Study of Pseudo-Parallel Genetic Algorithm with Ant Colony Optimization to Solve the TSP

Sheng Li†, Huiqin Chen and Zheng Tang††

Graduate School of Innovative Life Science, University of Toyama, Toyama-shi, Japan

Summary
The traveling salesman problem (TSP) has attracted many researchers' attention in the past few decades, and amounts of algorithms based on heuristic algorithms, genetic algorithms, particle swarm optimization, tabu search and memetic algorithms have been presented to solve it, respectively. Unfortunately, their results have not been satisfied at all yet. This paper is devoted to the presentation of a novel hybrid pseudo-parallel genetic algorithm with ant colony optimization (PPGA-ACO). The experimental results on small and large size TSP instances in TSPLIB (traveling salesman problem library) show that PPGA-ACO is more robust and efficient than the traditional algorithms.

Key words:
Traveling salesman, genetic algorithm, pseudo-parallel, ant colony optimization, hybridization.

1. Introduction

The traveling salesman problem (TSP) is one of the existing combinatorial optimization problems and it has been demonstrated to be an NP-hard problem [1,2]. Given a set of cities and the distances between them, TSP is to find a complete, minimal-cost tour visiting each city once. The TSP is a well-known combinational optimization problem with many real-world applications, such as job shop scheduling and VLSI routing [3]. The TSP has often served as a touchstone for new problem-solving strategies and algorithms; and many well-known combinatorial algorithms were first developed for the TSP. In this paper, we consider the symmetric TSP, where the distance from a city to another is the same as the distance in the opposite direction.

A large number of methods have been developed for solving TSP. The complexity of exact algorithms is often exponential. In order to tackle larger TSP instances effectively and decrease the computational cost, it is necessary to develop approximate algorithms that do not always aim at finding optimal solutions but at finding quasi-optimal solutions in an acceptable running time. An excellent survey on approximate algorithms for the TSP is provided in the reference [4]. These methods can be roughly divided into local search and global search approaches. In general, the local search approaches, such as 2-opt, 3-opt and Lin-Kernigan [4] are efficient and fast convergence, because the selection of reconnecting cities depends on geometric neighborhood information and the edges from other individuals in the population. Nevertheless, they might get struck at local minima because they do not deal with the diversity of feasible solutions.

Genetic Algorithm [5] is a global search algorithm appropriate for problems with huge search spaces such as the TSP, in which the crossover realizes the construction of the offspring and the mutation operator maintains the diversity of the individuals. Many improved GAs have been applied in TSP. One of these variant algorithms is the pseudo-parallel genetic algorithm (PPGA) [6], which introduce multi-population evolution thoughts of parallel genetic algorithm, and can run on personal PC. In order to realize the information exchange during each individual GA, a method should be assigned. In this paper, we utilize the ant colony optimization (ACO) [7] to achieve this main purpose, and further improve the local search ability of GA. Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. The more prefer for paths with a high pheromone level, the higher rate of growth of the amount of pheromone on shorter paths. The solution ability of ACO precedes some local search algorithm, such as 2-opt, 3-opt and so on [8]. By doing so, a hybrid algorithm called pseudo-parallel genetic algorithm with ant colony optimization (PPGA-ACO) is established to develop an effective method capable of finding high-quality solution for the problem in hand.

The basic idea of PPGA-ACO can be summarized as that, basically there are several genetic algorithm in a parallel manner to solve the problem, and these genetic algorithms are independent to each other from the view of implementation. Each genetic algorithm maintains a sub-population and runs a simple GA with crossover and mutation operators. Then the ant colony optimization is incorporated into the pseudo-parallel genetic algorithm, not only to realize the information exchange during all individual genetic algorithms, thus enhance the global search ability and improve the diversity of the whole population, but also to act as a local search mechanism to further improve the searching performance of the

Manuscript received March 5, 2011
Manuscript revised March 20, 2011

proposed algorithm. Experimental results based on several benchmark instances taken from the TSPLIB verify the effectiveness of the proposed PPGA-ACO when compared it to other traditional algorithms.

The remainder of this paper is organized as follows: a brief introduction of the genetic algorithm and pseudo-parallel genetic algorithm is given in the following section. Section 3 provides the details of the proposed PPGA-ACO by applying it on the STSP. Section 4 discusses the experimental results. Finally, some remarks and conclusions are summarized.

2. Genetic Algorithm

A Genetic Algorithm (GA) is a meta-heuristic inspired by the efficiency of natural selection in biological evolution. Genetic Algorithms (GAs) have been applied successfully to a wide variety of combinatorial optimization problems and are the subject of numerous recent books [9-13] and conference proceedings [14-16].

2.1 Basic description of GA

The basic ideas behind GAs evolved in the mind of John Holland at the University of Michigan in the early 1970s [17]. GAs were not originally intended for highly constrained optimization problems but were soon adapted to order-based problems like the TSP. The development of effective GA operators for TSPs led to a great deal of interest and research to improve the performance of GAs for this type of problem. Several summaries of solving TSPs with GAs have been published that provide comprehensive reviews of the operators and associated issues [18, 19].

In general, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are then evaluated according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem. The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes or until no significant improvement occurs in the population for a given number of iterations.

From the view of optimization, GA maintains a large number of solutions and performs comparatively little work on each one. The collection of solutions currently under consideration is called the population. Each member of the population (called an individual or a chromosome) is an encoded version of a solution. Each iteration of a GA consists of several operators that construct a new generation of solutions from the old one in a manner designed to preserve the genetic material of the better solutions (survival of the fittest). Many GA operators have been proposed; the three most common are reproduction, crossover, and mutation. Reproduction consists of simply copying the best solutions from the previous generation into the next, with the intention of preserving very high-quality solutions in the population as-is. Crossover takes two parents, randomly chosen, and produces one or more offspring that contain some combination of genes from the parents. Crossover can be performed in a deterministic manner (e.g., one point crossover), with genes appearing before a certain cutoff coming from parent 1 and genes after the cutoff coming from parent 2, or in a random manner, with each gene taken from a given parent with a certain probability. The mutation operator changes a few genes randomly, borrowing from the evolutionary concept that random genetic mutations may produce superior offspring (or, of course, inferior offspring, but such individuals are less likely to survive from one generation to the next). The general algorithm scheme of GA can be illustrated in the following.

Algorithm scheme for Basic Genetic Algorithm

```
1.  Begin
2.  Initialize the population P(0)
3.  Set generation number t=0
4.  While (t<= T) do
5.      For i=1 to M do
6.              Evaluate fitness of P(t)
7.              Selection operation to P(t)
8.              Crossover operation to P(t)
9.              Mutation operation to P(t)
10.     End For
11.     For i=1 to M do
12.             P(t+1)=P(t)
13.     End For
14. End While
15. End
```

2.2 Pseudo-parallel GA

In the application of GA, an obvious problem is the premature convergence that affects the result of GA. At the same time, we introduce pseudo-parallel genetic algorithm, which has the capacity to maintain the population diversity as well as enhance the running speed of GA. Thus the premature convergence may be restrained, but parallel GA requires the system running on parallel computer or local area network, while such high

performance running environment is not necessary for many practical applications. Therefore, Zhou Ming and Sun Shudong [20] proposed a pseudo-parallel genetic algorithm.

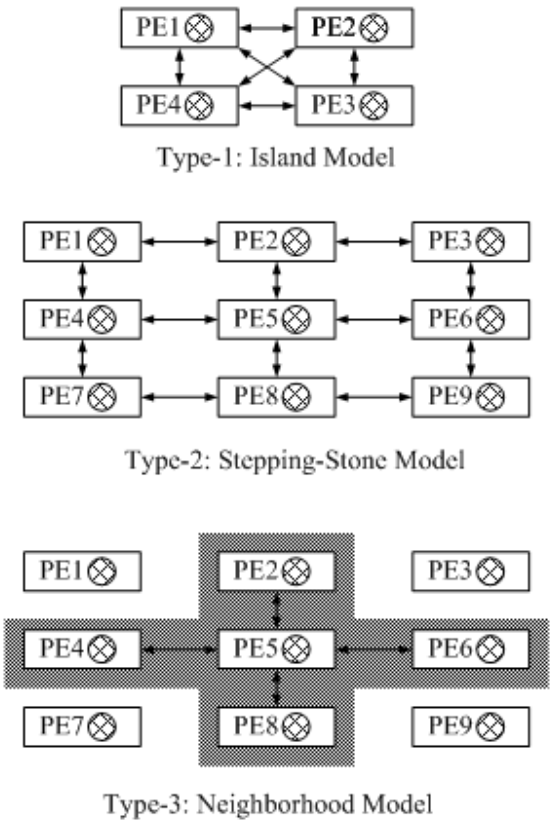

Fig. 1: Three Types of the information exchange models in PPGA.

In the PPGA, the population is divided into some subpopulations. Each sub-population evolves independently in certain pattern, and some sub-populations exchanges information at proper time. Thus the diversity of populations is maintained and the premature convergence is constrained. With these sub-populations executive serially on single processor rather than evolve independently on different processors in the algorithm, it is called pseudo-parallel genetic algorithm.

In the PPGA, the fitness of each sub-population is calculated, and then do the selection, crossover and mutation operations in each sub-population, finally, exchange information among populations according to the model of information exchange. The model of information exchange used at present are Island Model [21], Steppingstone Model [22] and Neighborhood Model [23], see Fig. 1 for illustration. The main differences of these models are the scale of sub-population and the method of information exchange. However, in the scale, they have some common points: the scales of subpopulations are same and invariable, i.e. the number of individuals in every sub-population and it does not change in the course of evolution.

3. Proposed PPGA-ACO

In this paper, we proposed a pseudo-parallel genetic algorithm with ant colony optimization (PPGA-ACO) to solve the TSP. As mentioned above, the aim of the embed ant colony optimization (ACO) is to realize the information exchange during different GAs. Although there are three types of the information exchange models in the reference, we adopt the island model because it is the simplest and the most used one. In this model, each GA communicates and cooperates with each other, and the ants in ACO play effect to accomplish this task. To make this paper self-explanatory, before actually proposing the hybrid PPGA-ACO, the principles, basic components and characteristics of ant colony optimization are briefly explained.

3.1 Ant colony optimization

Ant system (AS) comprises a set of cooperating agents called ants, which utilize an indirect form of communication mediated by a pheromone. The ant system has been inspired by the collective behavior of real ant colonies, and in particular, by their foraging behavior. The primary idea of the ant algorithm is the indirect communication among ants in the colony agents, based on pheromone trails. The pheromone trails constitute a type of distributed numerical information, which is altered by the ants to reflect the experiences gained when solving a particular problem. Ant system is known as one of the most efficient algorithm for TSP. Inspired by the collective behavior of ant colony, Dorigo developed the ant system (AS) [24], and later continue to design this system [25]. To demonstrate the AS method, Dorigo apply this approach to the jobshop scheduling problem, classical traveling salesman problem (TSP), and quadratic assignment problem (QAP). The ant system shows very excellent results in each applied area. More recently Dorigo has been designing expanded versions of the AS paradigm. The AS is one such extension and has been applied to the symmetric and asymmetric TSP with excellent results [26]. The AS has also been successfully applied to other combinatorial optimization problems, including the telecommunications networks, partitioning, scheduling, coloring, and vehicle routing problem [27]. Recently, the ant colony optimization (ACO) meta-heuristic has been proposed, representing a unifying framework to support most applications of ant algorithms to combinatorial optimization problems. All the ant

algorithms applied to the TSP fit perfectly into the ACO meta-heuristic; therefore, these algorithms are hereinafter also called ACO algorithms [28].

The steps of the ACO algorithm are illustrated as follows:

Algorithm scheme for Ant Colony Optimization

1. Begin
2. Initialize the pheromone table
3. Randomly allocate ants to every node
4. Every ant must walk to next city, depending on the probability distribution given in Eq. (2) (local search)
5. Compute the length of the path traveled by each ant, and allocate a quantity amount of pheromone to the path, according to the length of its path
6. Perform a local update
7. Compute whether a better solution is obtained in this time step than the last; if so, then perform a global update on the solution and empty the Tabu value; repeat Steps 3 to 7
8. End

3.2 Flowchart of PPGA-ACO

PPGA-ACO mimics the process of parallel computation. It designates several sub-populations which respectively evolve as the classic GA does and synchronizes them at proper time. For every sub-population, its evolvement process is controlled by genetic operators and at proper time, the chromosomes of the two sub-populations are moved between each other, which makes the fitness of the whole population a wholly improvement and accelerate the speed of convergence.

The flowchart of the proposed PPGA-ACO is illustrated in the following. It should be noted that although the information exchange model is selected to be the island model, the elite pool estimation principle is also employed. That is to say, after exchange and evaluation, the fittest ant will be record as memory and used as the global optimal solution for further purposes.

Algorithm scheme for PPGA-ACO

1. Begin
2. Set the generation number of PPGA: t=0
3. Randomly generate the initial population P(t), and divide it into several sub-populations according to Island model: $P(t)=\{P_1(t), P_2(t), \ldots, P_i(t), \ldots, P_M(t)\}$
4. Compute the fitness for each sub-population $P_i(t)$
5. Evolve individuals in each sub-population $P_i(t)$:
   5.1 Selection Operator: $P_i'(t) = selection[P_i(t)]$
   5.2 Crossover Operator: $P_i'(t) = crossover[P_i(t)]$
   5.3 Mutation Operator: $P_i'(t) = mutation[P_i(t)]$
6. Evaluate the fitness for newly sub-population $P_i'(t)$
7. Place the ants of ACO into each sub-population, implement the ACO algorithm; the information exchange will be accomplished through the pheromone on the TSP map.
8. If the termination conditions are fulfilled, output the fittest solution; otherwise set t=t+1, go to Step 5
9. End

The details in PPGA-ACO are explained as follows. As for the single sub-population initialization, in order to improve the performance of computation, a new population initialization method-intelligent population initialization is proposed. With this new method, a created chromosome will be checked whether it violates the constraints or not. If it does, it will be discarded and simultaneously another one, which conforms to the constraints, is created and replaces it. After initialization, the initial population contains no chromosome that violates the constraints. This would be very helpful in reducing computation burden.

As the operators concerned, the conventional two-generation competitive selection method was adopted. Parent generation chromosomes and offspring generation chromosomes are put together and ranked. Then the top M (the size of population, which is also the number of chromosomes in each generation) chromosomes are chosen as the new son generation. Obviously, this method can ensure that the average fitness value of population increases gradually, and it can also maintain the diversity of the population. The best one in every generation is passed down directly to the next generation. And at the same time, the best chromosome and its fitness value in every generation are recorded in an array. After the computation is finished, the chromosome with best fitness value in the array is taken as the optimal solution. The cycle crossover and one-point mutation [29] are adopted in this paper.

The followings describe the rules used in ACO. In discrete time steps, add one element (edge) to the solution of each ant until the termination condition is met. After an element is added, the quantity of pheromone associate with that element is altered. The amount of pheromone is a scalar value that allows ants to communicate with one another regarding the utility of an element. The accumulated strength of the pheromone on element i is represented by $\tau(i)$. At the beginning of each time step, Eq. (1) is applied to select the next element s to that is added to the solution. The elements that may still be added to the solution by ant k in step r are indicated by $J_k(r)$.$\eta(s)$ measures how good element s would be for the solution; restated, it represents an incremental cost measure. With respect to the TSP, the measure corresponds to the distance between two cities. The state transition rule, global updating rule, and local updating rule as Eqs. (1)- (3), respectively:

$$s = \begin{cases} \arg\max u \in J_k(\mathrm{r})\{[\tau(\mathrm{r},\mathrm{u})].[\eta(\mathrm{r},\mathrm{u})]^{\beta} & if \quad q \le q_0 \\ S & otherwise \end{cases} \quad (1)$$

$$P_k(r,s) = \begin{cases} \dfrac{[\tau(\mathrm{r},\mathrm{s})].[\eta(\mathrm{r},\mathrm{s})]^{\beta}}{\sum_{\mathrm{u} \in \mathrm{J_k}(r)}[\tau(r,z)].[\eta(r,z)]^{\beta}} & if \quad s \in J_k(r) \\ 0 & otherwise \end{cases} \quad (2)$$

$$\tau(r,s) = (1-\alpha)\cdot\tau(r,s) + \alpha\cdot\Delta\tau(r,s) \quad (3)$$

where

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

The characteristics of PPGA-ACO can be summarized as that: by incorporating the ACO into the PPGA, the search performance of both algorithms is improved. On the one hand, based on the initial pheromones on the repertoire (i.e. the map in TSP) secreted by the fittest antibodies in each sub-population, the ACO is utilized to act as a local search operator. As a result, the local search ability of GA can be enhanced. Moreover, due to all of the elite sub-population take effects synchronously and interact with each other that is realized by the pheromones, all of the elitist gene segments in each sub-population have been memorized by the repertoire and thus the repertoire can give some hits to lead the PPGA to construct fitter chromosomes based on these segments in the next generation. On the other hand, the speed at which the traditional ACO gives the solution is slow if there is little information pheromone on the path early. To solve this problem, in our proposed algorithm, ACO adopts PPGA to give information pheromone to distribute and thus the convergence speed of ACO can be accelerated.

4. Experimental Results

To demonstrate the performance of our proposed approach, we conduct some computer simulations on PC Pentium 4. In all the experiments of this section we set parameter values, if not differently indicated, as follows: the number of total population size is 100, each sub-population has 10 chromosomes, i.e., there are 10 islands in the Island Model. Besides, the probability of the crossover and mutation operators is 0.8 and 0.3, respectively. The maximum generation number for PPGA-ACO is set as 10000. In each subpopulation, place 30 ants into it and set $\beta = 2, \alpha = 0.1$. All simulations are run 50 repeats to make a statistical analysis.

First and foremost, the convergence speed of PPGA-ACO is depicted. The benchmark instance used is Kroa100 taken from the TSPLIB [1]. In order to find out whether the ACO has taken efforts on PPGA, and further the effects PPGA takes compared with the simple GA, we make a comparison with some other typical algorithm. That is, the simple GA used in [29]; the SA technique [31], using the 2-opt improvement technique; Budinich's SOM, which consists of a traditional SOM applied to the TSP, also presented in [31]; and the expanded SOM (ESom) [32] which, in each iteration, places the neurons close to their corresponding input data (cities) and, at the same time, places them at the convex contour determined by the cities; the pseudo-parallel GA (PPGA) described in [20]. The experimental results are shown in Fig. 2, where the horizontal axis denotes the average length of the best-so-far solutions found over fifty runs, while the vertical axis represents the generation number (iteration). From Fig. 2, it can be concluded that, with the iteration number goes, all the six algorithms are convergent. The fastest convergent speed is possessed by the proposed PPGA-ACO. Furthermore, the final solutions found by PPGA-ACO are also the best during all the algorithms.

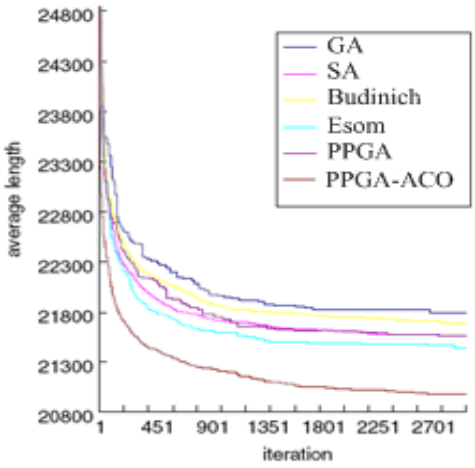

Fig. 2: Comparative results of the convergence speed versus the generation number on six typical algorithm: GA, SA, Budinich, Esom, PPGA, and PPGA-ACO.

In addition, we make an empirical experiments on a large number of traditional algorithms, not only some algorithms used above, but also other powerful approaches. Those approaches involves: The method that involves statistical methods between a SOM's neurons' weights [30] and has a global version (KG: Kohonen network incorporating explicit statistics global), where all cities are used in the neuron dispersion process, and a local version (KL), where only some represented cities are used in the neuron dispersion step; the efficient and integrated SOM

(EiSom) [33], where the Esom procedures are used and the winning neuron is placed at the mean point among its closest neighboring neurons; the efficient SOM technique (Setsp) [34], which defines the updating forms for parameters that use the TSP's number of cities; and Kohonen's cooperative adaptive network (CAN) [35] uses the idea of cooperation between the neurons' close neighbors, uses a number of neurons that is larger than the number of cities in the problem, and uses one improvement's technique for routes called near-tour to tour construction.

For the sake of perspicuity, some indexes are utilized to analyze the simulation results. The optimum tour length that listed in the TSPLIB is labeled as D_{opt}. The PDB which indicates the percentage deviation from the D_{opt} of best distance D_b is defined as follows:

$$PDB = (D_b - D_{opt}) \times 100 / D_{opt} \quad (4)$$

Table 1 lists the experimental results of the nine different methods. The results that recorded are PDBs, whose property is that the smaller values of which, the better qualities of the solutions. As can be observed from this table, PPGA-ACO always performs the best solutions than the other eight algorithms, revealing that the ACO not only enables PPGA to find better solutions, but also can outperform other state-of-art approaches.

Table 1: Experimental results with different methods.

	eil51	rd100	pr124	kroA200	pcb442
KG	2.86	2.62	0.49	6.57	10.45
KL	2.86	2.09	0.08	5.72	11.07
SA	2.33	3.26	1.26	5.61	9.15
Budinich	3.10	3.16	1.62	6.13	8.43
ESom	2.10	1.96	0.67	2.91	7.43
EiSom	2.56	-	-	1.64	6.11
Setsp	2.22	2.60	-	3.12	10.16
CAN	1.65	1.23	2.36	0.93	5.89
PPGA-ACO	0.00	0.10	0.01	0.31	1.26

5. Conclusions

In this paper, we proposed a hybrid new algorithm call PPGA-ACO, which combined the pseudo-parallel genetic algorithm with the ant colony optimization. The ant colony optimization carried out the function of information exchange during different sub-population of genetic algorithms, using a simple but effective island model. The novelty of this paper was the hybridization of the two algorithms, which was considered in the reference for the first time.

The performance of the proposed PPGA-ACO was verified through applying it to the famous TSP. Experimental results based on several benchmark instances taken from TSPLIB demonstrated that the PPGA-ACO was more robust and effective than its variant GA, PPGA, and other typical meta-heuristics, such as neural networks.

In the future, we plan to apply PPGA-ACO to other combinational optimization problems, such as the graph planarization problem, the job shop scheduling problem, and so on.

References

[1] G. Reinelt, "TSPLIB -a traveling salesman problem library," ORSA Journal on Computing, vol.3, pp.376-384, 1991.

[2] S. Chatterjee, C. Carrera and L. A. Lynch, "Genetic algorithms and traveling salesman problems," European Journal of Operational Research, vol.93, pp.490-510, 1996.

[3] G. Gutin and A. P. Punnen, The Traveling Salesman Problem and Its Variations, Kluwer, 2002.

[4] D. S. Johnson and L. A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization," in E. H. L. Aarts & J. K. Lenstra (Eds.), Local Search in Combinatorial Optimization (pp.215-310), John Wiley and Sons, 1997.

[5] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996.

[6] Z. H. Shen, Y. K. Zhao and X. R. Wang, "Niche Pseudo-parallel Genetic Algorithm for Path Optimization of Autonomous Robot," Modern Electronics Technique, vol.15, pp.85-87, 2005.

[7] M. Dorigo and T. Stutzle, Ant Colony Optimization, MIT Press, 2004.

[8] H. K. Tsai, J. M. Yang, Y. F. Tsai and C. Y. Kao, "An Evolutionary Algorithm for Large Traveling Salesman Problems," IEEE Trans. on Systems, Man, and Cybernetics-Part B, vol.34, no.4, pp.1718-1729, 2004.

[9] K. F. Man, K. S. Tang and S. Kwong, Genetic Algorithms: Concepts and Designs, Springer, New York, 1999.

[10] J. E. Rawlins and Gregory, Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991.

[11] L. D. Whitley, Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, 1993.

[12] G. Winter, J. Periaux, M. Galan and P. Cuesta, Genetic Algorithms in Engineering and Computer Science, Wiley, New York, 1995.

[13] A. M. S. Zalzala and P. J. Fleming, Genetic Algorithms in Engineering System, The Institution of Electrical Engineers, London, 1997.

[14] J. T. Alander (Eds.), Proceedings of the First Nordic Workshop on Genetic Algorithms and their Applications (INWGA), January 9-12, Vaasa Yliopiston Julkaisuja, Vaasa, 1995.

[15] J. R. Koza, D. E. Goldberg, D. B. Fogel and R. L. Riolo (Eds.), Genetic Programming: Proceedings of the First Annual Conference, Stanford University, July 28-31, MIT Press, Cambridge, 1996.

[16] W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela and R. E. Smith (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, FL, July 13-17, Morgan Kaufmann, San Mateo, CA, 1999.
[17] J. H. Holland, Adaption in Natural and Artificial System, MIT Press, Cambridge, MA, 1975.
[18] J. Potvin, "Genetic Algorithms for the Traveling Salesman Problems," Annals of Operations Research, vol.63, pp.330-370, 1996.
[19] L. Schmitt and M. Amini, "Performance Characteristics of Alternative Genetic Algorithm Approaches to the Traveling Salesman Problem using Path Representation: An Empirical Study," European Journal of Operational Research, vol.108, no.3, pp.551-570, 1998.
[20] M. Zhou and S. D. Sung, Principle and Application of Genetic Algorithm, Defense Industries Press, Beijing, 1999.
[21] X. F. Chen, W. H. Gui and M. Wu, "Chaotic migration-based pseudo parallel genetic algorithm and its application," Control Theory & Applications, vol. 21, no. 6, pp. 997-1002, 2004.
[22] L. Zou, J. C. Xia and G. A. Hu, "Real Coding Based Multi-population Parallel Genetic Algorithm," Mini-micro Systems, vol25, no. 6, pp.982-986, 2004.
[23] M. G. Schleuter, "ASPARAGOS: An Asynchronous Paoallel Genetic Optimization Strategy," Proc. of 3rd Int. Conf. On Genetic Algorithms, Morgan Kaufmann, pp. 422-427, 1989.
[24] M. Dorigo, V. Maniezzo and A. Colorni, "Positive Feedback as a Search Strategy," Technical Report 91-106, Dip. Elettronica, Polotecnico di Milano, 1991.
[25] E. Bonabeau, M. Dorigo and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, MIT Press, 2000.
[26] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning appraoch to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, vol.1, no.1, pp.53-66, 1997.
[27] C. Blum, "Ant colony optimization: Introduction and recent trends," Physics of Life Reviews, vol.2, pp.353-373, 2005.
[28] Z. Lee, S. Su, C. Chuang and K. Liu, "Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment," Applied Soft Computing, vol.8, pp.55-78, 2008.
[29] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[30] N. Aras, B. J. Oommen and I. K. Altinel, "The Kohonen network incorporating explicit statistics and its apllication to the travelling salesman problem," Neural Networks, vol.12, pp.1273-1284, 1999.
[31] M. Budinich, "A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing," Neural Computation, vol.8, pp.416-424, 1996.
[32] K. S. Leung, H. D. Jin and Z. B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," Neurocomputing, vol.62, pp.267-292, 2004.
[33] H. D. Jin, K. S. Leung, M. L. Wong and Z. B. Xu, "An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem," IEEE Trans. on Systems, Man, and Cybernetics-Part B, vol.33, no.6, pp.877-887, 2003.
[34] F. C. Vieira, A. D. Doria Neto and J. A. Costa, "An efficient approach to the traveling salesman problem using self-organizing maps," International Journal of Neural Systems, vol. 13, no.2, pp.59-66, 2003.
[35] E. M. Cochrane and J. E. Beasley, "The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem," Neural Networks, vol.16, no.10, pp.1499-1525, 2003.

Sheng Lir received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China in 2006 and an M.S. degree from University of Toyama, Toyama, Japan in 2009. Now, he is working toward the D.E. degree at University of Toyama, Toyama, Japan. His main research interests are intelligence computing, neural networks, swarm intelligent algorithms, and combinational optimization problems.

Huiqin Chen received the B.S. degree from HoHai University, Nanjing, China in 2006 and an M.S. degree from University of Toyama, Toyama, Japan in 2009. Now, she is working toward the D.E. degree at University of Toyama, Toyama, Japan. Her main research interests are intelligence computing, neural networks, swarm intelligent algorithms, and combinational optimization problems.

Zheng Tang received the B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tshinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an instructor in the institute of microelectronics at Tshinghua University. From 1990 to 1999, he was an associate professor in the department of electrical and electronic engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a professor in the department of intellectual information systems.