An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem

Leonora Bianchi[1], Luca Maria Gambardella[1], and Marco Dorigo[2]

[1] IDSIA, Strada Cantonale Galleria 2, CH-6928 Manno, Switzerland
{leonora,luca}@idsia.ch
http://www.idsia.ch
[2] Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brusselles, Belgium
mdorigo@ulb.ac.be
http://iridia.ulb.ac.be/~mdorigo/

Abstract. The Probabilistic Traveling Salesman Problem (PTSP) is a TSP problem where each customer has a given probability of requiring a visit. The goal is to find an a priori tour of minimal expected length over all customers, with the strategy of visiting a random subset of customers in the same order as they appear in the a priori tour.
We address the question of whether and in which context an a priori tour found by a TSP heuristic can also be a good solution for the PTSP. We answer this question by testing the relative performance of two ant colony optimization algorithms, Ant Colony System (ACS) introduced by Dorigo and Gambardella for the TSP, and a variant of it (pACS) which aims to minimize the PTSP objective function.
We show in which probability configuration of customers pACS and ACS are promising algorithms for the PTSP.

1 Introduction

Consider a routing problem through a set V of n customers. On any given instance of the problem each customer i has a known position and a probability p_i of actually requiring a visit, independently of the other customers. Finding a solution for this problem implies having a strategy to determine a tour for each random subset $S \subseteq V$, in such a way as to minimize the expected tour length. The most studied strategy is the a priori one. An a priori strategy has two components: the a priori tour and the updating method. The a priori tour is a tour visiting the complete set V of n customers; the updating method modifies the a priori tour in order to have a particular tour for each subset of customers $S \subseteq V$. A very simple example of updating method is the following: for every subset of customers, visit them *in the same order* as they appear in the a priori tour, skipping the customers that do not belong to the subset. The strategy related to this method is called the 'skipping strategy'. The problem of finding an a priori tour of minimum expected length under the skipping strategy is defined as the Probabilistic Traveling Salesman Problem (PTSP). This is an NP-hard problem [1,2], and was introduced in Jaillet's PhD thesis [3].

The PTSP approach models applications in a delivery context where a set of customers has to be visited on a regular (e.g., daily) basis, but all customers do not always require a visit, and where re-optimizing vehicle routes from scratch every day is unfeasible. In this context the delivery man would follow a standard route (i.e., an a priori tour), leaving out customers that on that day do not require a visit. The standard route of least expected length corresponds to the optimal PTSP solution.

In the literature there are a number of algorithmic and heuristic approaches used to find suboptimal solutions for the PTSP. Heuristics using a nearest neighbor criterion or savings criterion were implemented and tested by Jézéquel [4] and by Rossi-Gavioli [5]. Later, Bertsimas-Jaillet-Odoni [1] and Bertsimas-Howell [6] have further investigated some of the properties of the PTSP and have proposed some heuristics for the PTSP. These include tour construction heuristics (space filling curves and radial sort), and tour improvement heuristics (probabilistic 2-opt edge interchange local search and probabilistic 1-shift local search). Most of the heuristics proposed are an adaptation of a TSP heuristic to the PTSP, or even the TSP heuristic itself, which in some cases gives good PTSP solutions. More recently, Laporte-Louveaux-Mercure [7] have applied an integer L-shaped method to the PTSP and have solved to optimality instances involving up to 50 vertices.

No application of nature-inspired algorithms such as ant colony optimization (ACO) [8] or genetic algorithms has been done yet. This paper investigates the potentialities of ACO algorithms for the PTSP. In the remainder of the paper we first introduce the PTSP objective function and notations (section 2), then we describe the ACO algorithms which we tested (section 3), and finally we show the experimental results obtained (section 4).

2 The PTSP Objective Function

Let us consider an instance of the PTSP. We have a completely connected graph whose nodes form a set $V = \{i = 1, 2, ..., n\}$ of customers. Each customer has a probability p_i of requiring a visit, independently from the others. A solution for this instance is a tour λ over all nodes in V (an 'a priori tour'), to which is associated the expected length objective function

$$E[L_\lambda] = \sum_{S \subseteq V} p(S) L_\lambda(S) \ . \tag{1}$$

In the above expression, S is a subset of the set of nodes V, $L_\lambda(S)$ is the distance required to visit the subset of customers S (in the same order as they appear in the a priori tour), and $p(S)$ is the probability for the subset of customers S to require a visit:

$$p(S) = \prod_{i \in S} p_i \prod_{i \in V-S} (1 - p_i) \ . \tag{2}$$

Jaillet [3] showed that the evaluation of the PTSP objective function (eq.(1)) can be done in $O(n^2)$. In fact, let us consider (without loss of generality) an a priori tour $\lambda = (1, 2, \ldots, n)$; then its expected length is

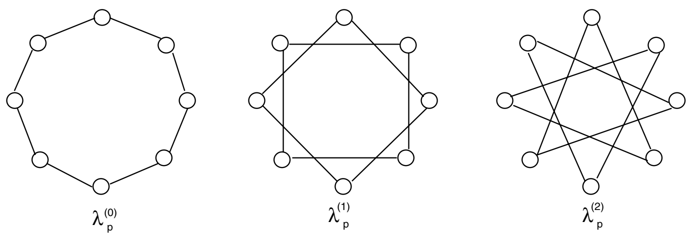

Fig. 1. The lengths of these sets of (sub)tours, $\lambda_p^{(0)}$, $\lambda_p^{(1)}$ and $\lambda_p^{(2)}$, constitute the first three terms of the expected length for the homogeneous PTSP. From left to right, the total length of each set of (sub)tours gives the terms $L_\lambda^{(0)}$, $L_\lambda^{(1)}$ and $L_\lambda^{(2)}$ of equation (4).

$$E[L_\lambda] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} d_{ij}p_ip_j \prod_{k=i+1}^{j-1}(1-p_k) + \sum_{i=1}^{n}\sum_{j=1}^{i-1} d_{ij}p_ip_j \prod_{k=i+1}^{n}(1-p_k)\prod_{l=1}^{j-1}(1-p_l) \ . \quad (3)$$

This expression is derived by looking at the probability for each arc of the complete graph to be used, that is, when the a priori tour is adapted by skipping a set of customers which do not require a visit. For instance, an arc (i,j) is actually used only when customers i and j do require a visit , while customers $i+1, i+2, ..., j$ do not require a visit. This event occurs with probability $p_ip_j \prod_{k=i+1}^{j-1}(1-p_k)$ (when $j \leq n$). In the special class of PTSP instances where $p_i = p$ for all customers $i \in V$ (the homogeneous PTSP), equation (3) becomes

$$E[L_\lambda] = p^2 \sum_{r=0}^{n-2}(1-p)^r L_\lambda^{(r)} \quad (4)$$

where $L_\lambda^{(r)} \equiv \sum_{j=1}^{n} d(j,(j+1+r) \bmod n)$. The $L_\lambda^{(r)}$'s have the combinatorial interpretation of being the lengths of a collection of gcd(n,$r+1$) sub-tours[1] $\lambda_p^{(r)}$, obtained from tour λ by visiting one customer and skipping the next r customers. As an example, Fig. 1 shows $\lambda_p^{(0)}$ (i.e., the a priori tour), $\lambda_p^{(1)}$ and $\lambda_p^{(2)}$ for a PTSP with 8 customers.

3 Ant Colony Optimization

In ACO algorithms a colony of artificial ants iteratively constructs solutions for the problem under consideration using artificial pheromone trails and heuristic information. The pheromone trails are modified by ants during the algorithm execution in order to store information about 'good' solutions. Most ACO algorithms follow the algorithmic scheme given in Fig. 2.

[1] The term 'gcd' stays for 'greatest common divisor'.

```
procedure ACO metaheuristic for combinatorial optimization problems
    Set parameters, initialize pheromone trails
    while (termination condition not met)
        ConstructSolutions
        (ApplyLocalSearch)
        UpdateTrails
    end while
```

Fig. 2. High level pseudocode for the ACO metaheuristic.

ACO are solution construction algorithms, which, in contrast to local search algorithms, may not find a locally optimal solution. Many of the best performing ACO algorithms improve their solutions by applying a local search algorithm after the solution construction phase. Our primary goal in this work is to analyze the PTSP tour construction capabilities of ACO, hence in this first investigation we do not use local search.

We apply to the PTSP Ant Colony System (ACS) [9,10], a particular ACO algorithm which was successfully applied to the TSP. We also consider a modification of ACS which explicitly takes into account the PTSP objective function (we call this algorithm probabilistic ACS, that is, pACS). In the following, we describe how ACS and pACS build a solution and how they update pheromone trails.

3.1 Solution Construction in ACS and pACS

A feasible solution for an n-city PTSP is an a priori tour which visits all customers. Initially m ants are positioned on their starting cities chosen according to some initialization rule (e.g., randomly). Then, the solution construction phase starts (procedure *ConstructSolutions* in Fig. 2). Each ant progressively builds a tour by choosing the next customer to move to on the basis of two types of information, the pheromone τ and the heuristic information η. To each arc joining two customers i, j it is associated a varying quantity of pheromone τ_{ij}, and the heuristic value $\eta_{ij} = 1/d_{ij}$, which is the inverse of the distance between i and j. When an ant k is on city i, the next city is chosen as follows.

– With probability q_0, a city j that maximizes $\tau_{ij} \cdot \eta_{ij}^{\beta}$ is chosen in the set $J_k(i)$ of the cities not yet visited by ant k. Here, β is a parameter which determines the relative influence of the heuristic information.
– With probability $1 - q_0$, a city j is chosen randomly with a probability given by

$$p_k(i,j) = \begin{cases} \frac{\tau_{ij} \cdot \eta_{ij}^{\beta}}{\sum_{r \in J_k(i)} \tau_{ir} \cdot \eta_{ir}^{\beta}}, & \text{if } j \in J_k(i) \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Hence, with probability q_0 the ant chooses the best city according to the pheromone trail and to the distance between cities, while with probability $1 - q_0$ it explores the search space in a biased way.

3.2 Pheromone Trails Update in ACS and pACS

Pheromone trails are updated in two stages. In the first stage, each ant, after it has chosen the next city to move to, applies the following local update rule:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \tag{6}$$

where ρ, $0 < \rho \leq 1$, and τ_0, are two parameters. The effect of the local updating rule is to make less desirable an arc which has already been chosen by an ant, so that the exploration of different tours is favored during one iteration of the algorithm.

The second stage of pheromone update occurs when all ants have terminated their tour. Pheromone is modified on those arcs belonging to the best tour since the beginning of the trial (best-so-far tour), by the following global updating rule

$$\tau_{ij} \leftarrow (1-\alpha) \cdot \tau_{ij} + \alpha \cdot \Delta\tau_{ij}, \tag{7}$$

where

$$\Delta\tau_{ij} = ObjectiveFunc_{best}^{-1} \tag{8}$$

with $0 < \alpha \leq 1$ being the pheromone decay parameter, and $ObjectiveFunc_{best}$ is the value of the objective function of the best-so-far tour. In ACS the objective function is the a priori tour length, while in pACS the objective function is the PTSP expected length of the a priori tour. In the next section we discuss in more detail the differences between ACS and pACS.

3.3 Discussion of Differences between ACS and pACS

Differences between ACS and pACS are due to the fact that the two algorithms exploit different objective functions in the pheromone updating phase. The global updating rule of equations (7) and (8) implies two differences in the way ACS and pACS explore the search space. The first and most important difference is the set of arcs on which pheromone is globally increased, which is in general different in ACS and pACS. In fact, the 'best tour' in eq. (8) is relative to the objective function. Therefore in ACS the search will be biased toward the shortest tour, while in pACS it will be biased toward the tour of minimum expected length. The second difference between ACS and pACS is in the quantity $\Delta\tau_{ij}$ by which pheromone is increased on the selected arcs. This aspect is less important than the first, because ACO in general is more sensitive to the difference of pheromone among arcs than to its absolute value.

The length of an a priori tour (ACS objective function) may be considered as an $O(n)$ approximation to the $O(n^2)$ expected length (pACS objective function). In general, the worse the approximation, the worse will be the solution quality of ACS versus pACS. The quality of the approximation depends on the set of customer probabilities p_i. In the homogeneous PTSP, where customer probability is p for all customers, it is easy to see the relation between the two objective functions. For a given a priori tour L_λ we have

$$\Delta = L_\lambda - E[L_\lambda] = (1-p^2)L_\lambda - \sum_{r=1}^{n-2}(1-p)^r L_\lambda^{(r)}, \tag{9}$$

which implies

$$\Delta \sim O(q \cdot L_\lambda) \tag{10}$$

for $(1-p) = q \to 0$. Therefore the higher the probability, the better is the a priori tour length L_λ as an approximation for the expected tour length $E[L_\lambda]$.

In the heterogeneous PTSP, it is not easy to see the relation between the two objective functions, since each arc of the a priori tour L_λ is multiplied by a different probabilistic weight (see eq.(3)), and a term with L_λ cannot be isolated in the expression of $E[L_\lambda]$, as in the homogeneous case.

ACS and pACS also differ in time complexity. In both algorithms one iteration (i.e., one cycle through the *while* condition of Fig. 2) is $O(n^2)$ [11], but the constant of proportionality is bigger in pACS than in ACS. To see this one should consider the procedure *UpdateTrail* of Fig. 2, where the best-so-far tour must be evaluated in order to choose the arcs on which pheromone is to be updated. The evaluation of the best-so-far tour requires $O(n)$ time in ACS and $O(n^2)$ time in pACS. ACS is thus faster and always performs more iterations than pACS for a fixed CPU time.

4 Experimental Tests

4.1 Homogeneous PTSP Instances

Homogeneous PTSP instances were generated starting from TSP instances and assigning to each customer a probability p of requiring a visit. TSP instances were taken from two benchmarks. The first is the TSPLIB at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/. From this benchmark we considered instances with a number of city between 50 and 200. The second benchmark is a group of instances where customers are randomly distributed on the square $[0, 10^6]$. Both uniform and clustered distributions where considered in this case, and the number of cities varied between 50 and 350. For generating random instances we used the Instance Generator Code of the 8th DIMACS Implementation Challenge at http://research.att.com/dsj/chtsp/download.html.

4.2 Computational Environment and ACS Parameters

Experiments were run on a Pentium Xeon, 1GB of RAM, 1.7 GHz processor. In order to asses the relative performance of ACS versus pACS independently from the details of the settings, the two algorithms were run with the same parameters. We chose the same settings which yielded good performance in earlier studies with ACS on the TSP [10]: $m = 10$, $\beta = 2$, $q_0 = 0.98$, $\alpha = \rho = 0.1$ and $\tau_0 = 1/(n \cdot Obj)$, where n is the number of customers and Obj is the value of the objective function evaluated with the nearest neighbor heuristic [10].

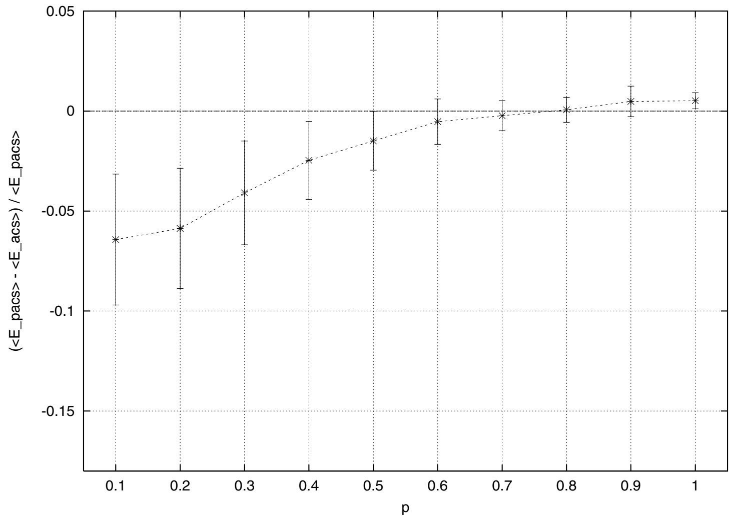

Fig. 3. Relative performance of pACS versus ACS for the homogeneous PTSP. The vertical axis represents $(E[L_\lambda(pACS)] - E[L_\lambda(ACS)])/E[L_\lambda(pACS)]$. On the horizontal axis there is the customer probability p. Each point of the graph is an average over 21 symmetric homogeneous PTSP instances. Error bars represent average deviation, defined as $\sum_{i=1}^{n} |x_i - <x> |/n$, with $n = 21$. Note that for $p = 1$ ACS outperforms pACS, since for a fixed CPU stopping time ACS makes more iterations.

The stopping criterion used in both algorithms is CPU time in seconds, chosen according to the relation *stoptime* $= k \cdot n^2$, with $k = 0.01$. This value of k lets ACS perform at least $17 \cdot 10^3$ iterations on problems with up to 100 customers, and at least $15 \cdot 10^3$ iterations on problems with more than 100 customers. For each instance of the PTSP, we ran 5 independent runs of the algorithms.

4.3 Results

For each TSP instance tested, nine experiments were done varying the value of the customer probability p from 0.1 to 0.9 with a 0.1 interval. Fig. 3 summarizes results obtained on 21 symmetric PTSP instances, one third of the instances were taken from the TSPLIB, the others were random instances (half of them uniform and half clustered). The figure shows the relative performance of pACS versus ACS, averaged over the tested instances. A typical result for a single instance is reported in Fig. 4.

As it was reasonable to expect, for small enough probabilities pACS outperforms ACS. In all problems we tested though, there is a range of probabilities $[p_0, 1]$ for which ACS outperforms pACS. The critical probability p_0 at which this happens depends on the problem.

The reason why pACS does not always perform better than ACS is clear if we consider two aspects: the complexity (speed) of ACS versus pACS, and

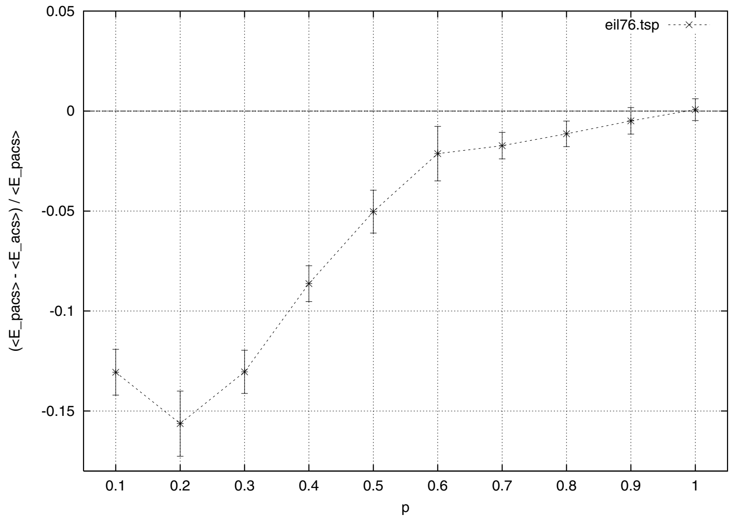

Fig. 4. Relative performance of pACS versus ACS for the eil76.tsp instance of the TSPLIB. Error bars represent the average deviation of the result over 5 independent runs of the algorithms.

the goodness of the PTSP objective function approximation as p approaches 1. When p is near to 1, a good solution to the TSP is also a good solution to the PTSP; therefore, ACS, which performs more iterations than pACS, has a better chance to find a good solution.

4.4 Absolute Performance

For the PTSP instances we tested, the optimal solution is not known. Therefore, an absolute performance evaluation of the pACS heuristic can only be done against a theoretical lower bound, when this is available and tight enough. A lower bound to the optimal solution would give us an upper bound to the error performed by the pACS heuristic. In fact, if LB is the lower bound and $E[L_{\lambda^*}]$ is the optimal solution, then by definition we have

$$E[L_{\lambda^*}] \geq LB \ . \tag{11}$$

If the solution value of pACS is $E[L_\lambda]$, then the following inequality holds for the relative error

$$\frac{E[L_\lambda] - E[L_{\lambda^*}]}{E[L_{\lambda^*}]} \leq \frac{E[L_\lambda] - LB}{LB} \ . \tag{12}$$

For the homogeneous PTSP and for instances where the optimal length L_{TSP} of the corresponding TSP is known, it is possible to use the following lower bound to the optimal expected length [6]

$$LB = pL_{TSP}(1 - (1-p)^{n-1}) \ . \tag{13}$$

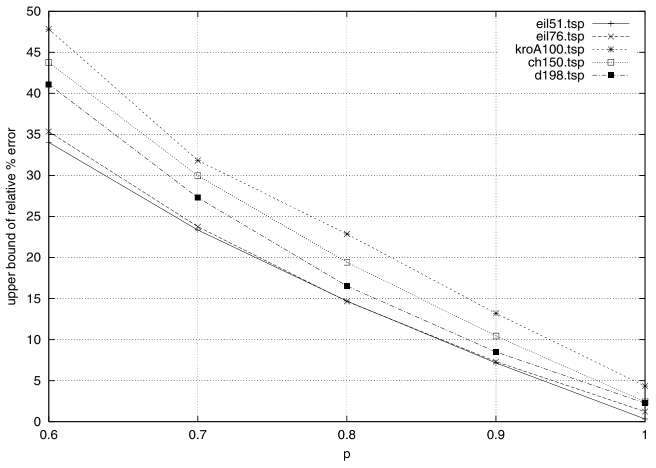

Fig. 5. Upper bound of relative percent error of pACS for 5 TSPLIB instances. The horizontal axis represents the customer probability.

If we put this lower bound into the right side of equation (12), we obtain an upper bound of the relative error of pACS. Fig. 5 shows the absolute performance of pACS, evaluated with this method, for a few TSPLIB instances. From the figure we see that, for instance, pACS finds a solution within 15% of the optimum for a homogeneous PTSP with customers probability 0.9.

5 Conclusions and Future Work

In this paper we investigated the potentialities of ACO algorithms for the PTSP. In particular, we have shown that the pACS algorithm is a promising heuristic for homogeneous TSP instances. Moreover, for customers probabilities close to 1, the ACS heuristic is a better alternative than pACS.

At present we are investigating the heterogeneous PTSP, for different probability configurations of customers. This is an interesting direction of research, since it is closer to a real-world problem than the homogeneous PTSP. We are also trying to improve pACS performance by inserting in the ants' tour construction criterion information about the customers probabilities. Work which will follow this paper also comprehends a comparison of pACS with respect to other PTSP algorithms. Moreover, pACS should be improved by adding to the tour construction phase a local search algorithm. The best choice and design of such a local search is also an interesting issue for the PTSP.

Acknowledgments

This research has been partially supported by the Swiss National Science Foundation project titled "On-line fleet management", grant 16R10FM, and by the

"Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this paper is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate.

References

1. D. J. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Operations Research*, 38:1019–1033, 1990.
2. D. J. Bertsimas. *Probabilistic Combinatorial Optimization Problems.* PhD thesis, MIT, Cambridge, MA, 1988.
3. P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, MIT, Cambridge, MA, 1985.
4. A. Jézéquel. *Probabilistic Vehicle Routing Problems.* Master's thesis, MIT, Cambridge, MA, 1985.
5. F. A. Rossi and I. Gavioli. *Aspects of Heuristic Methods in the Probabilistic Traveling Salesman Problem*, pages 214–227. World Scientific, Singapore, 1987.
6. D. J. Bertsimas and L. Howell. Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research*, 65:68–95, 1993.
7. G. Laporte, F. Louveaux, and H. Mercure. An exact solution for the a priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.
8. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
9. L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, Piscataway, NJ, 1996.
10. 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):53–66, 1997.
11. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.