \Gamma(e_1, e_2, \ldots, e_{21}, e_{22}) = (-1) + \frac{2-(-1)}{2^{22}-1} \cdot \Big(\sum_{i=1}^{22} e_i \cdot 2^{22-i}\Big).$$

2) Fitness value: we just use the optimizing function $f(x)$ as the fitness function.

3) We adopt directly (3) and (4) as selection and pheromone updating strategy respectively, and let $\alpha = 0$ and $\rho = 1$. It means that we do not consider the local heuristic information on each bit. And there exists no evaporation phenomenon so that all the pheromone will be preserved completely.

4) Let the size of colony $N = 30$, the initial pheromone c $= 3$, and the maximum iteration $NC_{\max} = 150$.

5) In the initial colony, because the pheromone on the bits is equal to each other, it is obvious that every bit-string's selected probability is equal to $(1/2)^{22}$. And we might as well suppose that bit-string r's fitness value is the best in the initial colony.

6) And then update the pheromone on the bits according to (4). After updating, we can see that every bit-string's selected probability will not be alike and the probability of choosing the bit-string r will be higher than that of choosing its complementary $\bar{r}$ or any others.

7) Steps 5) and 6) will work alternately until the iteration is beyond the upper bound ($NC_{\max}$).

The real line in the three subsequent diagrams is just the average performance of BBACO which runs 100 times separately (There are 150 iterations in each trial). Figs.1–3 character its performance from various angles.

Thus it can be seen that by selection and pheromone trail updating, both the whole colony's fitness value average and best individual's fitness value are improved. While the number of the iterations increases, the colony evolves gradually to close the best solution.

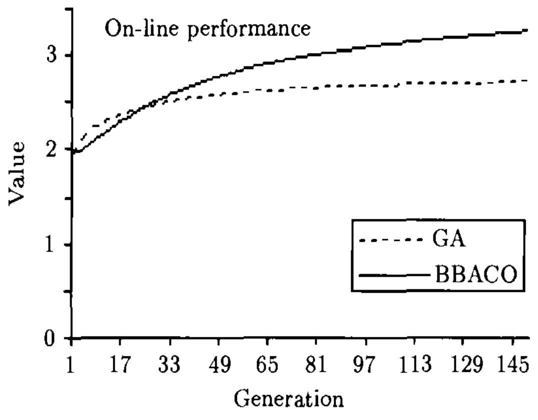

Fig.1. $P_{on\_line} = \frac{1}{T}\sum_{t=1}^{T} f(t)$, where $f(t)$ expresses the fitness value average under the t-th generation. The on-line performance function describes the whole colony's capability of evolution and the algorithm's dynamic behavior[14].

Meanwhile, we give the same problem's computational result implemented by GA[13] in which the length of chromosome is 22, the population is 30, the maximum generation is 150—all of the three are consistent totally with the corresponding setting in BBACO—the crossover probability is 0.8, the mutation probability is 0.05 and we use the fitness-proportionate selection, one-point crossover and elite preservation strategy. It is evident from the figures above that BBACO is better than GA.

Of course, these two versions are the most fundamental and "pure" without any other addition. By adding other optimal methods such as local search, each one may defeat the other; however, the topic has gone beyond this paper's discussion.

By the example, it can be found that BBACO shows its simple, robust, powerful searching ability and can run under the parallel environment.

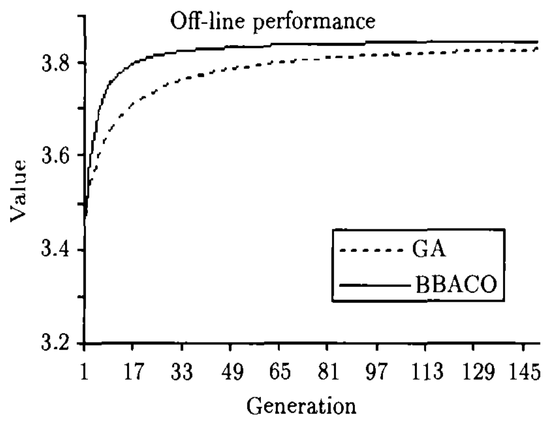

Fig.2. $P_{off\_line} = \frac{1}{T}\sum_{t=1}^{T} f^*(t)$, where $f^*(t)$ is the best fitness value found from the beginning to the t-th generation. The off-line performance function reflects the individual's capability of evolution and the algorithm's capability of search[14].

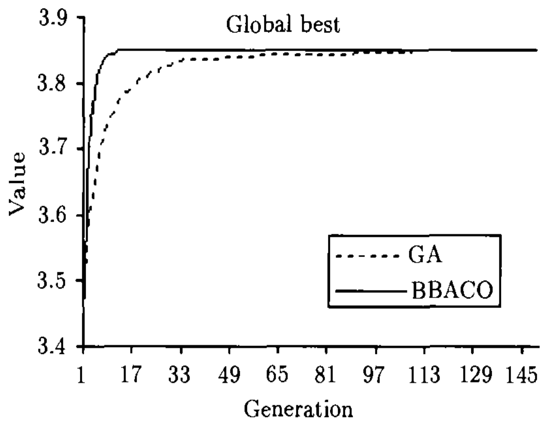

Fig.3. $P_{best} = f^*(t)$. This function can reflect the algorithm's speed of convergence and judge whether it will be premature or not.

5 ACO vs. GA

As a variation of ACO, BBACO is mainly characteristic of the conversion from a problem's parameter-space into coding-space—it is one of the key ideas in GA—instead of graph as usual. And it keeps ACO's two original and essential parts: probabilistic state transition rule and global pheromone updating rule. As a result, ACO and GA, which have totally different origin, are linked up.

By the concrete example in the previous section, we can find that:

The first step in GA generally is to determine the encoding strategy such as binary coding, tree coding, self-adaptive coding and to build the corresponding searching space. Similarly, ACO also needs the step.

Both ACO and GA have fitness function to evaluate every individual's fitness value. They will improve their offspring fitness value by some means in order to approach the optimal solution.

GA is influenced by Darwinian Theory to produce offspring stochastically by reforming (crossover), mutation and selection. It regards reforming as a leading actor and mutation a background role. However, it is worth noticing that either Evolutionary Planning (EP) or Evolutionary Strategy (ES) which is similar to GA usually does not use reforming[15]. Therefore the number of individuals in each of their generation need not be larger than 1.

Although ACO also produces offspring by reforming, its idea is inspired by ants' highly structural society upon which pheromone takes a leading effect. ACO's reforming is based on the intensity of the pheromone deposited by the historical ants. Consequently, its reforming is not based on just previous generation, but all generations. Likewise, its selection just aims to make sure which ants can represent the generation to excrete the pheromone to influence the next generation.

Table 1 summarizes the above discussion.

Hence, to be understood at a higher level, ACO can be considered as a new branch of Evolutionary Computing (EC). Moreover, when researching on ACO, we can absorb GA's vast advanced results, such as adopting mutation operator, adding parameters' self-adaptive mechanism and so on.

6 Convergence

Definition 3. *$p(n)$ denotes the probability that state 0 of the k-th bit in a bit-string was visited by every ant from the beginning to the n-th iteration. In other words, $p(n)$ represents the probability that state 1 of the bit was never visited by any ant from the beginning to the n-th iteration.*

Lemma. *In BBACO, if $0 < \rho < 1$, $0 < \lim_{n\to\infty} p(n) \leqslant 1$. If $\rho \geqslant 1$, $\lim_{n\to\infty} p(n) = 0$.*

Proof. According to the definition and the state transition rule, we know that:

$$p(n) = \prod_{i=1}^{n}\left[\frac{\tau(i-1,e_{k0})\cdot\eta^{\alpha}(e_{k0})}{\tau(i-1,e_{k0})\cdot\eta^{\alpha}(e_{k0})+\tau(i-1,e_{k1})\cdot\eta^{\alpha}(e_{k1})}\right]^{N} = \prod_{i=1}^{n}\left[1-\frac{\tau(i-1,e_{k1})\cdot\eta^{\alpha}(e_{k1})}{\tau(i-1,e_{k0})\cdot\eta^{\alpha}(e_{k0})+\tau(i-1,e_{k1})\cdot\eta^{\alpha}(e_{k1})}\right]^{N}.$$

Table 1. ACO vs. GA

	ACO	GA
Encoding	Yes	Yes
Fitness function	Yes	Yes
Size of colony (population)	Any	> 1, but unnecessary in EP or ES
Reforming	Based on the pheromone deposited by the historical ants	Based on the previous generation
Mutation	No	Yes
Selection	Based on the union of the previous generation and its reforming	Based on the union of the previous generation, its reforming and mutation

Let

$$A_i = \frac{\tau(i-1,e_{k1})\cdot\eta^{\alpha}(e_{k1})}{\tau(i-1,e_{k0})\cdot\eta^{\alpha}(e_{k0})+\tau(i-1,e_{k1})\cdot\eta^{\alpha}(e_{k1})}$$

then

$$p(n) = \prod_{i=1}^{n}[1-A_i]^N = \left\{\prod_{i=1}^{n}[1-A_i]\right\}^N$$

Since N is finite, the convergence of $\lim_{n\to\infty} p(n)$ is consistent with $\prod_{i=1}^{\infty}[1-A_i]$. And $\forall i$, $0 < A_i < 1$, the convergence of $\prod_{i=1}^{\infty}[1-A_i]$ is consistent with $\sum_{i=1}^{\infty} A_i$. The reason is that if $\sum_{i=1}^{\infty} A_i$ is convergent, then $\lim_{i\to\infty} A_i = 0$. Since $\lim_{i\to\infty}\dfrac{\ln(1-A_i)}{A_i} = 1$, $\sum_{i=1}^{\infty}\ln(1-A_i)$ is convergent and $\prod_{i=1}^{\infty}[1-A_i] = e^{\sum_{i=1}^{\infty}\ln(1-A_i)}$. Therefore $\prod_{i=1}^{\infty}[1-A_i]$ is convergent. Hence, the convergence of $\lim_{n\to\infty} p(n)$ and $\sum_{i=1}^{\infty} A_i$ is identical.

According to the pheromone updating Rule (4), we also know that:

$$\begin{cases} \tau(0,e_{k0}) = c, \\ \tau(0,e_{k1}) = c. \end{cases}$$

$$\begin{cases} \tau(n,e_{k0}) = \rho\cdot\tau(n-1,e_{k0}) + \sum_{s\in A} f(n-1,s) \\ \qquad = \rho^n\cdot c + \rho^{n-1}\cdot\sum_{s\in A} f(1,s) + \cdots + \\ \qquad \sum_{s\in A} f(n-1,s), \\ \tau(n,e_{k1}) = \rho^n\cdot\tau(0,e_{k1}) = \rho^n\cdot c. \end{cases}$$

Therefore, if let

$$B_n(const) = \rho^{n-1}\cdot c\cdot\eta^{\alpha}(e_{k1})/\{[\rho^{n-1}\cdot c + \rho^{n-2}\cdot const + \cdots + const]\cdot\eta^{\alpha}(e_{k1}) + \rho^{n-1}\cdot c\cdot\eta^{\alpha}(e_{k1})\}$$

then

$$B_n(max) \leqslant A_n \leqslant B_n(min)$$

where *min* denotes the lower bound of $\{\sum_{s\in A} f(1,s),\ldots,\sum_{s\in A} f(n-1,s)\}$, and *max* denotes the upper bound of it.

According to the integral test for the convergence of infinite series, it can be deduced that:

$$\begin{cases} \sum_{i=1}^{\infty} B_i(const) \text{ is convergent}, & 0<\rho<1; \\ \sum_{i=1}^{\infty} B_i(const) \text{ is divergent}, & \rho\geqslant 1. \end{cases}$$

Since $A_n \leqslant B_n(min)$, $\sum_{i=1}^{\infty} A_i$ is convergent when $0<\rho<1$. For $A_n \geqslant B_n(max)$, $\sum_{i=1}^{\infty} A_i$ is divergent when $\rho\geqslant 1$. Because the convergence of $\lim_{n\to\infty} p(n)$ and $\sum_{i=1}^{\infty} A_i$ is identical, it is easy to know that if $0<\rho<1$, $\lim_{n\to\infty} p(n)$ is convergent and $0 < \lim_{n\to\infty} p(n) \leqslant 1$. By the same reason, if $\rho\geqslant 1$, $\lim_{n\to\infty} p(n)$ is divergent. Due to $\lim_{n\to\infty} p(n) \leqslant 1$, its value is 0. □

Theorem. *In BBACO, if we keep the best solution generated from the beginning and $0<\rho<1$, the algorithm cannot promise to converge to the global optimal solution. If $\rho\geqslant 1$, the algorithm can do.*

Proof. From Definition 3, we know that $p(n)$ represents the probability that state 1 of the k-th bit was never visited by any ant from the beginning to the n-th iteration. According to the lemma above, if $0<\rho<1$, $0<\lim_{n\to\infty} p(n) \leqslant 1$. It means that there exist some bit-strings which may belong to the global optimal solutions in which the state of the k-th bit is 1, but will never be visited. So, the algorithm cannot converge to the best with guarantee.

If $\rho\geqslant 1$, $\lim_{n\to\infty} p(n) = 0$. That means, when $n\to\infty$, each bit-string will be visited with probability 1. Thus, if we keep the best solution from the beginning, the algorithm must converge to the optimal solution. □

7 Summary and Conclusion

The paper's main contributions are: (i) present a new binary-coding based ant colony optimization; (ii) combine the traditional ant colony optimization with genetic algorithms so that we can regard ACO as a variation of EC; (iii) analyze the convergence of the algorithm.

The analysis of the heuristic's convergence is theoretically valuable in certain cases. Other algorithms such as genetic algorithm, simulate annealing and so on can converge to the global optimal only if the iteration or the time goes infinitely. In other words, the time complexity of reaching the best solution is far bigger than that of enumerating all the solutions. But by the analysis, we can deepen the understanding of ACO, and realize the relationship among parameters and their roles in the algorithm.

At last, there are several issues which seem to be worth further investigating:

1) How to evaluate the quality of ACO and other heuristics? Some users are concerned about heuristics' speed and others care their results. So we should build a set of criterion of heuristics' performance and let users select.

2) In which areas can ACO behave better than other heuristics? Namely, we should find much more fields to which ACO can be applied much better.

3) ACO's stop criterion. Simply using the maximum iteration as the rule of stop is blind and unscientific. We should consider the algorithm's performance during the running to judge dynamically whether to stop or not.

4) ACO's convergent speed, or the efficiency of the algorithm, or the relationship among its param-

eters. The three are identical because the aim of investigating the relationship among parameters is just to find out a way to speed up the algorithm's convergence and improve its efficiency.

References

[1] Dorigo M, Caro G D. The Ant Colony Optimization Meta-Heuristic. In *New Ideas in Optimization*, Corne D, Dorigo M, Glover F (eds.), McGraw-Hill, 1999, pp.11–32.

[2] Dorigo M, Caro G D, Gambardella L M. Ant algorithms for discrete optimization. *Artificial Life*, 1999, 5(2): 137–172.

[3] Bonabeau E, Dorigo M, Theraulaz G. Swarm Intelligence: From Natural to Artificial Systems. Oxford Univ. Press, New York, 1999.

[4] Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behavior. *Nature*, 2000, 406(6): 39–42.

[5] Li M-Q, Kou J-S, Lin D, Li S-Q. Genetic Algorithm's Foundational Theory and Its Application. Science Press, Beijing, 2002.

[6] Liu Y, Kang L-S, Chen Y-P. Non-Numerical Parallel Algorithm: Genetic Algorithm. Science Press, Beijing, 1995.

[7] Deneubourg J-L, Aron S, Goss S, Pasteels J-M. The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 1990, 3: 159–168.

[8] Goss S, Aron S, Deneubourg J-L, Pasteels J-M. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 1989, 76: 579–581.

[9] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.*, 1997, 1(1): 53–66.

[10] Dorigo M, Maniezzo V. The ant system: Optimization by a colony of cooperating agents. *IEEE Trans. SMC*, 1996, 26(1): 29–41.

[11] Stutzle T, Hoos H H. MAX-MIN ant system and local search for combinatorial optimization problems. In *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Voss S, Martello S, Osman I H, Roucairol C (eds.), Kluwer, Boston, 1999, pp.313–329.

[12] Bu T-M, Yu S-N. Ant algorithm for the maximum clique problem. In *Fourth Shanghai Conference on Combinatorics*, Shanghai Jiaotong University, May 002.

[13] Wang X-P, Cao L-M. Genetic Algorithm—Theory, Application and Implement. Xi'an Jiaotong University Press, Xi'an, 2002.

[14] De Jong K A. An analysis of the behavior of a class of genetic adaptive system. University of Michigan, No.76–9381, 1975.

[15] Wang Zh-Zh, Bo T. Evolutionary Computing. National University of Defence Technology Press, Changsha, 2000.

Tian-Ming Bu received the M.S. degree in computer software and theory from Shanghai University, China, in 2003. And now he is a Ph.D. candidate of Fudan University in the same area of theory computer science. His research interests include algorithms, especially, heuristic algorithms and parallel algorithms, quantum computing and computational complexity.

Song-Nian Yu received the B.S. degree in mathematics from Xi'an University of Science and Technology, Xi'an, China, in 1981, the Ph.D. degree under Prof. L. Lovasz's guidance and from Lorand University, Budapest, Hungary, in 1990. Dr. Yu is a professor in the School of Computer Engineering and Science at Shanghai University. He was a visiting professor as a faculty member in Department of Computer Science at Nelson College of Engineering, West Virginia University, from 1998 to 1999. His current research interests include parallel algorithms' design and analyses, graph theory, combinatorial optimization, wavelet analyses, and grid computing.

Hui-Wei Guan received the B.S. degree in electronic engineering from Shanghai University, China, in 1982, the M.S. degree in computer engineering from China Textile University, China, in 1989, and the Ph.D. degree in computer science and engineering from Shanghai Jiaotong University, China, in 1993. He is an associate professor in the Department of Computer Science at North Shore Community College, USA. He is a member of IEEE. His current research interests are parallel and distributed computing, high performance computing, distributed database, massively parallel processing system, and intelligent control.