Multi-Colony Ant Algorithm

Enxiu Chen[1] and Xiyu Liu[2]
[1]*School of Business Administration, Shandong Institute of Commerce and Technology, Jinan, Shandong;*
[2]*School of Management & Economics, Shandong Normal University, Jinan, Shandong, China*

1. Introduction

The first ant colony optimization (ACO) called ant system was inspired through studying of the behavior of ants in 1991 by Macro Dorigo and co-workers [1]. An ant colony is highly organized, in which one interacting with others through pheromone in perfect harmony. Optimization problems can be solved through simulating ant's behaviors. Since the first ant system algorithm was proposed, there is a lot of development in ACO. In ant colony system algorithm, local pheromone is used for ants to search optimum result. However, high magnitude of computing is its deficiency and sometimes it is inefficient. Thomas Stützle et al. introduced MAX-MIN Ant System (MMAS) [2] in 2000. It is one of the best algorithms of ACO. It limits total pheromone in every trip or sub-union to avoid local convergence. However, the limitation of pheromone slows down convergence rate in MMAS.

In optimization algorithm, it is well known that when local optimum solution is searched out or ants arrive at stagnating state, algorithm may be no longer searching the global best optimum value. According to our limited knowledge, only Jun Ouyang et al [3] proposed an improved ant colony system algorithm for multi-colony ant systems. In their algorithms, when ants arrived at local optimum solution, pheromone will be decreased in order to make algorithm escaping from the local optimum solution.

When ants arrived at local optimum solution, or at stagnating state, it would not converge at the global best optimum solution. In this paper, a modified algorithm, multi-colony ant system based on a pheromone arithmetic crossover and a repulsive operator, is proposed to avoid such stagnating state. In this algorithm, firstly several colonies of ant system are created, and then they perform iterating and updating their pheromone arrays respectively until one ant colony system reaches its local optimum solution. Every ant colony system owns its pheromone array and parameters and records its local optimum solution. Furthermore, once a ant colony system arrives at its local optimum solution, it updates its local optimum solution and sends this solution to global best-found center. Thirdly, when an old ant colony system is chosen according to elimination rules, it will be destroyed and reinitialized through application of the pheromone arithmetic crossover and the repulsive operator based on several global best-so-far optimum solutions. The whole algorithm implements iterations until global best optimum solution is searched out. The following sections will introduce some concepts and rules of this multi-colony ant system.

This paper is organized as follows. Section II briefly explains the basic ACO algorithm and its main variant MMAS we use as a basis for multi-colony ant algorithm. In Section III we describe detailed how to use both the pheromone crossover and the repulsive operator to reinitialize a stagnated colony in our multi-colony ant algorithm. A parallel asynchronous algorithm process is also presented. Experimental results from the multi-colony ant algorithm are presented in Section IV along with a comparative performance analysis involving other existing approaches. Finally, Section V provides some concluding remarks.

2. Basic ant colony optimization algorithm

The principle of ant colony system algorithm is that a special chemical trail (pheromone) is left on the ground during their trips, which guides the other ants towards the target solution. More pheromone is left when more ants go through the trip, which improved the probability of other's ants choosing this trip. Furthermore, this chemical trail (pheromone) has a decreasing action over time because of evaporation of trail. In addition, the quantity left by ants depends on the number of ants using this trail.

Fig.1 presents a decision-making process of ants choosing their trips. When ants meet at their decision-making point *A*, some choose one side and some choose other side randomly. Suppose these ants are crawling at the same speed, those choosing short side arrive at decision-making point *B* more quickly than those choosing long side. The ants choosing by chance the short side are the first to reach the nest. The short side receives, therefore, pheromone earlier than the long one and this fact increases the probability that further ants select it rather than the long one. As a result, the quantity of pheromone is left with higher speed in short side than long side because more ants choose short side than long side. The number of broken line in Fig. 1 is direct ratio to the number of ant approximately. Artificial ant colony system is made from the principle of ant colony system for solving kinds of optimization problems. Pheromone is the key of the decision-making of ants.

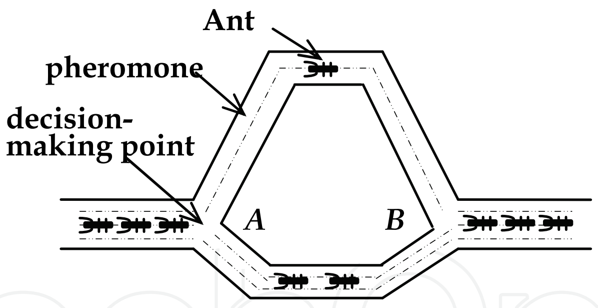

Fig. 1. A decision-making process of ants choosing their trips according to pheromone.

ACO was initially applied to the traveling salesman problem (TSP) [4][5]. The TSP is a classical optimization problem, and is one of a class of NP-Problem. This article also uses the TSP as an example application. Given a set of *N* towns, the TSP can be stated as the problem of finding a minimal length closed tour that visits each town once. Each city is a decision-making point of artificial ants.

Define (*i,j*) is an edge of city *i* and city *j*. Each edge (*i,j*) is assigned a value (length) d_{ij}, which is the distance between cities *i* and *j*. The general MMAX [2] for the TSP is described as following:

2.1 Pheromone updating rule

Ants leave their pheromone on edges at their every traveling when ants complete its one iteration. The sum pheromone of one edge is defined as following

$$\tau_{ij}(t+1) = \Delta\tau_{ij} + (1 - \rho)\tau_{ij}(t) \tag{1}$$

$\rho \in (0,1)$, 1-ρ is persistence rate of previous pheromone. ρ is defined as evaporation rate of pheromone.
In MMAS, only the best ant updates the pheromone trails and that the value of the pheromone is bound. Therefore, the pheromone updating rule is given by

$$\tau_{ij}(t+1) = [\Delta\tau_{ij}^{best} + (1 - \rho)\tau_{ij}(t)]_{\tau_{\min}}^{\tau_{\max}} \tag{2}$$

where τ_{max} and τ_{min} are respectively the upper and lower bounds imposed on the pheromone; and $\Delta\tau_{ij}^{best}$ is:

$$\Delta\tau_{ij}^{best} = \begin{cases} 1 / L_{best} & \text{if}(i,j) \text{ belongs to the best tour,} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where L_{best} is solution cost of either the iteration-best or the best-so-far or a combination of both [2].

2.2 Ants moving rule

Ants move from one city to another city according to probability. Firstly, cities accessed must be placed in taboo table. Define a set of cities never accessed of the kth ant as $allowed_k$. Secondly, define a visible degree η_{ij}, $\eta_{ij} = 1/d_{ij}$. The probability of the kth ant choosing city is given by

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum\limits_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha}[\eta_{ik}]^{\beta}} & j \in allowed_k \\ 0 & \text{else} \end{cases} \tag{4}$$

where a and β are important parameters which determine the relative influence of the trail pheromone and the heuristic information.
In this article, the pseudo-random proportional rule given in equation (5) is adopted as in ACO [4] and modified MMAS [6].

$$j = \begin{cases} \arg \max\limits_{k \in allowed_k} \{\tau_{ik}(t)[\eta_{ik}]^{\beta}\} & \text{if } p \le p_0 \\ J & \text{else} \end{cases} \tag{5}$$

where p is a random number uniformly distributed in [0,1]. Thus, the best possible move, as indicated by the pheromone trail and the heuristic information, is made with probability $0 \le p_0 < 1$ (exploitation); with probability 1-p_0 a move is made based on the random variable J with distribution given by equation (4) (biased exploration).

2.3 Pheromone trail Initialization

At the beginning of a run, we set $\tau_{max} = 1/((1-\rho)C_{nn})$, $\tau_{min} = \tau_{max}/2N$, and the initial pheromone values $\tau_{ij}(0) = \tau_{max}$, where C_{nn} is the length of a tour generated by the nearest-neighbor heuristic and N is the total number of cities.

2.4 Stopping rule

There are many conditions for ants to stop their traveling, such as number limitation of iteration, CPU time limitation or the best solution.
From above describing, we can get detail procedure of MMAS. MMAS is one of the most studied ACO algorithms and the most successful variants [5].

3. Multi-colony ant system based on a pheromone arithmetic crossover and a repulsive operator

3.1 Concept

1. **Multi-Colony Ant System Initiating.** Every ant colony system owns its pheromone array and parameters a, β and ρ. In particular, every colony may possess its own arithmetic policy. For example, all colonies use different ACO algorithms respectively. Some use basic Ant System. Some use elitist Ant System, ACS, MMAS, or rank-based version of Ant System etc. The others maybe use hyper-cube framework for ACO.
Every ant colony system begins to iterate and update its pheromone array respectively until it reaches its local optimum solution. It uses its own search policy. Then it sends this local optimum solution to the global best-found center. The global best-found center keeps the global top M solutions, which are searched thus far by all colonies of ant system. The global best-found center also holds parameters a, β and ρ for every solution. These parameters are equal to the colony parameters while the colony finds this solution. Usually M is larger than the number of colonies.
2. **Old Ant Colony being Eliminated Rule.** We destroy one of the old colonies according to following rules:
    a. A colony who owns the smallest local optimum solution among all colonies.
    b. A colony who owns the largest generations since its last local optimum solution was found.
    c. A colony that has lost diversity. In general, there are supposed to be at least two types of diversity [7] in ACO: (i) diversity in finding tours, and (ii) diversity in depositing pheromone.
3. **New Ant Colony Creating by Pheromone Crossover.** Firstly, we select m ($m \ll M$) solutions from M global best-so-far optimums in the global best-found center randomly. Secondly, we deliberately initialize the pheromone trails of this new colony to $\rho(t)$ which starts with $\rho(t)=\tau\max(t)=1/((1-\rho)\cdot L_{best}(t))$, achieving in this way a higher exploration of solutions at the start of algorithm and a higher exploitation near the top m global optimum solutions at the end of algorithm. Where $L_{best}(t)$ is the best-so-far solution cost of all colonies in current t time. Then these trails are modified using arithmetic crossover by

$$\tau_{ij} = \rho(t) + \sum_{k=1}^{m} c_k \mathrm{rand}_k()\Delta\tau_{ij}^k \tag{6}$$

where $\Delta\tau_{ij}^k = 1/L_{best}^k$ in which edge (i,j) is on the kth global-best solution and L_{best}^k denotes the kth global-best solution cost in the m chosen solutions; $rand_k()$ is a random function uniformly distributed in the range [0,1]; c_k is the weight of $\Delta\tau_{ij}^k$ and $\sum_{k=1}^{m} c_k = 2$ because the mathematical expectation of $rand_k()$ equals $1/2$. Last, the parameters a, β and ρ are set using arithmetic crossover by:

$$\alpha = \sum_{k=1}^{m} c_k \mathrm{rand}_k()\alpha_k \ , \beta = \sum_{k=1}^{m} c_k \mathrm{rand}_k()\beta_k \ , \rho = \sum_{k=1}^{m} c_k \mathrm{rand}_k()\rho_k \qquad (7)$$

where a_k, β_k and ρ_k belong to the kth global-best solution in the m chosen solutions.
After these operations, the colony starts its iterating and updating its local pheromone anew.
4. **Repulsive Operator.** As Shepherd and Sheepdog algorithm [8], we introduce the attractive and repulsive ACO in trying to decrease the probability of premature convergence further. We define the attraction phase merely as the basic ACO algorithm. In this phase the good solutions function like "attractors". In the colony of this phase, the ants will then be attracted to the solution space near good solutions. However, the new colony which was just now reinitialized using pheromone arithmetic crossover maybe are redrawn to the same best-so-far local optimum solution again which was found only a moment ago. As a result, that wastes an amount of computational resource. Therefore, we define the second phase repulsion, by subtracting the term $\Delta\tau_{ij}^{best}$ in association with the best-so-far solution in the pheromone-update formula when we reinitialize a new colony. Then the equation (6) will become as:

$$\tau_{ij} = \rho(t) + \sum_{k=1}^{m} c_k \mathrm{rand}_k()\Delta\tau_{ij}^{k} - c_{best}\Delta\tau_{ij}^{best} \qquad (8)$$

where $\Delta\tau_{ij}^{best} = 1 / L_{best}$ in which edge (i,j) is on the best-so-far solution, L_{best} denotes the best-so-far solution cost, c_{best} is the weight of $\Delta\tau_{ij}^{best}$, and the other coefficients are the same as in the equation (6). In that phase the best-so-far solution functions like "a repeller" so that the ants can move away from the vicinity of the best-so-far solution.
We identify our implementation of this model based on a pheromone crossover and a repulsive operator with the acronym MCA.

3.2 Parallel asynchronous algorithm design for multi-colony ant algorithms

As in [9], we propose a parallel asynchronous algorithm process for our multi-colony ant algorithm in order to make efficient use of all available processors in a heterogeneous cluster or heterogeneous computing environments. Our process design follows a master/slave paradigm. The master processor holds a global best-found center, sends colony initialization parameters to the slave processors and performs all decision-making processes such as the global best-found center updates and sorts, convergence checks. It does not perform any ant colony algorithm iteration. However, the slave processors repeatedly execute ant colony algorithm iteration using the parameters assigned to them. The tasks performed by the master and the slave processors are as follows:

- Master processor

1. Initializes all colonies' parameters and sends them to the slave processors;
2. Owns a global best-found center which keeps the global top M solutions and their parameters;
3. Receives local optimum solution and parameters from the slave processors and updates its global best-found center;
4. Evaluate the effectiveness of ant colonies in the slave processors;
5. Initializes a set of new colony's parameters by using both a pheromone crossover and a repulsive operator based on multi-optimum for the worst ant colony;

6. Chooses one of the worst ant colonies to kill and sends the new colony parameters and kill command to the slave processor who owns the killed colony;
7. Checks convergence.

- Slave processor

1. Receives a set of colony's parameters from the master processor;
2. Initializes an ant colony and starts iteration;
3. Sends its local optimum solution and parameters to the master processor;
4. Receives kill command and parameters from the master processor, and then use these parameters to reinitialize and start iteration according to the equation (8).

Once the master processor has performed the initialization step, the initialization parameters are sent to the slave processors to execute ant colony algorithm iteration. Because the contents of communication between the master processor and the slave processors only are some parameters and sub-optimum solutions, the ratio of communication time between the master and the slaves to the computation time of the processors of this system is relatively small. The communication can be achieved using a point-to-point communication scheme implemented with the Message Passing Interface (MPI). Only after obtaining its local optimum solution, the slave processor sets message to the master processor (Fig. 2). During this period, the slave processor continues its iteration until it gets kill command from the master processor. Then the slave processor will initialize a new ant colony and reiterate.

To make the most use of the heterogeneity of the band of communication between the master processor and the slave processors, we can select some slave processors the band between which and the master processor is very straightway. We never kill them but only send the global best-so-far optimum solution to them in order to speed up their local pheromone arrays update and convergence.

A pseudo-code of a parallel asynchronous MCA algorithm is present as follow:

```
• Master processor
Initialize Optimization
    Initialize parameters of all colonies
    Send them to the slave processors;
Perform Main-Loop
    Receive local optimum solution
       and parameters from the slave processors
    Update the global best-found center
    Check convergence
    If (eliminating rule met) then
       Find the worst colony
       Send kill command and a set of new parameters to it
Report Results
• Slave processor
Receive Initialize parameters from the master processor
Initialize a new local ant colony
Perform Optimization
    For $k$ = 1, number of iterations
      For $i$ = 1, number of ants
        Construct a new solution
```

```
            If (kill command and a set of new parameters received) then
              Goto initialize a new local ant colony;
        Endfor
        Modify its local pheromone array
        Send its local optimum solution and parameters to the master processor
    Endfor
End
```

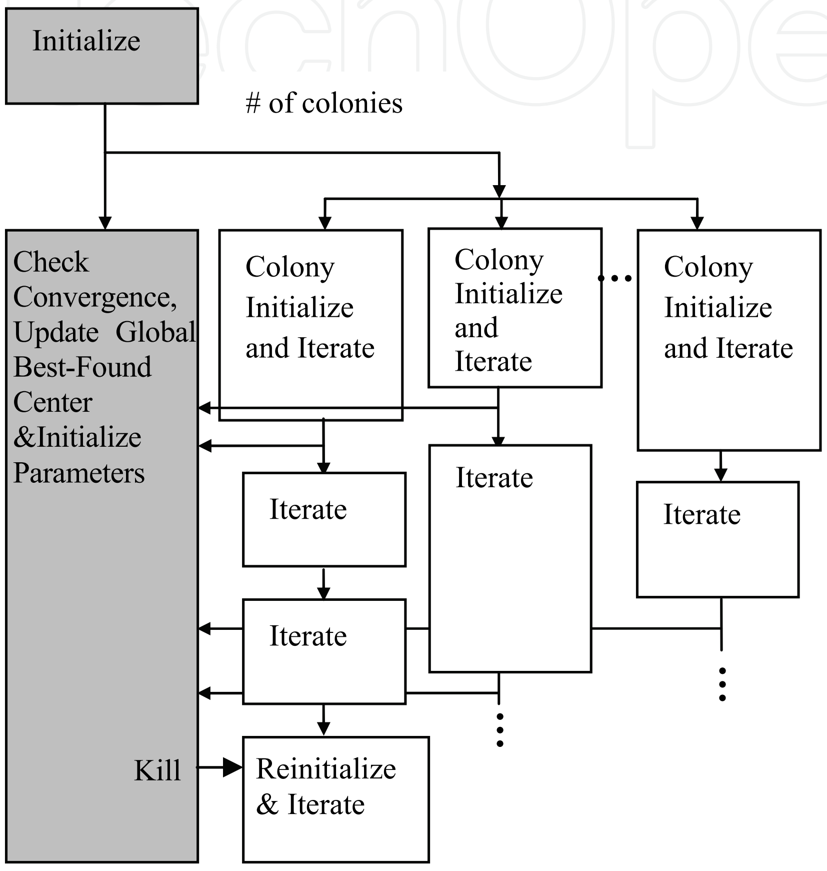
Initialize
# of colonies
Check Convergence, Update Global Best-Found Center &Initialize Parameters
Colony Initialize and Iterate
Colony Initialize and Iterate
Colony Initialize and Iterate
Iterate
Iterate
Iterate
Iterate
Kill
Reinitialize & Iterate

Fig. 2. Block diagram for parallel asynchronous algorithm. Grey boxes indicate activities on the master processor.

4. Experiment results

4.1 Parallel independent runs & Sequential algorithm

In this parallel model, *k* copies of the same sequential MMAS algorithm are simultaneously and independently executed using different random seeds. The final result is the best solution among all the obtained ones. Using parallel independent runs is appealing as basically no communication overhead is involved and nearly no additional implementation effort is necessary. We identify the implementation of this model with the acronym PIR. Max Manfrin et al [10] find that PIR owns the better performance than any other parallel model.

In order to have a reference algorithm for comparison, we also test the equivalent sequential MMAS algorithm. It runs for the same overall generations as a parallel algorithm (*k*-times the generations of a parallel algorithm). We identify the implementation of this model with the acronym SEQ.

4.2 Experimental setup

For this experiment, we use MAX-MIN Ant System as a basic algorithm for our parallel implementation. We remain the occasional pheromone re-initializations applied in the MMAS described in [2], and a best-so-far pheromone update. Our implementation of MMAS is based on the publicly available ACOTSP code [11]. Our version also includes a 3-opt local search, uses the mean 0.05-branching factor and *don't look bits* for the outer loop optimization, and sets q_0=0.95.

We tested our algorithms on the Euclidian 2D TSP PCB442 from the TSPLIB [12]. The smallest tour length for this instance is known to be 50778. As parameter setting we use α=1, β=2 and ρ=0.1 for PIR and SEQ; and $\alpha\in[0.8,1.2]$, $\beta\in[2,5]$, $\rho\in[0.05,0.15]$ for MCA. Computational experiments are performed with k=8 colonies of 25 ants over T=200 generations for PIR and MCA, but 25 ants and T=1600 for SEQ, i.e. the total number of evaluated solutions is 40000 (=25*8*200=25*1600). We select m=4 best solutions, c_k=2/4=0.5 and c_{best}=0.1 in the pheromone arithmetic crossover and the repulsive operator for MCA. All given results are averaged over 1000 runs. As far as eliminated rules in MCA, we adopt rule (b). If a colony had run more than 10 generations (2000 evaluations) for PCB442 since its local optimum solution was updated, we think it had arrived at the stagnating state.

4.3 Experimental result

The Fig. 3 shows cumulative run-time distribution that certain levels of solution quality are obtained depending on the number so far evaluated solutions. There is a rapid decrease in tour length early in the search in the SEQ algorithm because it runs more generations than SEQ and MCA in the same evaluations. After this, the improvement flattened out for a short while before making another smaller dip. Finally, the SEQ algorithm decreases at a much slower pace quickly and tends to stagnate prematurely. Although, the tour length decreases more slowly in PIR and MCA than in SEQ early in the search, after about 6600 evaluations SEQ and MCA all give better results than PIR in average. Moreover, for every level of solutions MAC gives the better performance than PIR. Conclusively, SEQ has great risk of getting stuck on a local optimum; however, the MCA is able to escape local optima because of the repulsive operator and the pheromone crossover.

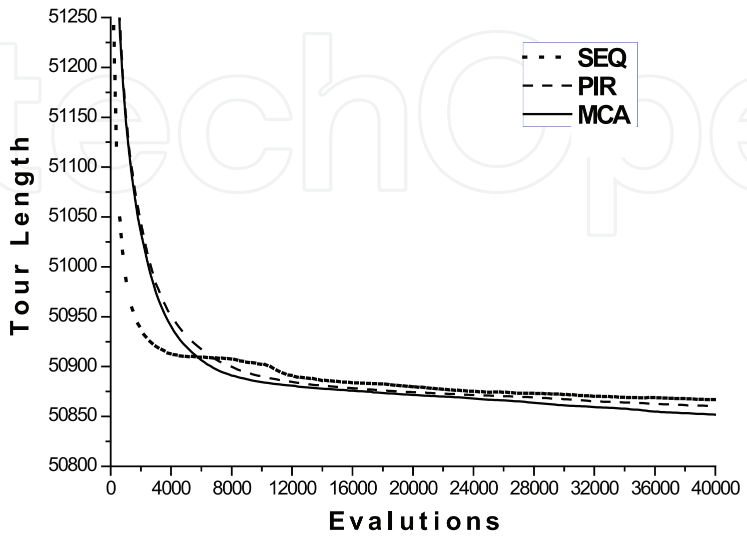

Fig. 3. The cumulative run-time distribution is given for PVB442

5. Conclusion

In this paper, an improved ant colony system, multi-colony ant algorithm, is presented. The main aim of this method is to increase the ant colonies' capability of escaping stagnating state. For this reason, a new concept of multiple ant colonies has been presented. It creates a new colony of ants to iterate, which is accomplished through application of the pheromone arithmetic crossover and the repulsive operator based on multi-optimum when meeting at the stagnating state of the iteration or local optimum solution. At the same time, the main parameters α, β and ρ of algorithm are self-adaptive. In this paper, a parallel asynchronous algorithm process is also presented.
From above exploring, it is obvious that the proposed multi-colony ant algorithm is an effective facility for optimization problems. The result of experiment has shown that the proposed multi-colony ant algorithm is a precise method for TSP. The speed of multi-colony ant algorithm's convergence is faster than that of the parallel independent runs (PIR).
At the present time, our parallel code only allows for one computer. In future versions, we will implement MPI-based program on a computer cluster.

6. Acknowledgment

Research is also supported by the Natural Science Foundation of China (No.60873058, No.60743010), the Natural Science Foundation of Shandong Province (No. Z2007G03), and the Science and Technology Project of Shandong Education Bureau. This research is also carried out under the PhD foundation of Shandong Institute of Commerce and Technology, China.

7. References

[1] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive Feedback as a Search Strategy", *Technical Report 91-016*, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy,1991.
[2] T. Stützle and H. H. Hoos, "MAX-MIN Ant System", *Future Generation Computer systems*, 16(8), pp. 889-914, 2000.
[3] J. Ouyang and G. R. Yan, "A multi-group ant colony system algorithm for TSP", *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, pp. 117-121, 2004.
[4] M. Dorigo and L. M. Gambardella, "Ant Colony System: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on evolutionary computation*, 1(1), pp. 53-66, April 1997.
[5] M. Dorigo, B. Birattari, and T. Stuzle, "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, 1(4), pp. 28-39, 2006.
[6] T. Stützle. "Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New", Applications, vol. 220 of DISKI. Sankt Augustin, Germany, Infix, 1999.
[7] Y. Nakamichi and T. Arita, "Diversity Control in Ant Colony Optimization", *Artificial Life and Robotics*, 7(4), pp. 198-204, 2004.

[8] D. Robilliard and C. Fonlupt, "A Shepherd and a Sheepdog to Guide Evolutionary Computation", *Artificial Evolution*, pp. 277-291, 1999.
[9] B. Koh, A. George, R. Haftka, and B. Fregly , "Parallel Asynchronous Particle Swarm Optimization", *International Journal for Numerical Methods in Engineering*, 67(4), pp. 578-595, 2006.
[10] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem", *IRIDIA – Technical Report Series*, TR/IRIDIA/2006-007, March 2006.
[11] T.Stützle. ACOTSP.V1.0.tar.gz
http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html, 2006.
[12] G. Reinelt. TSPLIB95
http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html, 2008.