

SOLVING THE PROFITABLE TOUR PROBLEM USING ANT COLONY
SYSTEM

by

Aykun Açıl

B.S., Mechanical Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

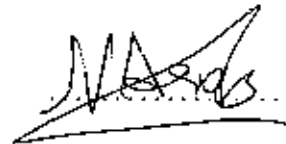
Graduate Program in Industrial Engineering
Boğaziçi University

2008

SOLVING THE PROFITABLE TOUR PROBLEM USING ANT COLONY
SYSTEM

APPROVED BY:

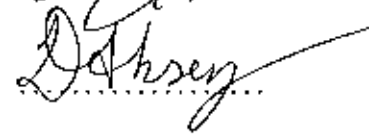
Assoc. Prof. Necati Aras
(Thesis Supervisor)



Asst. Prof. Tınaz Elkim Aşıcı



Asst. Prof. Deniz Aksen



DATE OF APPROVAL: 16.06.2008

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Assoc. Prof. Necati Aras for his great support, encouragement, advise and patience throughout this work.

I am grateful to Asst. Prof. Tınaz Ekim Aşıcı for her guidance, comments and interest on the committee.

I would like to thank to Asst. Prof. Deniz Aksen for his interest, comments and willingness to serve as my committee member.

I would like to thank to all members of Boğaziçi University Industrial Engineering Department for their great support. Honestly, I have to say that I am proud of being a member of this big family.

I also thank to my parents Ayşe Ağıl and Coşkun Ağıl for their trust, patience, support and love.

Finally all my grateful thanks and all my love goes to my cousins Derya and Seda and my best friends for their moral support.

This research has been partly supported by Boğaziçi University Research Fund Grant No: 06HA303

ABSTRACT

SOLVING THE PROFITABLE TOUR PROBLEM USING
ANT COLONY SYSTEM

The Traveling Salesman Problem (TSP) is one of the most widely studied combinatorial optimization problems. Many heuristic algorithms such as tabu search, genetic algorithm, and simulated annealing are applicable to this problem. There are some extensions of the TSP such as the Profitable Tour Problem (PTP), the Orienteering Problem (OP) and the Prize-Collecting TSP (PCTSP). The PTP includes a different objective function than the TSP where the objective is to maximize the profit while minimizing the traveling cost. Hence, it is not an obligation to visit all of the cities.

In this thesis, a hybrid version of the ACS is used to solve the PTP for the first time in the literature. A local search model which includes inversion, insertion, double insertion, extraction and double extraction procedures is proposed. The ACS algorithm is adjusted to be used in the PTP. Four different strategies are presented in the paper and their results are compared with the optimal solutions found by using the Cplex solver. Results show that the PTP can efficiently be solved by the ACS algorithm.

ÖZET

KÂRLI TUR PROBLEMİNİN KARINCA KOLONİSİ ALGORİTMASIYLA ÇÖZÜMÜ

Gezgin satıcı problemi üzerinde en çok çalışılan kombinatoriyel optimizasyon problemlerinden biridir. Tabu arama, Genetik Algoritma ve Yapay ısı işlem algoritmaları gibi bir çok algoritma bu probleme uygulanabilmektedir. Gezgin satıcı probleminin Kârlı Tur Problemi, Yön Bulma Problemi ve Ödül Toplayan Gezgin satıcı problemi gibi uzantıları vardır. Kârlı Tur Probleminin Gezgin satıcı probleminden farklı bir amaç fonksiyonu vardır. Kârlı Tur Probleminde amaç kârı maksimize ederken yol maliyetlerini minimize etmektir. Bu sebeple tüm şehirleri gezme zorunluluğu yoktur. Karınca kolonisi algoritmaları Kârlı Tur Problemini çözebileceği halde bugüne kadar uygulanmamıştır. Bu tezde, Hibrid Karınca Koloni Sistemi algoritması Kârlı Tur Problemini çözmek için kullanılmıştır. Yerel arama modeli olarak şehir çıkarma, şehir ekleme, çift şehir çıkarma ve çift şehir ekleme prosedürleri önerilmiştir. Karınca Koloni Sistemi algoritması Kârlı Tur Probleminde kullanabilmek için uyarlanmıştır. Bu tezde dört farklı strateji sunulmuş ve sonuçları Cplex çözücüsü tarafından bulunan optimal çözümlerle kıyaslanmıştır. Sonuçlar Kârlı Tur Probleminin Karınca Koloni Sistemi algoritmasıyla çözülebildiğini göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF SYMBOLS/ABBREVIATIONS	xii
1. INTRODUCTION	1
2. LITERATURE REVIEW	3
2.1. Ant Colony Algorithms	3
2.1.1. Ant System	4
2.1.2. Ant Density	5
2.1.3. Ant Quantity	5
2.1.4. Rank Based Ant System	6
2.1.5. Max-Min Ant system	6
2.1.6. Global Ant Colony Algorithm	7
2.1.7. Ant Colony System	8
2.2. Profitable Tour Problem	12
2.3. Variants of TSPP	15
2.3.1. Orienteering Problem	15
2.3.2. Prize Collecting Traveling Salesman Problem	18
3. SOLUTION PROCEDURE	20
3.1. Adapting ACS to Solve the PTP: Changing Transition Rule and Adding Inversion Procedure	20
3.2. Incorporating Deletion and Insertion Procedures	23
3.3. Incorporating Double Deletion and Double Insertion Procedures	25
3.4. Changing Global Pheromone Update Rule	27
4. COMPUTATIONAL RESULTS	29
4.1. Experimental Set up	29
4.2. Results	31

5. CONCLUSIONS and RECOMMENDATIONS FOR FUTURE RESEARCH	46
5.1. Conclusion	46
5.2. Recommendations and Future Work	47
APPENDIX A: PSEUDOCODE OF ACS-PTP3	48
APPENDIX B: Coordinates of the Instances	54
APPENDIX C: Example Solution	80
REFERENCES	81

LIST OF FIGURES

Figure C.1. Optimal solution of 11 th instance	80
---	----

LIST OF TABLES

Table 4.1.	Data set properties and Cplex results	30
Table 4.2.	Results of ACS-PTP1	33
Table 4.3.	Results of ACS-PTP2	34
Table 4.4.	Results of ACS-PTP3	36
Table 4.5.	Results of HACS-PTP	37
Table 4.6.	Results of HACS-PTP with different T_{max} values	39
Table 4.7.	Results of HACS-PTP with different cl values	40
Table 4.8.	Results of HACS-PTP with forming candidate lists with increasing distance	41
Table 4.9.	Results of HACS-PTP by placing ants randomly in each iteration .	43
Table 4.10.	Results of HACS-PTP with different transition rules	44
Table 4.11.	Results of HACS-PTP with different global pheromone update formula	45
Table B.1.	Coordinates of 24 th instance	54
Table B.2.	Coordinates of 23 rd instance	55
Table B.3.	Coordinates of 22 nd instance	56

Table B.4.	Coordinates of 21 st instance	57
Table B.5.	Coordinates of 20 th instance	58
Table B.6.	Coordinates of 19 th instance	59
Table B.7.	Coordinates of 18 th instance	60
Table B.8.	Coordinates of 17 th instance	61
Table B.9.	Coordinates of 16 th instance	62
Table B.10.	Coordinates of 15 th instance	63
Table B.11.	Coordinates of 14 th instance	64
Table B.12.	Coordinates of 13 rd instance	65
Table B.13.	Coordinates of 12 nd instance	66
Table B.14.	Coordinates of 11 st instance	67
Table B.15.	Coordinates of 10 th instance	68
Table B.16.	Coordinates of 9 th instance	69
Table B.17.	Coordinates of 8 th instance	70
Table B.18.	Coordinates of 7 th instance	71
Table B.19.	Coordinates of 6 th instance	72

Table B.20. Coordinates of 5 th instance	73
Table B.21. Coordinates of 4 th instance	74
Table B.22. Coordinates of 3 rd instance	75
Table B.23. Coordinates of 2 nd instance	76
Table B.24. Coordinates of 1 st instance	77
Table B.25. Optimal Tours of Instances 1-17	78
Table B.26. Optimal Tours of Instances 18-25	79

LIST OF SYMBOLS/ABBREVIATIONS

c_{ij}	The cost of traveling the edge between vertex i and vertex j
c	City index
c_{max}	Maximum amount of cost that can be paid
$Clan_i$	The i^{th} ant clan
D	The depot
E	set of edges
e	Edge
G	Graph
J_i^k	Candidate list of k^{th} ant when it is in vertex i
L^+	Length of the best tour
L_{nn}	The length of the tour given by the nearest neighborhood heuristic
n	The number of vertices in the data set
p	Profit of a city
p_{iM}^k	Probability of k^{th} ant moving from vertex i to vertex M
p_{min}	Minimum amount of profit that must be collected
q_0	Tunable parameter determining degree of exploration
q	Random variable between 0 and 1
S	Vertex subset
T_{max}	The number of iterations that the algorithm works
V	Set of vertices
v	Vertex
w	The weighting parameter of pheromone table of one clan
x	x coordinate of a city
x_{ij}	Variable used for indicating whether the edge between vertex i and vertex j is used in the tour
y	y coordinate of a city
y_i	Variable used for indicating whether vertex i is visited in the tour

α	Parameter which controls the weight of the profit for determining the city to visit
β	Parameter which controls the weight of the distance on determining which vertex to visit
γ	Profit to distance ratio
η_{iu}	The inverse of distance between edges i and u at iteration t , visibility
π_i	The profit that can be collected by visiting vertex i
Π_L	The net profit of tour L
ρ	Parameter which determines the amount of pheromone decay
$\tau_{\max}$	Maximum pheromone level
$\tau_{\min}$	Minimum pheromone level
τ_{iu}	The pheromone level distributed on the arc between vertices i and u at iteration t
τ_0	Initial pheromone level on all of the edges
ACO	Ant Colony Optimization
ACOMAC	Ant Colony Optimization with Multiple Ant Clans
ACS	Ant Colony System
AD	Ant Density
AET	Average Execution Time
Ant-Q	Ant Quantity
AS	Ant System
ATSP	Asymmetric Traveling Salesman Problem
GA	Genetic Algorithm
GACO	Global Ant Colony System
k-MST	the k Minimum Spanning Tree
LSP	Lot Sizing Problem
MMACS	a novel ant colony system for TSP
MMAS	Max-Min Ant System
MNP	Maximum Net Profit

OP	Orienteering Problem
PCTSP	Prize Collecting Traveling Salesman Problem
PDAP	Percent Deviation of Average Profit from the optimal objective value
PDSTAP	Percent Deviation of Standard Deviation of Average Profit from the optimal objective value
PTP	Profitable Tour Problem
RBAS	Rank based Ant system
TCTSP	Time Constraint Traveling Salesman Problem
TOP	Team Orienteering Problem
TSP	Traveling Salesman Problem
TSSP	Traveling Salesman Subset Problem
VND	Variable Neighborhood Descent

1. INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most widely studied combinatorial optimization problems. In the TSP, a salesman starts his trip from a depot and visits all the cities in his agenda only once. The objective is to minimize the total traveling cost. There are some extensions of the TSP in which visiting all customers is not necessary and revenue from visiting cities is also taken into account. These extensions of the TSP can be gathered under the name of Traveling Salesman Problem with Profits (TSPP).

In the Orienteering Problem (OP), the aim is to find a route which maximizes profit while travel costs do not exceed a predetermined value. The OP aims to find a route between specified points. In the Prize-Collecting TSP (PCTSP), the aim is to find a circuit which minimizes travel costs while the collected prize is not smaller than a predetermined value. The PCTSP includes a penalty function for the unvisited cities. In the Profitable Tour Problem (PTP), the aim is to find a tour that maximizes the profit which can be found by the total revenue minus the total traveling cost. Many algorithms such as tabu search, genetic algorithm, neural networks and simulated annealing are applicable on these problems.

Ant colony algorithms are applied to the TSP and the OP. Although ant colony algorithms can also be used to solve the PTP, there is no such study in the literature. Ant algorithms use positive feedback of past visits. There are several ant algorithms such as Ant Colony system, Ant System, Max-Min Ant System, Ant Quantity System and Rank Based Ant System.

In this thesis, Ant Colony System (ACS) is used to solve the PTP. In the ACS, pheromone trails are updated in two steps. In the first step, evaporation of the pheromone trails on each edge travelled by each ant is achieved. In the second step, only the ant achieving the best tour is globally updated. In other words, positive feedback is applied only on the best tour. The ACS is also applied on the Orienteering

Problem (Liang and Smith, 2006).

There are several reasons for selecting the ACS to apply on the PTP. First of all, it is adaptable to apply different extensions of the TSP. Moreover, the Ant Colony Optimization (ACO) algorithms are population based. Furthermore, hybridization is possible. In other words, the ACS can be combined with a local search strategy. Since the TSP with Profits is an NP-Hard problem, local search is very important to have an efficient algorithm. Positive feedback is also an important factor. Pheromone trails distributed on the edges between the cities represent the knowledge. Pheromone levels are determined by the ants leading to the best tours in each iteration. Tours are formed by ants which follow pheromone trails. ACS uses a candidate list strategy and there is a candidate list for each city that is formed from the preferred cities to be visited. This candidate list includes a predetermined number of cities and these cities are selected by a principle. In the TSP, these cities are ordered and selected by increasing distance. If there is no unvisited city in the candidate list, nearest possible city is added to the tour. However, this is not necessary in the PTP. Candidate list strategy can be also effective in solving the PTP.

In our model, we propose a modified ACS algorithm enforced with a local search which includes swapping, insertion, double insertion, deletion and double deletion procedures. This algorithm is called HACS-PTP (Hybrid Ant Colony System specialized to solve the Profitable Tour Problem).

The remainder of this thesis is organized as follows. Section 2 presents the preliminaries, literature review and a mathematical formulation of our model. The solution procedure is explained in Section 3. Section 4 contains the computational results. Finally, Section 5 provides a summary and concludes the paper.

2. LITERATURE REVIEW

2.1. Ant Colony Algorithms

Ants live in groups called colonies. They forage systematically as a colony. Most of the ant species have trail-laying trail-following behavior during the foraging. Each worker ant deposits pheromone trails, a chemical substance, on the paths they move. In other words, an ant is influenced to move towards a food source by the pheromone released by another ant.

Released pheromone warns other worker ants to follow a path. Pheromone level on a path is proportional to the number of ants which followed this path. Hence, ants are not interested in the length of the path, but the pheromone level on the path. (Bonabeau et al., 1999)

Some ant species like *Lasius Niger* also use instincts with trail-laying trail-following behavior (Beckers et al., 1992). On the other hand, *Linepithema Humile* can solve minimal spanning tree problem. Moreover, their different nests are interconnected. In other words, knowledge is shared between different colonies (Aron et al., 1990). These interconnected nests lead to the exploration of parallel implemented ACO algorithms (Randal and Lewis, 2002) and algorithms including multiple ant clans (Tsai et. al, 2004).

The ant colony algorithms use the following assumptions when solving the TSP or its variants.

1. Environment or data is assumed to be a bidimensional grid.
2. Each worker ant lays pheromone both on the way to food and when returning to the nest. Moreover, some part of the pheromone is evaporated at each iteration. This action is to prevent excess reinforcement on some tours and converging to a suboptimal tour in early iterations. It is expected that in the long run, ants

following shorter paths return faster. Hence, the number of ants following the shorter path increases gradually. This can be achieved only by evaporating some part of the pheromone laid on edges.

3. Ants return to the nest after collecting food.
4. Ants bring all collected food to the nest. They do not eat food on the road.
5. An ant can carry all the food or profit at once (for uncapacitated problems).
6. Artificial ants have some memory and they are not completely blind.
7. Time is assumed to be discrete.
8. Ants are not allowed to visit each point more than once.

Before moving forward, it is important to indicate that routing problems can be represented with graphs. In these problems, each city can be represented by a vertex (v); each arc between two connected cities can be represented by an edge (e). Hence, a graph $G = (V, E)$ can be constructed using edge set $E = \{e_1, \dots, e_n\}$ and vertex set $V = \{v_1, \dots, v_m\}$.

If there is an edge between all pairs of vertices in the tour, the graph is called complete. Moreover, a graph is undirected in the case that there is no distinction between two directions associated with two vertices. Hence, any output of a symmetric profitable tour problem can be represented by an undirected complete graph.

2.1.1. Ant System

The first ant colony algorithm is introduced by Dorigo (1992) in his Ph.D. thesis which is called Ant System (AS). AS is a new approach to stochastic combinatorial optimization problems. AS can also be called a special distributed reinforcement learning technique. It is initially applied to the TSP, and to the quadratic assignment problem (Dorigo et al., 1996a).

AS relies on the idea of positive feedback (Dorigo et al., 1991). It is a population based algorithm which includes pheromone updating and transition rules to get positive feedback to construct shorter tours. Pheromone update rules are applied on each edge

along the path between two cities when an ant moves along this edge. In other words, all of the ants lay pheromone on each edge which is proportional to the traveling cost. This gives rise to the situation ants traversing shorter paths release higher amounts of pheromone. Alaykiran and Engin (2005) compared the results obtained by AS with the exact solutions of the TSP. They found that results of AS is sensitive on parameters. Moreover, the percent errors between the AS results and exact results increase as the number of cities increases.

Elitist ant strategy is also proposed by Dorigo (1996a). In this strategy, elitist ants are added to the tour to reinforce the best solution so far. The idea behind this is to additionally update pheromone trails of the best tour found so far to increase the probability of other ants to follow this path. Elitist ants update pheromone levels on the edges of the best tour. Hence, the pheromone levels on the edges of the best tour are updated more than once. Experimental results showed that the number of elitist ants should not be very high to reach better tours. If an excess amount of the elitist ants are used, suboptimal solutions are obtained in early phases of the search. Some of the pheromone is evaporated to let the algorithm search the solution space better. AS is also applied on the Vehicle Routing Problem by Wade and Salhi (2001).

2.1.2. Ant Density

Ant Density (AD) algorithm also differs from AS by the way it updates pheromone trails (Dorigo et al., 1991). In the AD, the amount of pheromone update on each edge depends neither on the length of the edges nor on the total distance of the tour. A constant amount of the pheromone is laid on an edge each time an ant passes through this edge. In other words, the pheromone update rule of AD is different from the pheromone update rule of AS.

2.1.3. Ant Quantity

Ant Quantity (Ant-Q) is a variant of the AS algorithm (Gambardella and Dorigo, 1995). The Ant-Q is based on similarities with Q-learning (Watkins, 1992). Moreover,

AS can be thought of as a particular instance of the Ant-Q algorithm. Dorigo et al. (1995) proposed the Ant-Q to strengthen the connection between positive feedback and the AS algorithm. The basic difference between the Ant-Q and the AS is the pheromone trail updating rule. In the Ant-Q algorithm, the amount of pheromone level on each edge depends on the length of these edges. However, the amount of pheromone level on each edge depends on the total distance of the tour in AS. There are two important properties of the Ant-Q algorithm. In the Ant-Q, ants do not converge to the same tour. Moreover, pheromone levels can be exploited by the ants to find short tours. In most cases, the Ant-Q gives better results than heuristics such as Simulated Annealing, Elastic Net, Self Organizing Map and Farthest Insertion. However, the AD algorithm is superior to the Ant-Q when solving the TSP.

2.1.4. Rank Based Ant System

In the Rank Based Ant System (RBAS) algorithm, only a pre-determined amount of ants finding the best tours update pheromone trails and pheromone updating level is based on how good the tours are. Moreover, the RBAS is a revised version of AS with elitist strategy. In the RBAS algorithm, generated ants are sorted with respect to their tour lengths. Furthermore, contribution of ants to the pheromone updating process is weighted with respect to rank of the ants. Since only a predetermined amount of the ants update the pheromone levels, early convergence to sub-optimal paths is avoided. In other words, overemphasizing the pheromone levels is the result of the fact that many ants can use the same paths on early tours. Limiting the number of ants that contribute to the pheromone level updating prevents early convergence. Results show that the RBAS can compete with algorithms such as Simulated Annealing and Genetic Algorithm to solve the TSP. In fact, the RBAS gives better results for large instances (Bullnheimer et al., 1999).

2.1.5. Max-Min Ant system

Max-Min Ant system (MMAS) is another algorithm derived from the AS algorithm. There are basic differences between AS and MMAS. First, MMAS is greedier

than AS, because only the ant finding the best solution updates pheromone levels in each iteration. This can be the iteration-best ant or the global-best ant. Such an update mechanism leads to strong exploitation of the solution space. Also, a limiting mechanism on pheromone levels is applied to avoid premature convergence of the search. In other words, the pheromone levels have upper bounds and lower bounds on the edges to prevent local convergence. In addition to this, the pheromone trails are assigned to the maximum possible level to increase exploration capability of the virtual ants.

Stagnation or premature convergence of the algorithm depends not only on the way how pheromone trails are updated. Sometimes stagnation is caused by the search. This means that a tour can have arcs with dramatically higher pheromone levels. Stützle and Hoos (2000) propose a way to determine the convergence. If an arc between vertex i and vertex j has maximum pheromone level and all other arcs including vertex i has minimum pheromone level, one can say that MMAS has converged. On the other hand, the maximum amount of pheromone level dynamically changes if the global best ant changes.

MMAS is also used to solve the team orienteering problem (TOP) (Liangjun et al, 2008). In the TOP, an ant decides which vehicle to move and where to move. In other problems such as the OP or the TSP, an ant only decides where to move. Liangjun et. al (2008) propose an algorithm in which candidate solutions are formed. Then local search which includes 2-opt and insertion procedures is applied on candidate solutions until no improvement occurs. This is the same as local search proposed by Chao et al. (1996b). After the local search, pheromone levels on the edges which belong to the best solution are updated.

2.1.6. Global Ant Colony Algorithm

Global Ant Colony Algorithm (GACO) has some important differences from the other ACO algorithms stated above. In the GACO, ants are not restricted to visit all of the nodes. Furthermore, ants do not need to complete their tours. The GACO

algorithm does not include local pheromone update. Global pheromone update and ant's path selection are realized by considering complete path information. Moreover, GACO is used for Lot Sizing Problem (LSP) (Keskinturk and Soyler, 2006).

The transition rule in GACO is similar to existing ACO algorithms. However, there is a tour between two selected nodes instead of a path. Ants make tour selections probabilistically by considering the global pheromone levels and distances. Pheromone updating is done globally by all of the ants. In other words, pheromone updating is not only done by the ant leading to the best tour.

GACO also includes local search. Path mutations are applied on tours to prevent convergence to local minima or maxima. This method is similar to mutation on Genetic Algorithm (GA). Results showed that the GACO algorithm can be used as an alternative to solve the LSP.

2.1.7. Ant Colony System

Ant Colony System (ACS) is introduced by Dorigo and Gambardella (1996). ACS includes mainly four modifications over the AS algorithm. It has a different transition rule and a different pheromone trail update rule. Moreover, ants update pheromone trails at each time they move to new edges. This action is called local pheromone trail update. In addition to this, ants are restricted about where to move by forming candidate lists for all of the edges (Dorigo et al., 1997a).

Transition of an ant from edge i to edge j is determined by the following equation.

$$j = \begin{cases} \arg \max_{u \in J_i^k} \{ [\tau_{iu}(t)] [\eta_{iu}(t)]^\beta \} & \text{if } q \leq q_0; \\ M & \text{if } q > q_0 \end{cases} \quad (2.1)$$

where

- $\tau_{iu}(t)$: The pheromone level distributed on the arc between vertices i and u at iteration t ,
- $\eta_{iu}(t)$: Visibility, the inverse of the distance between edges i and u at iteration t ,
- β : Parameter which controls the weight of the distance on determining which vertex to visit,
- q_0 : Tunable parameter determining degree of exploration, $q_0 \in [0, 1]$,
- q : Random variable between 0 and 1.
- J_i^k : Candidate list of k^{th} ant when it is in vertex i

If $q > q_0$, ant k in vertex i selects M which is a vertex in the candidate list of vertex i with a probability;

$$p_{iM}^k(t) = \frac{[\tau_{iM}(t)] \cdot [\eta_{iM}(t)]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)] \cdot [\eta_{il}(t)]^\beta} \quad (2.2)$$

where

$p_{iM}^k(t)$: Probability of k^{th} ant moving from vertex i to vertex M .

The first part of equation (2.1) uses knowledge about the problem. However, the second part encourages ants to explore other solutions in the search space. Hence, increasing q_0 makes the algorithm concentrate on local optima. On the other hand, if the candidate list is empty for a given vertex, ants move to the closest vertex existing in the instance.

Pheromone trail updates are performed both globally and locally. Global pheromone update is done by the ant leading to the best tour found so far after an iteration. Global pheromone update rule is defined as:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \frac{\rho}{L^*} \quad (2.3)$$

where

$\tau_{ij}(t)$: Pheromone level on the edge between vertex i and vertex j at iteration t ,

ρ : Parameter which determines the amount of pheromone decay, Global

L^+ : Length of the best tour.

pheromone update rule directs ants to explore the solutions close to the best tour. Local update of pheromone trails is performed by an ant that is in vertex i and selects vertex j to move. Local pheromone update rule is given as:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0 \quad (2.4)$$

where τ_0 is the initial value of the pheromone level on all edges.

In ACS, an initial amount of pheromone is laid on the edges before building up the tours. For the TSP, there is a way to determine the initial value of the pheromone levels on the edges. Setting $\tau_0 = (nL_{nn})^{-1}$ where L_{nn} is the length of the tour given by the nearest neighborhood heuristic gives good results. (Dorigo and Gambardella, 1996b; Dorigo et al., 1995)

Local update decreases the pheromone levels on edges while global update increases the pheromone levels on edges. In other words, local update helps ants to explore different paths while global update leads to a directed exploration of the solution space.

In ACS, the initial pheromone level is assigned to a value which can be considered as a minimum value, since ACS is expected to give shorter tours than the nearest neighborhood heuristic. However, the initial pheromone level is assigned to the maximum possible value in MMAS.

Candidate lists include preferred vertices to move. Every vertex has its own candidate list. Vertices in these lists are examined first. For the TSP, if it is impossible to visit one of the vertices in the candidate list without subtour formation, ACS looks other vertices not in the candidate list and chooses the closest one to move. In other

words, ants only choose other vertices outside of the candidate list if all vertices in the candidate list are already visited. (Dorigo et al., 1997a)

Tsai et. al (2004) proposed a meta-heuristic called Ant colony optimization with multiple ant clans (ACOMAC) to solve the TSP. In this method, different ant colonies are located in the same map and they can communicate with each other. This is similar to parallel genetic algorithm.

In this method, the ACS algorithm is performed a number of times for each ant colony. Subsequently, these colonies communicate with each other with respect to the equation below:

$$\begin{cases} Clan_i = Clan_k \cdot (1 - w) + Clan_i \cdot w & \text{if } i = 1; \\ Clan_i = Clan_{i-1} \cdot (1 - w) + Clan_i \cdot w & \text{otherwise} \end{cases} \quad (2.5)$$

where

$Clan_i$: The i^{th} ant clan,

w : The weighting parameter of pheromone table of one clan.

Tsai et. al (2004) propose two approaches to initialize pheromone levels. The first one is called Multiple Nearest Neighborhood approach. Nearest Neighborhood heuristic is started from different initial edges. Arcs used in building tours are accumulated in a table. Density of the initial pheromone level on arcs is determined by how many times these arcs are used in building tours.

The second approach is called Dual Nearest Neighborhood approach. In this approach, an edge is selected randomly as the first edge. The farthest edge to this edge is also determined. Nearest Neighborhood heuristic is applied on these two edges at the same time. However, an edge can be visited at most once. Finally, the graph should be incomplete. In other words, two routes should not be connected. Initial pheromone levels are formed by using these two routes. Randal and Lewis (2002) proposed a

similar method about parallel implementations of the ACO algorithms.

Dorigo and Blum (2005) discuss convergence, connections between the ACO algorithms and influence of search bias on the ACO algorithms. Botee and Bonabeau (1998) compare results of the ACS algorithm with the GA for solving the TSP. Dorigo et al. (2006) give a brief summary of the ACO algorithms, successful applications and open research topics.

A novel ant colony system for TSP (MMACS) with hybrid mutation based on minimum 1-tree is proposed to solve the TSP (Wang et. al, 2005). In this algorithm, they use MMACS with candidate list strategy in order to improve convergence of the algorithm and to reduce run times. However, Wang et al. (2005) form a candidate list strategy different from candidate list strategies based on nearest neighborhood (Dorigo and Gambardella, 1995). They form the candidate list strategy by using a genetic algorithm method (Yang et al., 2003).

Wang et. al (2006) also propose a local search method applied to current best tour. In this local search algorithm, they use swapping, inversion, insertion procedures.

2.2. Profitable Tour Problem

The Profitable Tour Problem (PTP) is a variant of the TSP. The PTP includes an objective function which aims to maximize the profit. In other words, the objective is to maximize revenue while minimizing the traveling cost. Hence, it is not an obligation to travel all of the cities (Dell'Amico et al., 1995). The formulation of the PTP is given

as follows:

$$\min \sum_{i \in V} \pi_i y_i - \sum_{i,j \in A} c_{ij} x_{ij} \quad (2.6)$$

subject to

$$\sum_{j \in V \setminus \{i\}} x_{ij} = y_i \quad \{i \in V\} \quad (2.7)$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = y_j \quad \{j \in V\} \quad (2.8)$$

$$\sum_{\substack{j \neq i \\ i,j \in S}} x_{ij} \leq |S| - 1 \quad (S \subset V, |S| \geq 2, D \notin S) \quad (2.9)$$

$$y_D = 1 \quad (2.10)$$

$$x_{ij} \in \{0, 1\} \quad (i, j) \in E \quad (2.11)$$

$$y_j \in \{0, 1\} \quad (j) \in V \quad (2.12)$$

where

- x_{ij} : 1 if the corresponding arc is used in the tour. 0, otherwise.
- y_i : 1 if vertex i is visited. 0, otherwise.
- c_{ij} : The cost of traveling the edge between vertex i and vertex j .
- S : Vertex subset,
- π_i : The revenue of vertex i .
- D : The depot

In the model, the first two constraints are assignment constraints. The third constraint is about eliminating the subtours, since each city can be visited at most once. The fourth constraint represents the fact that the depot should be included in the solution.

There are also variants of the PTP. One of them includes penalty in the objective function for unvisited cities. Another variant is Asymmetric Profitable Tour Problem in which the distance between city i to city j is different from the distance between city j to city i .

There are two approximations to solve the PTP. The first algorithm is proposed by Bienstock et al. (1993). They consider a simplified version of the Prize Collecting TSP to find a tour that visits a subset of cities. In this simplified problem, the aim is to minimize the length of the tour and the sum of penalties due to not visiting some of the cities. The minimum prize constraint is also omitted. This problem can be equivalent to the PTP if the penalties are omitted and each visited city has a revenue. Hence, this algorithm can be used to analyze symmetric PTP in which there is no distinction between two directions associated with two vertices. Bienstock et al. (1993) propose an algorithm which is based on Linear Programming relaxation with ellipsoid method. In this algorithm, the solution is transformed into a feasible integer solution and then Christofides' algorithm (1976) is applied. The best of generated solutions is selected and the algorithm stops. The second approximation algorithm proposed by Goemans and Williamson (1992) is combinatorial. The algorithm maintains different tours from which the best one is selected. In this algorithm, a greedy selection procedure is followed to add edges in each iteration.

An important property of the PTP is that it can be transformed into asymmetric TSP (ATSP). Volgenant and Jonker (1987) proposed a transformation scheme which helps to solve the PTP by using algorithms proposed to solve the ATSP. If the original problem includes n vertices, there are $2n + 1$ vertices in the transformed problem. Let $V = \{v_1, \dots, v_n\}$ is a set of vertices. The depot is duplicated as the $(n + 1)^{th}$ vertex. Vertices $v_{n+2}, \dots, v_{2n+1}$ are also added to duplicate vertices $v_2, \dots, v_{n+1}$. In the transformed problem, transformed vertex set $V' = \{v_1, \dots, v_{2n+1}\}$ is formed. The cost coefficients (distances) for the arcs between vertices of the original problem remain the same. Arcs from the original vertices to the duplicated ones have cost coefficients which are equal to the revenue of the original vertices. All other arcs have zero cost coefficients. After this transformation, the PTP can be solved with any algorithm designed for solving the ATSP. It is important to indicate that this transformation is not applicable to other TSP's with profits.

2.3. Variants of TSPP

Variants of TSPP is first grouped by Mittenthal and Noon (1992) under the name of Traveling Salesman Subset Problems (TSSP). In these problems, a feasible tour does not need to include every city in the instance. Orienteering Problem, Prize Collecting Traveling Salesman Problem, Time Constraint Traveling Salesman Problem and Profitable Tour Problem can be formulated as the TSSP.

Mittenthal and Noon (1992) proposes an insert/ delete heuristic to solve the TSSP's. In this heuristic, an initial tour is found. After that, in each step, a city is added to the tour until no improvement occurs. It is important to explain that the TSSP is about selecting and sequencing. Insertion is about both selecting and sequencing. However, deletion is about selecting. This heuristic can be used for both to improve the objective value and to decrease infeasibility for problems such as OP, PCTSP, TCTSP. Deletion is used by Kuehn and Hamburger (1963) for solving a facility location problem.

2.3.1. Orienteering Problem

The Orienteering Problem (OP) can be modeled with given n nodes that have a score or profit greater than zero, except the starting and the finishing nodes of the route. The formulation of the OP is given as follows:

$$\max \sum_{v_i \in V} \pi_i y_i \quad (2.13)$$

s. t.

$$(2.7) - (2.12) \quad (2.14)$$

$$\sum_{(v_i, v_j) \in A} c_{ij} x_{ij} \leq c_{max} \quad (2.15)$$

Equation (2.13) represents objective function of the OP. The time constraint which is given as constraint (2.15) is also included to restrict the total traveling cost not to exceed c_{max} . If the time constraint is relaxed and the starting and the finishing nodes of the route are not specified, the OP returns into a regular TSP. In other words, the Orienteering Problem is a variant of the TSP.

There are various algorithms for the OP. One of them is an artificial Hopfield neural network to solve the OP (Wang et al., 1995). The Hopfield neural network is formed as a matrix. In this matrix, rows are used for vertices and columns are used for positions in the route. In this model, Wang et al. (1995) used local search procedures which are route insertion and 2-opt procedures to improve the method.

Another metaheuristic to solve the OP is proposed by Chao et al. (1996a). It uses nearest neighborhood insertion procedure to obtain an initial solution. In this procedure, after a tour is built, the next tour is built with the remaining vertices. This step continues until any uncovered vertex remains. In other words, vertices are partitioned into a set of feasible tours. This part is repeated several times and different sets which include tours are formed. The set containing the tour with the highest profit is kept. In the second step of the algorithm, the record-to-record approach (Dueck, 1993) is applied. Local search procedures which are two-point exchange, one-point exchange and 2-opt are used in this step. In the two-point exchange procedure, two consecutive vertices of a tour are inserted to another tour of the same set. Similarly, vertex of a tour can be inserted to another tour of the same set. The record-to-record approach allows deterioration of the solution. However, the deterioration should not be greater than 10%. By using these procedures, the best tour can be improved and changes are done as soon as the highest score is improved. It is important to note that the algorithm does not focus on a single tour. It tries to find the best tour out of a set of tours. Solution is diversified by shifting some vertices from the best tour to the other tours after t number of iterations.

Tasgetiren and Smith (2000) propose a Genetic Algorithm to solve the OP. First of all, initial population is formed by considering distances and the time constraint. In

this step, algorithm probabilistically forms each individual of the population. After that two individuals are selected and they are used to generate offsprings through order-based crossover. After each generation of offsprings, the population size increases. Then, the local search is applied to selected individuals of the population. In the local search, insertion, deletion and replacing an on-tour vertex with an off-tour vertex procedures are applied. Also, a penalty function is used to the solutions by considering their infeasibility.

A Tabu Search algorithm is proposed to solve the OP (Ramesh and Brown, 1991). This algorithm has four main steps. The first one of these steps is the insertion step. In this step, a tour is formed by inserting vertices in pairs. As soon as double insertions become impossible, single insertion of vertices is performed until it is not possible to insert any vertex into the tour. In this step, the time constraint can be violated. The second step is the improvement step in which 2-opt and 3-opt operations are used. 3-opt is used if a threshold in the objective value is reached. After these improvements, the algorithm returns to the first step. In the third step, a vertex is deleted from the tour and the algorithm returns to the first step. For this cycle, Ramesh and Brown use a stopping criterion based on the improvement between iterations and the total number of iterations performed so far. When this stopping criterion is met, the fourth step is applied which is another insertion phase. To avoid cycling of the algorithm, a tabu list is used to store the tours. Fischetti et al. (1998) adapted the Tabu Search algorithm and use it inside Branch and Cut method to solve the OP.

The ACO approach is also used to solve the Orienteering Problem (ACS-OP) (Liang and Smith, 2006). Transition probabilities used in this algorithm are the same as the ACS-TSP and they are defined in equations (2.1)–(2.2). However, visibility factor is selected as π_j/c_{ij} in the ACS-OP while visibility factor is selected as $1/c_{ij}$ in the ACS-TSP. Another difference between the ACS-OP and the ACS-TSP is the initial value of pheromone levels. $\tau_0 = \frac{1}{c_{max}}$ is the initial value of pheromone level in the ACS-OP where c_{max} is the time constraint for the OP. After all ants complete their paths, a local search is applied. Liang and Smith (2006) use a penalty function for infeasible solutions. This penalty function is proposed by Ramalhinho and Serra (1998).

In the global pheromone updating rule, the total score of the best solution is used instead of $1/L^+$. Variable Neighborhood Descent (VND) which explores neighborhood of the solution is effectively used in the ACS-OP. (Mladenovic and Hansen, 1997) The VND includes insertion, deletion, swapping, interchanging, forward insertion and backward insertion procedures. These procedures help to decrease infeasibility and to increase the total score. Liang et al. (2002) compare the results of the ACS and Tabu Search for the OP. Results show that the ACS reaches competitive results with the Tabu Search.

Team Orienteering Problem (TOP) is a variant of the OP. In the TOP, the aim is to maximize profit by forming multiple tours with a team of vehicles without exceeding the time limits. A city can only be visited once. The TOP is an NP-hard problem. The TOP can be a model for many real time applications such as the service scheduling routing technicians (Tang and Miller-Hooks, 2005) or the game of team orienteering (Chao et al., 1996b). There is a version of the TOP in which the aim is to maximize the total collected profit by forming multiple tours with a team of vehicles. However, there are also time and capacity limits. An exact method for the TOP is proposed by Boussier et al. (2006). There are metaheuristics such as the tabu search and the variable neighborhood search that can be used to solve the TOP. Archetti et al. (2007) propose two tabu search algorithms and a variable neighborhood algorithm to solve the TOP. They also compare their algorithms with the exact method of Boussier et al. (2006).

2.3.2. Prize Collecting Traveling Salesman Problem

Prize Collecting Traveling Salesman Problem (PCTSP) is another variant of the TSP. The formulation of the PCTSP is given as follows:

$$\min \sum_{(v_i, v_j) \in A} c_{ij} x_{ij} \quad (2.16)$$

s. t.

$$(2.7) - (2.12) \tag{2.17}$$

$$\sum_{v \in V} p_i y_i \geq p_{\min} \tag{2.18}$$

Equation (2.16) represents the objective function. Constraint (2.18) is included to restrict the amount of collected profit not to be less than $p_{\min}$.

There are various heuristics for the PCTSP. For example, there is a proposed approximation algorithm which is derived originally from solving the k Minimum Spanning Tree problem (k -MST problem) (Awerbuch et al., 1998). The k -MST problem is about finding a tree that spans exactly k vertices and has a minimum total distance. Instead of pushing the algorithm to visit k vertices, Awerbuch et al. (1998) use a constraint to make a minimum profit and adapt the algorithm for solving the PCTSP. Dell'Amico et al. (1998) also proposed two heuristics for the PCTSP.

3. SOLUTION PROCEDURE

In this thesis, we focus on PTP, and try to solve it using an ACO algorithm. The reason is twofold. ACO algorithms are versatile and robust. It is easy to adapt them to varieties of the TSP. Moreover, ACO algorithms are population based. Hence, parallel implementations are possible by using the knowledge of positive feedback.

During the solution of the PTP, we follow a procedure which helps to modify the ACS algorithm to the PTP. First of all, meaning of the pheromone level on an arbitrary arc is redefined. Pheromone trails in the TSP direct ants to the shorter paths. However, pheromone trails in the PTP should direct ants to more profitable paths. Moreover, the heuristic decisions that an ant has to make during tour formation should be redefined with respect to the PTP. Hence, the transition rules given in equations (2.1)–(2.2) are redefined.

The results of many ACO applications to NP-hard combinatorial optimization problems show that local search increases the performance of ACO algorithms (Dorigo and Di Caro, 1999; Dorigo and Stützle, 2002). Hence, we also implement a local search using different procedures. The results of our algorithm with different local search strategies will be compared in the experimental results. Dorigo and Stützle (2004) discuss how to apply local search and advice to make global update after the local search. We also discuss the same issue for the PTP.

3.1. Adapting ACS to Solve the PTP: Changing Transition Rule and Adding Inversion Procedure

In the first step, we try to adapt the ACS to the PTP. First of all, pheromone levels on all of the edges are initialized by using the tour obtained by the Nearest Neighborhood algorithm as follows.

$$\tau_0 = (nL_{nn})^{-1} \quad (3.1)$$

Then all ants are placed randomly on the cities of the instance. The number of ants is equal to the number of cities in the instance. Hence it is possible that some cities may not be starting points for ants. In the next step, we determine the candidate list by using a measure called revenue to distance ratio (γ) for each city. For any city i , other cities are sorted in decreasing order of γ . Revenue in the numerator is the revenue associated with city j and distance in the denominator is the distance between city i and city j . The top 25% of the cities with largest γ values are selected and placed to the candidate list. In other words, the size of candidate list depends on the number of the cities in the instance.

In this strategy, the revenue is used as a parameter in the transition rules. Each ant starts its tour from its home city and selects which city to visit by using equations (3.2)–(3.3):

$$j = \begin{cases} \arg \max_{u \in J_i^k} \{ [\tau_{iu}(t)] [\pi_u]^\alpha [\eta_{iu}(t)]^\beta \} & \text{if } q \leq q_0 \\ M & \text{if } q > q_0 \end{cases} \quad (3.2)$$

where

- π_u : The revenue that can be collected by visiting city u ,
- α : The weight of the revenue for determining the city to visit.

If $q > q_0$, ant k in city i selects M which is a city in the candidate list of city i with a probability;

$$p_{iM}^k(t) = \frac{[\tau_{iM}(t)] \cdot [\pi_u]^\alpha \cdot [\eta_{iM}(t)]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)] \cdot [\pi_u]^\alpha \cdot [\eta_{il}(t)]^\beta} \quad (3.3)$$

After adding a city to the tour, the local pheromone update is done by using equation (2.4). If the ant reaches a city which does not have any unvisited city in its candidate list, it returns to the initial position and completes the tour. Then, the

ant stops adding new cities, except the home city. Home city is inserted to the best possible position in the tour. After all ants complete their tours, the ant following the most profitable path globally updates the pheromone trails according to equation (2.3). T_{max} iterations later, an inversion operation is applied to all of the tours in the algorithm. Inversion is a 2-opt method.

During the inversion procedure, some part of the tour is inverted. For example, there is a tour $L(a, b, c, \dots, i, d, \dots, e, j, k, \dots, y, z)$. Inversion procedure starts by inverting the chain between cities a and b . If tour is not improved, we invert each chain $(a - c, \dots, a - i, a - d, \dots, a - y)$ including city a . Then we start inversions by using city b $(b - c, \dots, b - i, b - d, \dots, b - y)$ and so on. If the chain between cities i and j is inverted, a new tour $L'(a, b, c, \dots, j, e, \dots, d, i, k, \dots, y, z)$ is obtained. If the profit of tour L' is greater than the profit of tour L , L is replaced by L' . The following equation is used to calculate the change in the profit.

$$\Pi_{L'} - \Pi_L = (c_{id} + c_{ej}) - (c_{js} + c_{di}) \quad (3.4)$$

where

Π_L : The profit of tour L

If any improvement occurs, the tour is updated. Otherwise, the tour remains the same. If the tour is updated, inversion procedure is not started from the beginning. In the above example, after replacing L with L' , the chain between cities j and k is inverted. In other words, inversion is applied on the remaining chains of the tour. Hence, inversion procedure is applied $LS = 10$ times from the beginning, since there may be new chains that may be profitable to invert.

After the inversion operation, the ant with the most profitable tour is stored and the algorithm is started again. Algorithm is run multiple times until a predetermined

number of runs (MR) is obtained. However, the initial pheromone levels are not changed in this run and ants are placed randomly on cities again. We call the solution approach adopted in this section ACS-PTP1.

3.2. Incorporating Deletion and Insertion Procedures

Solution methods for variants of the TSP show that local search is very effective to improve the solution quality. Hence, incorporating some local search procedures may be very effective.

The PTP is about selecting and sequencing cities. Candidate list formation is the only selection step in the above algorithm. Since the selection of a city from the candidate list is a probabilistic step, there is a probability of leaving a profitable city unvisited. Due to this fact, local search is necessary for our algorithm. Therefore, two local search procedures are incorporated in the algorithm defined in the first section. They are used after the inversion procedure. It is also possible that there may be cities in the tour which are not profitable. The main reason is that the number of cities in a candidate list of a city is a definite number. Hence, an ant adds a city to its tour, if the city in which the ant stays has any unvisited city in the candidate list. Ants can add cities even if these cities are not profitable to visit. Hence, deleting one city from the tour can also be profitable. By deleting a city from the tour, two edges of the tour are broken. Two neighbors of the deleted city is connected with a new edge. For example, there is a tour such that $L(a, \dots, i, k, j, \dots, z)$. If city k is deleted, then a new tour $L'(a, \dots, i, j, \dots, z)$ is obtained. If the profit of tour L' is greater than the profit of tour L , the ant follows new tour L' . The following equation is used to calculate the change in the profit by deleting city k from the tour.

$$\Pi_{L'} - \Pi_L = (c_{ik} + c_{kj}) - \pi_k - c_{ij} \quad (3.5)$$

During the deletion procedure, each visited city is deleted from the tour to see whether the profit is improved or not. If an improvement occurs in the profit, that city is deleted from the tour. This local search procedure is sequentially applied on each ant after the inversion method. It is important to note that a tour with a deleted city is another tour. Deletion of a city can make an on-tour city, which was profitable to visit before, unprofitable. After a city is deleted, deletion method is started from the city which precedes the deleted city. The deletion procedure may be thought of as selecting cities that are desired in the tour. After deletion, an insertion method is applied.

Selection of cities is done by the candidate list strategy. However, a local search method, namely insertion can be used to select unvisited cities that can increase the profit. The unvisited city is inserted by breaking an edge between two cities of the tour. By inserting an unvisited city, we create two new edges in the tour. For example, there is a tour $L(a, \dots, i, j, \dots, z)$. City k is inserted every possible location on the tour. If city k is inserted between cities i and j , then a new tour $L'(a, \dots, i, k, j, \dots, z)$ is obtained. If the profit of tour L' is greater than the profit of tour L , the algorithm stores the new tour L' . The following equation is used to calculate the change in the profit by inserting city k to the tour.

$$\Pi_{L'} - \Pi_L = \pi_k + c_{ij} - (c_{ik} + c_{kj}) \quad (3.6)$$

After inserting city k into each possible location, the algorithm compares the results of these insertions with the original tour L . If there are more than one edge in which inserting city k results in more profitable tour, the algorithm selects the most profitable insertion location. During the insertion procedure, each unvisited city is inserted into the edges of the tour to see whether the profit is improved or not. Insertion procedure may be thought of as both sequencing and selecting cities. It is important to note that insertion of a city can make an off-tour city which was unprofitable to visit

before, profitable. Also, insertion of a city can make an on-tour city, whose visit was profitable before, unprofitable. Due to this fact, deletion is started from the beginning after a city is inserted. Moreover, inversion can improve the tours after inserting or deleting a city. Hence local search is applied more than once (i.e. $LS = 10$) in the algorithm. If it is not possible to improve the result with deletion and insertion methods, the algorithm stores the ant with the most profitable tour and then starts again.

After the global pheromone update, local search can be applied on the ant with the largest profit. Since it is not guaranteed to get the best result by applying local search to a single ant, it is applied on all of the ants. We call the solution approach explained in this section ACS-PTP2.

3.3. Incorporating Double Deletion and Double Insertion Procedures

It is important to indicate that sometimes inserting an unvisited city to the tour is not profitable alone. However, it can be profitable if it is inserted with another unvisited city together. Also, deleting a visited city from the tour may not be profitable alone. However, it can be profitable if it is deleted with another visited city. Due to these facts, double insertion and double deletion is added to the algorithm to see whether it improves the performance or not. Also, increase in the execution time should be investigated.

There are more than one local search method used in the algorithm. This makes the sequence in which local search methods are applied important. We propose that after the ACS steps are completed, the inversion procedure is applied on each ant to make optimization on the tour length. In other words, the inversion method tries to minimize the tour length without deleting the on-tour cities or adding the off-tour cities. Inversion method minimizes the tour lengths like optimizing the TSP tours. Before trying to add or delete cities, the inversion method should be applied to make the tours more compact.

For a given subset of cities that are visited by an ant, deletion, double deletion, insertion, double insertion methods are applied after minimizing the tours with the inversion method. In double deletion procedure, city pairs on the tour are sequentially deleted and compared with the original tour. If any improvement occurs in the profit, double deletion operation is applied. While using this method, it is important to hold the depot in the tour, since it has zero profit. For example, there is a tour $L(a, \dots, i, k, l, j, \dots, z)$. If cities k and l are deleted, then a new tour $L'(a, \dots, i, j, \dots, z)$ is obtained. If the profit of L' is greater than the profit of L , the ant follows the new tour L' . If the profit is improved, the local search starts from the deletion procedure. The following equation is used to calculate the change in the profit by deleting cities k and l from the tour.

$$\Pi_{L'} - \Pi_L = (c_{ik} + c_{kl} + c_{lj}) - \pi_k - \pi_l - c_{ij} \quad (3.7)$$

Double deletion procedure is applied on each pair of cities if there is an edge between the cities. In other words, two cities of the tour which are visited sequentially are deleted from the tour to see whether the profit is improved or not. There is an exception about the depot. There are two pairs including the depot. For example, there is a tour $L(a, b, \dots, y, z)$. Two city pairs $a - b$ and $z - a$ are not deleted from the tour. Double deletion procedure is applied on each ant after the deletion procedure is completed.

In double insertion, cities that are not in the tour are inserted in pairs and the effect of this insertion on the profit is considered. Also, the same pair of cities is inserted with the inverse sequence and results of these two insertions are compared. If an improvement occurs in the profit, double insertion operation is applied. For example, there is a tour $L(a, \dots, i, j, \dots, z)$. If cities k and l are inserted between cities i and j , then two new tours $L'(a, \dots, i, k, l, j, \dots, z)$ and $L''(a, \dots, i, l, k, j, \dots, z)$ are obtained. If the profit of L' is greater than the profit of L and L'' , ant follows new tour L' . The

following equation is used to calculate the change in the profit by inserting cities k and l to the tour.

$$\Pi_{L'} - \Pi_L = \pi_k + \pi_l + c_{ij} - (c_{ik} + c_{kl} + c_{lj}) \quad (3.8)$$

During the double insertion procedure, each unvisited city pair is tried to be between all possible cities in the tour to see whether the profit is improved or not. If the tour is improved, the local search is continued with the deletion procedure.

As stated above, after making a size change in the tour by using a local search method, other methods should also be applied. In other words, deleting or adding cities can make unvisited cities profitable or visited cities unprofitable. Hence local search is applied more than once ($LS = 10$) in the algorithm. If it is not possible to improve the tour with deletion, double deletion, insertion and double insertion methods, the ant with the most profitable tour is retained. We call this solution approach ACS-PTP3.

3.4. Changing Global Pheromone Update Rule

Until this step, we try to adapt the ACS algorithm to solve the PTP. Also, we try to develop a local search including different procedures. However, using these procedures increases the time to find a reasonable solution. In this step, we propose a way to increase the efficiency of the algorithm.

In the local search, inversion, single deletion, double deletion, single insertion, double insertion procedures are applied one after the other. Then, the ant following the most profitable path, globally updates the pheromone trails according to the equation below:

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \frac{\rho \Pi}{(L^+)^2} \quad (3.9)$$

The basic difference between equations (2.3) and (3.9) is that equation (3.9) includes the profit in the numerator and total length is squared in the denominator. In the previous rule, if a decrease in the total distance is realized, the pheromone levels on each edge used by the ant with the best profit was increased after the global pheromone update. However, the profit can be increased by increasing the collected revenue. In other words, the PTP problem is not only about minimizing the distance. It is about maximizing the collected revenue while minimizing the distance. Hence, both the profit and the length of the tour is incorporated in the global pheromone update rule. After the global pheromone update, the algorithm stores the ant with the most profitable tour as the global best tour and goes on with the next iteration. In each iteration, the algorithm compares the best result of previous iterations. If any improvement occurs in the profit, the global best tour is updated. Finally, T_{max} iterations later, the algorithm outputs the global best tour.

In the above strategies, the algorithm is run for multiple times. Moreover, local search is applied more than once. After each run, the pheromone levels remain the same before starting to the next run. In other words, after the first run, the pheromone levels are not initiated with the method that is used for determining initial pheromone levels. Due to the fact that local search is applied after the ACS and some edges of the best tour are not globally updated. Hence, the local search is inserted into the ACS algorithm instead of making it after. This method is more efficient than the previous methods, since there is no multiple applications of local search and there is no multiple runs of the algorithm. We call this strategy HACS-PTP.

4. COMPUTATIONAL RESULTS

4.1. Experimental Set up

We generate 25 instances to apply our strategies. Table 4.1 shows the number of cities in each instance. Cplex 11 solver is used to find the optimal solutions. These solutions and execution times are also given in the same table.

The codes are written in Visual Basic 6.0. Runs are taken in a computer having 16 GB RAM and two Intel Xeon 3.16 Ghz quadcore processors. For each instance, five runs are taken with each heuristic. For each instance, the average value of the profit, the execution time, and the standard deviation of the profit are calculated over these five runs.

T_{max} is used as a termination criterion for the algorithm. For the first three heuristics, the algorithm is run for 5 times ($MR = 5$) and the local search is run for 10 times ($LS = 10$) starting from the inversion procedure. However, there are no multiple runs in HACS-PTP strategy. For the last three heuristics, if any improvement occurs during the local search, local search returns to insertion procedure of the same local search run.

Table 4.1. Data set properties and Cplex results

Instance Number	Number of cities	Optimal Objective Value	Execution Time (s)
1	40	16.12	14
2	40	26.01	14
3	40	8.8	4
4	45	38.3	29
5	45	53.20	26
6	45	20.27	61
7	50	50.75	50
8	50	74.90	37
9	50	49.09	202
10	55	84.96	69
11	55	60.08	366
12	60	90.84	152
13	60	81.88	230
14	65	90.84	225
15	65	88.67	115
16	70	120.93	784
17	70	101.10	784
18	75	163.50	93
19	75	133.82	277
20	80	150.43	3889
21	85	187.14	12867
22	90	205.28	34986
23	95	238.95	14522
24	100	268.00	28221
25	125	419.81-426.11	37431

4.2. Results

In this section, results of four heuristics are given. We used some abbreviations in tables which are explained below:

- AP : Average profit,
- STAP : Standard deviation of average profit,
- OOV : Optimal objective value
- PDAP : Percent deviation of average profit from the optimal objective value,
- PDSTAP : Percent deviation of standard deviation of the profit from the optimal objective value,
- AET : Average execution time over five runs.

PDAP is calculated as follows:

$$PDAP = \frac{(OOV-AP)}{OOV} \times 100 \quad (4.1)$$

PDSTAP is calculated as follows:

$$PDSTAP = \frac{(OOV-STAP)}{OOV} \times 100 \quad (4.2)$$

Unless otherwise stated, parameters used in the heuristics are given as:

	ACS-PTP1	ACS-PTP2	ACS-PTP3	HACS-PTP
T_{max}	10	10	10	10
cl	$n/4$	$n/4$	$n/4$	$n/4$
β	2	2	2	2
α	2	2	2	2
ρ	0.1	0.1	0.1	0.1
q_0	0.9	0.9	0.9	0.9
MR	5	5	5	1
LS	10	10	10	1

Table 4.2 shows the results of ACS-PTP1. If the results of ACS-PTP1 are compared with the optimal results, it is easy to see that ACS-PTP1 could not find the optimal solutions. Moreover, execution time increases with an increase in the number of cities. Results show that solution quality of this strategy increases with the increasing number of cities in the instance. For example, there are two runs in 3th instance, that ACS-PTP1 could not find a solution. By considering these results, one can say that ACS-PTP1 does not give good results and it is not sufficient to solve the PTP.

Table 4.2. Results of ACS-PTP1

#	ACS-PTP1		
	PDAP (%)	PDSDAP (%)	AET (s)
1	41.87	92.68	22
2	42.75	85.89	22
3	85.45	96.59	22
4	9.79	94.02	36
5	25.86	93.03	37
6	79.97	91.81	38
7	10.09	97.28	55
8	23.34	95.93	56
9	38.44	94.85	55
10	22.90	94.82	92
11	21.60	96.22	91
12	30.28	96.14	131
13	22.31	96.51	129
14	37.99	97.82	181
15	24.69	95.34	180
16	24.92	96.95	255
17	26.41	95.49	257
18	27.22	96.94	356
19	22.13	96.84	369
20	20.70	98.12	461
21	16.24	98.32	601
22	15.55	98.37	780
23	21.84	96.22	1096
24	11.03	98.46	1279
25	9.94	99.46	3557

Table 4.3. Results of ACS-PTP2

#	ACS-PTP2		
	PDAP (%)	PDSDAP (%)	AET (s)
1	1.80	95.97	23
2	1.81	95.92	23
3	24.77	81.70	23
4	0.00	100.00	38
5	0.13	95.41	40
6	6.51	99.90	38
7	0.00	100.00	60
8	1.55	96.86	62
9	1.41	97.58	57
10	2.84	96.98	100
11	0.52	99.28	95
12	2.23	99.24	141
13	3.05	98.22	134
14	2.47	99.16	193
15	2.58	97.81	184
16	4.47	99.89	271
17	2.41	99.76	252
18	5.78	98.18	386
19	2.21	98.03	360
20	0.75	99.24	471
21	2.00	98.85	607
22	3.16	98.95	776
23	2.03	99.50	1062
24	3.16	99.41	1327
25	2.27	99.22	3601

Table 4.3 shows the results ACS-PTP2. If we compare the results of ACS-PTP2 with the optimal objective values, one can easily see that ACS-PTP2 could find the optimal solutions for small instances. In two instances, ACS-PTP2 finds optimal solution

in each run. If the average execution times in Tables 4.2 and 4.3 are compared, it can be said that there is a slight increase in the execution time. Moreover, execution time increases with an increase in the number of cities. Results show that adding insertion and deletion procedures to the algorithm improves the solution quality.

Table 4.4 shows the results of ACS-PTP3. A comparison of the results of ACS-PTP3 with the optimal objective values indicates that ACS-PTP3 could find the optimal solutions for most of these sets. In seven instances, ACS-PTP3 finds the optimal solution in each run. However, the execution time increases with an increase in the number of cities. Results show that adding double insertion and double deletion procedures to the algorithm improves the solution quality. If average execution times in Tables 4.3 and 4.4 are compared, it can be said that there is a slight increase in the execution time.

The results of HACS-PTP is given in Table 4.5. If the results of HACS-PTP are compared with the optimal solutions, we can observe that HACS-PTP could find the optimal solutions for most of these sets. Moreover, HACS-PTP gives better results than the above heuristics in smaller execution times. Results show that adding local search procedures into the heuristics greatly improves the solution quality. A comparison of the results indicates that HACS-PTP is the best among the four heuristics.

Table 4.4. Results of ACS-PTP3

#	ACS-PTP3		
	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	31
2	0.00	100.00	29
3	0.00	100.00	30
4	0.00	100.00	51
5	0.00	100.00	44
6	0.10	99.90	47
7	0.00	100.00	75
8	0.00	100.00	66
9	0.06	100.00	70
10	1.35	98.12	106
11	0.25	99.42	113
12	1.85	98.86	157
13	0.26	99.65	158
14	0.89	98.80	227
15	0.45	99.29	222
16	1.54	98.68	303
17	0.55	98.76	310
18	5.79	99.83	418
19	1.67	98.48	437
20	0.57	99.58	560
21	0.79	99.73	707
22	1.29	99.95	876
23	2.30	99.61	1169
24	2.20	99.60	1424
25	1.24	99.17	3740

Table 4.5. Results of HACS-PTP

#	HACS-PTP		
	PDAP (%)	PDSAP (%)	AET (s)
1	0.00	100.00	7
2	0.00	100.00	6
3	0.00	100.00	7
4	0.00	100.00	10
5	0.00	100.00	10
6	0.00	100.00	10
7	0.00	100.00	17
8	0.00	100.00	14
9	0.00	100.00	15
10	0.00	100.00	22
11	0.00	100.00	25
12	0.00	100.00	34
13	0.13	99.71	34
14	0.51	99.31	46
15	0.12	99.73	47
16	1.69	98.92	64
17	1.01	98.54	66
18	1.79	97.07	84
19	0.55	98.76	93
20	0.64	99.69	121
21	0.74	99.85	153
22	0.84	98.80	189
23	0.75	99.76	237
24	0.51	98.85	269
25	0.93	99.70	767

Table 4.6 shows the results of HACS-PTP with different iteration numbers such as 20 and 30. Results show that increasing the number of iterations slightly increases the average profit. However, the average execution time dramatically increases. These results indicate that local search is powerful and the algorithm can find the optimal solution. Hence, increasing iteration number does not improve the net profit drastically.

Table 4.7 shows the results of HACS-PTP with candidate list sizes as $n/2$ and $n/8$. Results show that increasing number of cities in the candidate list increases the average execution time. If we compare the results given in Tables 4.5 and 4.7, we can say that $n/4$ yields better results than $n/2$ and $n/8$. Using $n/4$ as a candidate list size outperforms HACS-PTP with $n/2$. However, candidate list sizes of $n/4$ and $n/8$ provide similar results except 9th and 18th instances. Using $n/8$ as candidate list size leads better average net profit than using $n/4$ as candidate list size in two instances. These results indicate that an excess amount of cities in candidate lists causes ants follow longer paths even if it is not profitable. Hence, candidate list size should not be too large.

In all of the above strategies, the candidate list of each city is formed by using γ . The first cl amount of these cities are selected to the candidate list. Results of HACS-PTP strategy by determining candidate lists with distance is given in Table 4.8. If results in Table 4.5 and Table 4.8 are compared, one can easily see that using parameter γ leads to better or same results in each instance.

Table 4.6. Results of HACS-PTP with different T_{max} values

#	$T_{max} = 20$			$T_{max} = 30$		
	PDAP (%)	PDSDAP (%)	AET (s)	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	13	0.00	100.00	20
2	0.00	100.00	13	0.00	100.00	19
3	0.00	100.00	13	0.00	100.00	19
4	0.00	100.00	20	0.00	100.00	30
5	0.00	100.00	20	0.00	100.00	29
6	0.00	100.00	20	0.00	100.00	30
7	0.00	100.00	30	0.00	100.00	47
8	0.00	100.00	29	0.00	100.00	41
9	0.00	100.00	29	0.00	100.00	43
10	0.00	100.00	44	0.00	100.00	67
11	0.00	100.00	45	0.00	100.00	68
12	0.00	100.00	64	0.00	100.00	96
13	0.00	100.00	68	0.00	100.00	97
14	0.02	99.98	88	0.00	100.00	135
15	0.00	100.00	92	0.00	100.00	137
16	1.03	99.06	119	0.73	98.92	183
17	0.00	100.00	121	0.00	100.00	188
18	1.57	97.74	169	0.00	100.00	254
19	0.07	99.85	173	0.00	100.00	263
20	0.62	99.65	223	0.53	99.65	340
21	0.52	100.00	291	0.09	99.67	437
22	0.42	99.45	360	0.23	99.57	545
23	0.41	99.75	477	0.24	99.72	723
24	0.00	100.00	596	0.00	100.00	870
25	0.58	99.62	1627	0.35	99.71	2409

Table 4.7. Results of HACS-PTP with different cl values

#	$cl = n/8$			$cl = n/2$		
	PDAP (%)	PDSDAP (%)	AET (s)	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	4	0.06	99.94	15
2	0.00	100.00	4	0.00	100.00	14
3	0.11	99.77	3	0.00	100.00	15
4	0.00	100.00	6	0.00	100.00	25
5	0.00	100.00	5	0.00	100.00	25
6	0.05	99.90	5	0.00	100.00	26
7	0.00	100.00	9	0.00	100.00	71
8	1.50	97.94	9	0.40	99.11	37
9	2.97	94.81	9	0.00	99.98	36
10	1.68	98.47	12	0.35	99.21	59
11	1.00	99.07	13	0.00	100.00	56
12	0.06	99.92	17	0.50	99.35	78
13	0.15	99.68	17	0.27	99.63	83
14	0.84	98.78	23	2.49	98.90	117
15	0.00	100.00	25	0.12	99.73	124
16	2.46	99.88	31	0.48	99.40	168
17	0.71	99.02	31	0.00	100.00	172
18	6.75	100.00	42	2.13	97.96	238
19	0.55	98.76	42	0.46	99.14	245
20	0.51	99.67	58	1.35	99.22	298
21	0.74	99.85	72	1.33	99.06	390
22	0.72	99.80	93	2.13	99.19	504
23	1.02	99.58	119	2.29	98.61	662
24	0.78	99.51	136	2.06	99.10	848
25	0.82	99.45	391	2.48	99.36	2271

Table 4.8. Results of HACS-PTP with forming candidate lists with increasing distance

#	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	6
2	0.00	100.00	6
3	0.11	99.89	6
4	0.00	100.00	10
5	0.00	100.00	9
6	0.00	100.00	10
7	0.00	100.00	14
8	0.00	100.00	13
9	0.00	100.00	14
10	0.35	99.21	21
11	0.00	99.98	22
12	0.56	99.23	31
13	0.11	99.76	31
14	0.02	99.98	44
15	0.12	99.73	44
16	0.06	99.92	58
17	1.45	98.61	60
18	4.22	97.37	80
19	0.00	100.00	84
20	1.42	98.71	109
21	2.65	97.47	139
22	3.93	97.03	174
23	2.70	98.56	229
24	4.77	99.00	275
25	2.17	99.08	751

In the first three heuristics, we use multiple runs of the same heuristic. In these runs, ants are randomly placed on cities in each run. However, they are placed on the same locations in each iteration. To see the effect of randomization in each iteration, we used HACS-PTP. This time ants are randomly placed on cities at each iteration.

Results of HACS-PTP by randomly placing ants on the cities in each iteration are summarized in Table 4.9. If results in Table 4.5 and Table 4.9 are compared, we can see that randomly placing ants in each iteration gives better results in only three instances. However, the case in which ants start their tours from the same position at each iteration gives better results than randomly placing ants in each iteration in ten instances. Hence, randomly placing ants on the cities in each iteration does not lead better results.

In all of the above strategies, transition rule defined in equations (3.2) and (3.3) is used instead of equations defined in equations (2.1) and (2.3). Two differences between these equations are π variable and α parameter. To see the effect of the profit term, α parameter is adjusted as 0 and 3 and runs are taken by using HACS-PTP. If results in Table 4.5 and Table 4.10 are compared, HACS-PTP gives better results when $\alpha = 0$ or $\alpha = 2$. Increasing the weight of the revenue more than 2 does not lead to good results. However, α values 0 and 2 lead to similar results. α value of 2 leads slightly better average values in ten instances, while $\alpha = 2$ gives slightly better average values in six instances. After comparing the results, one can say that adding profit parameter to the transition rule slightly increases the average profit. This modification is proposed by Liang and Smith (2006) to solve the OP. Results show that it can be used in the PTP as well.

HACS-PTP strategy has a different global pheromone update rule which is defined in equation (3.9) than the global pheromone update rule defined in equation (2.3) used in the first three strategies. Results of HACS-PTP strategy with global pheromone update rule which is defined in equation (2.4) are given in Table 4.11. If results in Table 4.5 and Table 4.11 are compared, it can be observed that our modification improves the average profit.

Table 4.9. Results of HACS-PTP by placing ants randomly in each iteration

#	PDAP (%)	PDSAP (%)	AET (s)
1	0.00	100.00	7
2	0.00	100.00	7
3	0.23	99.77	7
4	0.00	100.00	11
5	0.00	100.00	11
6	0.15	100.00	11
7	0.00	100.00	17
8	0.00	100.00	15
9	0.00	100.00	16
10	0.00	100.00	25
11	0.00	100.00	25
12	0.00	100.00	36
13	0.35	99.46	37
14	0.51	99.31	52
15	0.12	99.73	52
16	1.05	98.81	69
17	0.00	100.00	71
18	4.43	97.30	96
19	0.67	98.50	101
20	0.41	99.61	131
21	1.14	99.47	168
22	2.30	98.74	210
23	1.59	99.43	282
24	0.89	99.40	332
25	1.58	99.24	905

Table 4.10. Results of HACS-PTP with different transition rules

#	$\alpha = 0$			$\alpha = 3$		
	PDAP (%)	PDSDAP (%)	AET (s)	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	7	0.00	100.00	7
2	0.00	100.00	6	0.00	100.00	7
3	0.00	100.00	6	0.23	99.77	7
4	0.00	100.00	10	0.00	100.00	11
5	0.00	100.00	10	0.00	100.00	10
6	0.00	100.00	10	0.00	100.00	11
7	0.00	100.00	16	0.00	100.00	17
8	0.40	99.11	15	0.00	100.00	15
9	0.00	100.00	15	0.02	99.96	17
10	1.06	99.03	24	1.68	98.47	25
11	0.00	100.00	25	0.00	100.00	26
12	0.62	99.16	35	0.29	99.37	37
13	0.24	99.46	42	0.11	99.76	38
14	0.33	99.27	50	2.26	98.75	54
15	0.26	99.65	50	0.00	100.00	51
16	0.69	99.43	66	2.77	98.29	71
17	1.80	98.31	68	1.25	98.84	70
18	0.43	99.42	93	7.09	97.36	97
19	0.00	100.00	96	0.13	99.82	104
20	0.64	99.41	125	1.08	99.51	130
21	0.22	99.86	161	1.16	99.06	168
22	0.42	99.49	205	2.75	98.86	211
23	0.94	99.48	274	3.48	99.01	279
24	0.62	99.37	334	2.53	99.29	342
25	1.06	99.52	914	3.04	99.15	919

Table 4.11. Results of HACS-PTP with different global pheromone update formula

#	PDAP (%)	PDSDAP (%)	AET (s)
1	0.00	100.00	6
2	0.00	100.00	7
3	0.11	99.89	7
4	0.00	100.00	11
5	0.00	100.00	10
6	0.05	99.90	11
7	0.00	100.00	17
8	0.00	100.00	15
9	0.02	99.96	17
10	0.00	100.00	25
11	0.00	100.00	26
12	0.02	99.98	36
13	0.22	99.77	37
14	0.29	99.37	51
15	0.10	99.77	52
16	2.88	99.06	69
17	0.00	100.00	70
18	6.14	99.41	98
19	2.59	99.75	100
20	0.76	99.96	129
21	0.98	99.64	166
22	2.93	99.58	209
23	2.58	99.18	279
24	2.60	98.61	350
25	2.25	99.55	939

5. CONCLUSIONS and RECOMMENDATIONS FOR FUTURE RESEARCH

5.1. Conclusion

In this paper, the Profitable Tour Problem is solved by using the ACS algorithm. The classical ACS algorithm used to solve the Traveling Salesman Problem has to be modified. In other words, adaptations for the PTP are required. Only initializing pheromone levels and local pheromone updating rules remain the same as in the ACS-TSP.

In the first strategy, candidate list selection rule and transition rules are modified. Another difference between ACS-PTP1 and ACS-TSP is that ACS-PTP1 algorithm stops adding off-tour cities if the candidate list is empty for the city on which ant stays. After a predetermined number of iterations, inversion procedure is also applied to all of the ants. In the second strategy, in addition to the modifications presented in the first strategy, insertion and deletion procedures are applied to all of the ants after T_{max} iterations. In the third strategy, in addition to insertion and deletion procedures, double insertion and double deletion procedures are applied to all of the ants. In these three strategies, both the algorithm and the local search are run multiple times. Moreover, changes which are made on the tours during the local search do not affect the global pheromone update in these three strategies. In other words, some part of the knowledge is lost and local search does not have an effect on positive feedback. In the fourth strategy, all of the local search procedures which are presented in the third strategy are embedded into the algorithm. Moreover, the global pheromone update rule is modified. After ants complete their tours, global pheromone update rule is applied by considering the ant gathering the best net profit in this iteration. In other words, changes which are made on the tours during the local search, affect global pheromone updates in the fourth strategy.

Results show that fourth strategy which is the hybrid ACS algorithm is time

efficient and it gives better results. This strategy gets the optimal results for the known exact solutions. Moreover, selecting a candidate list size affect both the execution time and solution quality. Also, adding profit parameter to the transition rules and global pheromone update rule affects the solution quality. If ants are randomly distributed on the cities at each of iterations, some of the knowledge gained by pheromone updates is not used. Hence, results of such cases are not so good.

5.2. Recommendations and Future Work

PTP is a TSP with profit. Hence, some modifications presented in this paper can be also used in other TSP's with profit. On the other hand, off-tour cities added by local search do not locally update pheromone trails. This action prevents pheromone evaporation for the edges that are added to the tour with local search and it may accelerate the convergence of the algorithm. However, our instances are not large enough to increase the number of iterations. For the problems which require more iterations, one can use limits for the pheromone levels as in MMAS.

APPENDIX A: PSEUDOCODE OF ACS-PTP3

NOTATION

L_{nn}	:	The length of tour found with the nearest neighborhood heuristic,
n	:	The number of the cities in the instance,
$Citycounter(k)$	:	The number of the cities ant k visits,
$otherscounter(k)$	:	The number of the cities ant k does not visit,
$TOUR(k)$	:	The tour created by ant k ,
$best_lstour(m)$	:	The best tour found until m^{th} iteration of local search,
$best_globaltour(m)$	:	The best tour found until m^{th} run of the algorithm,
MR	:	The number of times algorithm is restarted,
LS	:	The number of times local search is applied,
$OTHERS(k)$	:	Cities that are not visited by ant k .

PSEUDOCODE

```

CALL  $L_{nn}$ 
FOR each edge in the instance
    Initialize pheromone level with  $(n * L_{nn})^{-1}$ 
END FOR
//multiple runs
FOR  $MR = 1$  to 10
    FOR  $k = 1$  to  $n$ 
        Place ant  $k$  on a randomly chosen city in the instance
    END FOR
    FOR  $t = 1$  to  $T_{max}$ 
        FOR each city in the instance
            Sort the other cities in the instance with decreasing gamma ratio

```

```

    Form a candidate list by selecting first  $n/4$  amount of
    sorted cities
  END FOR
  FOR each ant  $k$ 
     $Citycounter(k) = 1$ 
    IF there is at least one unvisited city in the candidate
    list of ant  $k$  in city  $i$ 
      Apply the transition rule to move from city  $i$  to
      city  $j$ 
      After a transition from city  $i$  to city  $j$ , apply
      the local pheromone update rule
       $Citycounter(k) = Citycounter(k) + 1$ 
    ELSE IF
      Ant  $k$  completes  $TOUR(k)$  and returns its initial location
    END IF
  END FOR
  FOR each ant  $k$ 
    IF the depot is not visited in  $TOUR(k)$ 
      Insert the depot from the closest edge
       $Citycounter(k) = Citycounter(k) + 1$ 
    END IF
  END FOR
  FOR each ant  $k$ 
    Compute the profit of  $TOUR(k)$ 
  END FOR
  Find the best tour
  FOR every edge  $(i, j)$  the best tour
    apply global pheromone update
  END FOR
END FOR
FOR each ant  $k$ 

```

```

Store the cities which are out of  $TOUR(k)$  in  $OTHERS(k)$ 
 $otherscounter(k) = n - Citycounter(k)$ 
//multiple application of local search
FOR  $LS = 1$  to 10
  //inversion procedure
  FOR  $i = 1$  to  $Citycounter(k) - 1$ 
    FOR  $j = i + 1$  to  $Citycounter(k)$ 
      IF inverting chain between cities  $i$  and  $j$ 
        improves profit of  $TOUR(k)$ 
          Update  $TOUR(k)$  by inverting chain
            between cities  $i$  and  $j$ 
        END IF
      END FOR
    END FOR
  END FOR
  Bas:
  //deletion procedure
  FOR  $i = 1$  to  $Citycounter(k)$ 
    Bas2:
    IF city  $i$  is not the depot
      IF deleting city  $i$  improves profit of
         $TOUR(k)$ 
        Update  $TOUR(k)$  by deleting city  $i$ 
        Decrease  $Citycounter(k)$  by one
        Increase  $otherscounter(k)$  by one
        Decrease  $i$  by one
        Go to Bas2
      END IF
    END IF
  END FOR
  //double deletion procedure
  FOR  $i = 1$  to  $Citycounter(k) - 1$ 

```

```

IF one of cities  $i$  and  $i + 1$  is not the depot
  IF deleting cities  $i$  and  $i + 1$  improves
    profit of  $TOUR(k)$ 
      Update  $TOUR(k)$  by deleting city  $i$  and
       $i + 1$ 
      Decrease  $Citycounter(k)$  by two
      Increase  $otherscounter(k)$  by two
      Go to Bas
    END IF
  END IF
END FOR
//insertion procedure
FOR  $i = 1$  to  $otherscounter(k)$ 
  FOR  $j = 1$  to  $Citycounter(k)$ 
    IF inserting city  $i$  into the edge between  $j$ 
    and  $j + 1$  improves profit of  $TOUR(k)$ 
      IF there was no profitable edge to insert
        city  $i$  so far
        Store the improvement in the profit
        obtained by inserting city  $i$  as
        the best improvement in the profit
        of  $TOUR(k)$ 
        Store the insertion location
      ELSE IF the improvement in the profit
        obtained by inserting city  $i$  is
        greater than the best improvement in
        the profit of  $TOUR(k)$ 
        Store the improvement in the profit
        obtained by inserting city  $i$  as
        the best improvement in the profit
        of  $TOUR(k)$ 

```

```

        Store the insertion location
    END IF
END IF
END FOR
IF there is an insertion location
    Increase Citycounter(k) by one
    Decrease otherscounter(k) by one
    Update TOUR(k) by inserting city i from the
    insertion location
END IF
Go to Bas
END FOR
//double insertion procedure
FOR i = 1 to otherscounter(k) - 1
    FOR j = i + 1 to otherscounter(k)
        FOR l = 1 to Citycounter(k)
            IF there is more than one unvisited city
            for TOUR(k)
                IF any improvement occurs in the
                objective function by inserting
                the city i and j between city
                l and city l + 1
                    Update TOUR(k) by inserting
                    the city i and j between
                    city l and city l + 1 with
                    the order maximum increase
                    in profit is obtained
                END IF
            END IF
        END IF
    END IF
END FOR

```

```

        END FOR
    END FOR
END FOR
FOR each ant k
    IF the profit of  $TOUR(k)$  is greater than the profit of
 $best\_tour(LS)$ 
        Store  $TOUR(k)$  as  $best\_tour(LS)$ 
    END IF
END FOR
IF the profit of  $best\_tour(LS)$  is greater than the profit of
 $best\_globaltour(MR)$ 
    Store  $best\_tour(LS)$  as  $best\_globaltour(MR)$ 
END IF
END FOR
PRINT  $best\_globaltour(MR)$ 

```

APPENDIX B: Coordinates of the Instances

Table B.1. Coordinates of 24th instance

c	x	y	p	c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9	79	57	48	2
2	35	17	6	28	41	37	6	54	57	29	1	80	56	37	12
3	55	45	15	29	64	42	11	55	63	23	5	81	55	54	3
4	55	20	9	30	40	60	2	56	53	12	10	82	15	47	15
5	15	30	11	31	31	52	11	57	32	12	12	83	14	37	13
6	25	30	10	32	35	69	10	58	36	26	11	84	11	31	14
7	20	50	11	33	53	52	11	59	21	24	2	85	16	22	9
8	10	43	3	34	65	55	15	60	17	34	4	86	4	18	13
9	55	60	12	35	63	65	9	61	12	24	2	87	28	18	14
10	30	60	6	36	2	60	6	62	24	58	10	88	26	52	1
11	20	65	7	37	20	20	2	63	27	69	2	89	26	35	3
12	50	35	5	38	5	5	15	64	15	77	8	90	31	67	6
13	30	25	7	39	60	12	13	65	62	77	5	91	15	19	11
14	15	10	4	40	40	25	15	66	49	73	4	92	22	22	10
15	30	5	10	41	42	7	7	67	67	5	10	93	18	24	2
16	10	20	13	42	24	12	9	68	56	39	10	94	26	27	8
17	5	30	2	43	23	3	15	69	37	47	10	95	25	24	13
18	20	40	9	44	11	14	12	70	37	56	5	96	22	27	10
19	15	60	11	45	6	38	6	71	57	68	6	97	25	21	14
20	45	65	3	46	2	48	13	72	47	16	5	98	19	21	6
21	45	20	5	47	8	56	10	73	44	17	3	99	20	26	2
22	45	10	10	48	13	52	15	74	46	13	9	100	18	18	6
23	55	5	13	49	6	68	4	75	49	11	8	101	35	35	13
24	65	35	15	50	47	47	10	76	49	42	15				
25	65	20	6	51	49	58	12	77	53	43	6				
26	45	30	3	52	27	43	4	78	61	52	5				

Table B.2. Coordinates of 23rd instance

c	x	y	p	c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9	79	57	48	2
2	35	17	6	28	41	37	6	54	57	29	1	80	56	37	12
3	55	45	15	29	64	42	11	55	63	23	5	81	55	54	3
4	55	20	9	30	40	60	2	56	53	12	10	82	15	47	15
5	15	30	11	31	31	52	11	57	32	12	12	83	14	37	13
6	25	30	10	32	35	69	10	58	36	26	11	84	11	31	14
7	20	50	11	33	53	52	11	59	21	24	2	85	16	22	9
8	10	43	3	34	65	55	15	60	17	34	4	86	4	18	13
9	55	60	12	35	63	65	9	61	12	24	2	87	28	18	14
10	30	60	6	36	2	60	6	62	24	58	10	88	26	52	1
11	20	65	7	37	20	20	2	63	27	69	2	89	26	35	3
12	50	35	5	38	5	5	15	64	15	77	8	90	31	67	6
13	30	25	7	39	60	12	13	65	62	77	5	91	15	19	11
14	15	10	4	40	40	25	15	66	49	73	4	92	22	22	10
15	30	5	10	41	42	7	7	67	67	5	10	93	18	24	2
16	10	20	13	42	24	12	9	68	56	39	10	94	26	27	8
17	5	30	2	43	23	3	15	69	37	47	10	95	25	24	13
18	20	40	9	44	11	14	12	70	37	56	5	96	22	27	10
19	15	60	11	45	6	38	6	71	57	68	6				
20	45	65	3	46	2	48	13	72	47	16	5				
21	45	20	5	47	8	56	10	73	44	17	3				
22	45	10	10	48	13	52	15	74	46	13	9				
23	55	5	13	49	6	68	4	75	49	11	8				
24	65	35	15	50	47	47	10	76	49	42	15				
25	65	20	6	51	49	58	12	77	53	43	6				
26	45	30	3	52	27	43	4	78	61	52	5				

Table B.3. Coordinates of 22nd instance

c	x	y	p	c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9	79	57	48	2
2	35	17	6	28	41	37	6	54	57	29	1	80	56	37	12
3	55	45	15	29	64	42	11	55	63	23	5	81	55	54	3
4	55	20	9	30	40	60	2	56	53	12	10	82	15	47	15
5	15	30	11	31	31	52	11	57	32	12	12	83	14	37	13
6	25	30	10	32	35	69	10	58	36	26	11	84	11	31	14
7	20	50	11	33	53	52	11	59	21	24	2	85	16	22	9
8	10	43	3	34	65	55	15	60	17	34	4	86	4	18	13
9	55	60	12	35	63	65	9	61	12	24	2	87	28	18	14
10	30	60	6	36	2	60	6	62	24	58	10	88	26	52	1
11	20	65	7	37	20	20	2	63	27	69	2	89	26	35	3
12	50	35	5	38	5	5	15	64	15	77	8	90	31	67	6
13	30	25	7	39	60	12	13	65	62	77	5	91	15	19	11
14	15	10	4	40	40	25	15	66	49	73	4				
15	30	5	10	41	42	7	7	67	67	5	10				
16	10	20	13	42	24	12	9	68	56	39	10				
17	5	30	2	43	23	3	15	69	37	47	10				
18	20	40	9	44	11	14	12	70	37	56	5				
19	15	60	11	45	6	38	6	71	57	68	6				
20	45	65	3	46	2	48	13	72	47	16	5				
21	45	20	5	47	8	56	10	73	44	17	3				
22	45	10	10	48	13	52	15	74	46	13	9				
23	55	5	13	49	6	68	4	75	49	11	8				
24	65	35	15	50	47	47	10	76	49	42	15				
25	65	20	6	51	49	58	12	77	53	43	6				
26	45	30	3	52	27	43	4	78	61	52	5				

Table B.4. Coordinates of 21st instance

c	x	y	p	c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9	79	57	48	2
2	35	17	6	28	41	37	6	54	57	29	1	80	56	37	12
3	55	45	15	29	64	42	11	55	63	23	5	81	55	54	3
4	55	20	9	30	40	60	2	56	53	12	10	82	15	47	15
5	15	30	11	31	31	52	11	57	32	12	12	83	14	37	13
6	25	30	10	32	35	69	10	58	36	26	11	84	11	31	14
7	20	50	11	33	53	52	11	59	21	24	2	85	16	22	9
8	10	43	3	34	65	55	15	60	17	34	4	86	4	18	13
9	55	60	12	35	63	65	9	61	12	24	2				
10	30	60	6	36	2	60	6	62	24	58	10				
11	20	65	7	37	20	20	2	63	27	69	2				
12	50	35	5	38	5	5	15	64	15	77	8				
13	30	25	7	39	60	12	13	65	62	77	5				
14	15	10	4	40	40	25	15	66	49	73	4				
15	30	5	10	41	42	7	7	67	67	5	10				
16	10	20	13	42	24	12	9	68	56	39	10				
17	5	30	2	43	23	3	15	69	37	47	10				
18	20	40	9	44	11	14	12	70	37	56	5				
19	15	60	11	45	6	38	6	71	57	68	6				
20	45	65	3	46	2	48	13	72	47	16	5				
21	45	20	5	47	8	56	10	73	44	17	3				
22	45	10	10	48	13	52	15	74	46	13	9				
23	55	5	13	49	6	68	4	75	49	11	8				
24	65	35	15	50	47	47	10	76	49	42	15				
25	65	20	6	51	49	58	12	77	53	43	6				
26	45	30	3	52	27	43	4	78	61	52	5				

Table B.5. Coordinates of 20th instance

c	x	y	p	c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9	79	57	48	2
2	35	17	6	28	41	37	6	54	57	29	1	80	56	37	12
3	55	45	15	29	64	42	11	55	63	23	5	81	55	54	3
4	55	20	9	30	40	60	2	56	53	12	10				
5	15	30	11	31	31	52	11	57	32	12	12				
6	25	30	10	32	35	69	10	58	36	26	11				
7	20	50	11	33	53	52	11	59	21	24	2				
8	10	43	3	34	65	55	15	60	17	34	4				
9	55	60	12	35	63	65	9	61	12	24	2				
10	30	60	6	36	2	60	6	62	24	58	10				
11	20	65	7	37	20	20	2	63	27	69	2				
12	50	35	5	38	5	5	15	64	15	77	8				
13	30	25	7	39	60	12	13	65	62	77	5				
14	15	10	4	40	40	25	15	66	49	73	4				
15	30	5	10	41	42	7	7	67	67	5	10				
16	10	20	13	42	24	12	9	68	56	39	10				
17	5	30	2	43	23	3	15	69	37	47	10				
18	20	40	9	44	11	14	12	70	37	56	5				
19	15	60	11	45	6	38	6	71	57	68	6				
20	45	65	3	46	2	48	13	72	47	16	5				
21	45	20	5	47	8	56	10	73	44	17	3				
22	45	10	10	48	13	52	15	74	46	13	9				
23	55	5	13	49	6	68	4	75	49	11	8				
24	65	35	15	50	47	47	10	76	49	42	15				
25	65	20	6	51	49	58	12	77	53	43	6				
26	45	30	3	52	27	43	4	78	61	52	5				

Table B.6. Coordinates of 19th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9
2	35	17	6	28	41	37	6	54	57	29	1
3	55	45	15	29	64	42	11	55	63	23	5
4	55	20	9	30	40	60	2	56	53	12	10
5	15	30	11	31	31	52	11	57	32	12	12
6	25	30	10	32	35	69	10	58	36	26	11
7	20	50	11	33	53	52	11	59	21	24	2
8	10	43	3	34	65	55	15	60	17	34	4
9	55	60	12	35	63	65	9	61	12	24	2
10	30	60	6	36	2	60	6	62	24	58	10
11	20	65	7	37	20	20	2	63	27	69	2
12	50	35	5	38	5	5	15	64	15	77	8
13	30	25	7	39	60	12	13	65	62	77	5
14	15	10	4	40	40	25	15	66	49	73	4
15	30	5	10	41	42	7	7	67	67	5	10
16	10	20	13	42	24	12	9	68	56	39	10
17	5	30	2	43	23	3	15	69	37	47	10
18	20	40	9	44	11	14	12	70	37	56	5
19	15	60	11	45	6	38	6	71	57	68	6
20	45	65	3	46	2	48	13	72	47	16	5
21	45	20	5	47	8	56	10	73	44	17	3
22	45	10	10	48	13	52	15	74	46	13	9
23	55	5	13	49	6	68	4	75	49	11	8
24	65	35	15	50	47	47	10	76	49	42	15
25	65	20	6	51	49	58	12				
26	45	30	3	52	27	43	4				

Table B.7. Coordinates of 18th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4	53	55	57	11
2	36	26	6	28	35	16	5	54	67	41	3
3	21	45	15	29	52	26	2	55	10	70	13
4	45	35	3	30	43	26	14	56	6	25	5
5	55	20	1	31	31	76	12	57	65	27	7
6	33	34	10	32	22	53	12	58	40	60	5
7	50	50	4	33	26	29	15	59	70	64	14
8	55	45	7	34	50	40	3	60	64	4	9
9	26	59	3	35	55	50	14	61	36	6	6
10	40	66	13	36	54	10	14	62	30	20	4
11	55	65	15	37	60	15	7	63	20	30	2
12	35	51	7	38	47	66	5	64	15	5	8
13	62	35	13	39	30	60	6	65	50	70	1
14	62	57	12	40	30	50	7	66	57	72	1
15	62	24	9	41	12	17	11	67	45	42	15
16	21	36	12	42	15	14	6	68	38	33	15
17	33	44	10	43	16	19	3	69	50	4	11
18	9	56	8	44	21	48	11	70	66	8	15
19	62	48	9	45	50	30	1	71	59	5	4
20	66	14	4	46	51	42	15	72	35	60	6
21	44	13	12	47	50	15	7	73	27	24	12
22	26	13	5	48	43	21	4	74	40	20	6
23	11	28	10	49	12	38	12	75	40	37	14
24	7	43	5	50	15	56	5	76	40	40	12
25	17	64	14	51	29	39	15				
26	41	46	6	52	54	38	4				

Table B.8. Coordinates of 17th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9
2	35	17	6	28	41	37	6	54	57	29	1
3	55	45	15	29	64	42	11	55	63	23	5
4	55	20	9	30	40	60	2	56	53	12	10
5	15	30	11	31	31	52	11	57	32	12	12
6	25	30	10	32	35	69	10	58	36	26	11
7	20	50	11	33	53	52	11	59	21	24	2
8	10	43	3	34	65	55	15	60	17	34	4
9	55	60	12	35	63	65	9	61	12	24	2
10	30	60	6	36	2	60	6	62	24	58	10
11	20	65	7	37	20	20	2	63	27	69	2
12	50	35	5	38	5	5	15	64	15	77	8
13	30	25	7	39	60	12	13	65	62	77	5
14	15	10	4	40	40	25	15	66	49	73	4
15	30	5	10	41	42	7	7	67	67	5	10
16	10	20	13	42	24	12	9	68	56	39	10
17	5	30	2	43	23	3	15	69	37	47	10
18	20	40	9	44	11	14	12	70	37	56	5
19	15	60	11	45	6	38	6	71	57	68	6
20	45	65	3	46	2	48	13				
21	45	20	5	47	8	56	10				
22	45	10	10	48	13	52	15				
23	55	5	13	49	6	68	4				
24	65	35	15	50	47	47	10				
25	65	20	6	51	49	58	12				
26	45	30	3	52	27	43	4				

Table B.9. Coordinates of 16th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4	53	55	57	11
2	36	26	6	28	35	16	5	54	67	41	3
3	21	45	15	29	52	26	2	55	10	70	13
4	45	35	3	30	43	26	14	56	6	25	5
5	55	20	1	31	31	76	12	57	65	27	7
6	33	34	10	32	22	53	12	58	40	60	5
7	50	50	4	33	26	29	15	59	70	64	14
8	55	45	7	34	50	40	3	60	64	4	9
9	26	59	3	35	55	50	14	61	36	6	6
10	40	66	13	36	54	10	14	62	30	20	4
11	55	65	15	37	60	15	7	63	20	30	2
12	35	51	7	38	47	66	5	64	15	5	8
13	62	35	13	39	30	60	6	65	50	70	1
14	62	57	12	40	30	50	7	66	57	72	1
15	62	24	9	41	12	17	11	67	45	42	15
16	21	36	12	42	15	14	6	68	38	33	15
17	33	44	10	43	16	19	3	69	50	4	11
18	9	56	8	44	21	48	11	70	66	8	15
19	62	48	9	45	50	30	1	71	59	5	4
20	66	14	4	46	51	42	15				
21	44	13	12	47	50	15	7				
22	26	13	5	48	48	21	4				
23	11	28	10	49	12	38	12				
24	7	43	5	50	15	56	5				
25	17	64	14	51	29	39	15				
26	41	46	6	52	54	38	4				

Table B.10. Coordinates of 15th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9
2	35	17	6	28	41	37	6	54	57	29	1
3	55	45	15	29	64	42	11	55	63	23	5
4	55	20	9	30	40	60	2	56	53	12	10
5	15	30	11	31	31	52	11	57	32	12	12
6	25	30	10	32	35	69	10	58	36	26	11
7	20	50	11	33	53	52	11	59	21	24	2
8	10	43	3	34	65	55	15	60	17	34	4
9	55	60	12	35	63	65	9	61	12	24	2
10	30	60	6	36	2	60	6	62	24	58	10
11	20	65	7	37	20	20	2	63	27	69	2
12	50	35	5	38	5	5	15	64	15	77	8
13	30	25	7	39	60	12	13	65	62	77	5
14	15	10	4	40	40	25	15	66	49	73	4
15	30	5	10	41	42	7	7				
16	10	20	13	42	24	12	9				
17	5	30	2	43	23	3	15				
18	20	40	9	44	11	14	12				
19	15	60	11	45	6	38	6				
20	45	65	3	46	2	48	13				
21	45	20	5	47	8	56	10				
22	45	10	10	48	13	52	15				
23	55	5	13	49	6	68	4				
24	65	35	15	50	47	47	10				
25	65	20	6	51	49	58	12				
26	45	30	3	52	27	43	4				

Table B.11. Coordinates of 14th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4	53	55	57	11
2	36	26	6	28	35	16	5	54	67	41	3
3	21	45	15	29	52	26	2	55	10	70	13
4	45	35	3	30	43	26	14	56	6	25	5
5	55	20	1	31	31	76	12	57	65	27	7
6	33	34	10	32	22	53	12	58	40	60	5
7	50	50	4	33	26	29	15	59	70	64	14
8	55	45	7	34	50	40	3	60	64	4	9
9	26	59	3	35	55	50	14	61	36	6	6
10	40	66	13	36	54	10	14	62	30	20	4
11	55	65	15	37	60	15	7	63	20	30	2
12	35	51	7	38	47	66	5	64	15	5	8
13	62	35	13	39	30	60	6	65	50	70	1
14	62	57	12	40	30	50	7	66	57	72	1
15	62	24	9	41	12	17	11				
16	21	36	12	42	15	14	6				
17	33	44	10	43	16	19	3				
18	9	56	8	44	21	48	11				
19	62	48	9	45	50	30	1				
20	66	14	4	46	51	42	15				
21	44	13	12	47	50	15	7				
22	26	13	5	48	48	21	4				
23	11	28	10	49	12	38	12				
24	7	43	5	50	15	56	5				
25	17	64	14	51	29	39	15				
26	41	46	6	52	54	38	4				

Table B.12. Coordinates of 13rd instance

c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9
2	35	17	6	28	41	37	6	54	57	29	1
3	55	45	15	29	64	42	11	55	63	23	5
4	55	20	9	30	40	60	2	56	53	12	10
5	15	30	11	31	31	52	11	57	32	12	12
6	25	30	10	32	35	69	10	58	36	26	11
7	20	50	11	33	53	52	11	59	21	24	2
8	10	43	3	34	65	55	15	60	17	34	4
9	55	60	12	35	63	65	9	61	12	24	2
10	30	60	6	36	2	60	6				
11	20	65	7	37	20	20	2				
12	50	35	5	38	5	5	15				
13	30	25	7	39	60	12	13				
14	15	10	4	40	40	25	15				
15	30	5	10	41	42	7	7				
16	10	20	13	42	24	12	9				
17	5	30	2	43	23	3	15				
18	20	40	9	44	11	14	12				
19	15	60	11	45	6	38	6				
20	45	65	3	46	2	48	13				
21	45	20	5	47	8	56	10				
22	45	10	10	48	13	52	15				
23	55	5	13	49	6	68	4				
24	65	35	15	50	47	47	10				
25	65	20	6	51	49	58	12				
26	45	30	3	52	27	43	4				

Table B.13. Coordinates of 12nd instance

c	x	y	p	c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4	53	55	57	11
2	36	26	6	28	35	16	5	54	67	41	3
3	21	45	15	29	52	26	2	55	10	70	13
4	45	35	3	30	43	26	14	56	6	25	5
5	55	20	1	31	31	76	12	57	65	27	7
6	33	34	10	32	22	53	12	58	40	60	5
7	50	50	4	33	26	29	15	59	70	64	14
8	55	45	7	34	50	40	3	60	64	4	9
9	26	59	3	35	55	50	14	61	36	6	6
10	40	66	13	36	54	10	14				
11	55	65	15	37	60	15	7				
12	35	51	7	38	47	66	5				
13	62	35	13	39	30	60	6				
14	62	57	12	40	30	50	7				
15	62	24	9	41	12	17	11				
16	21	36	12	42	15	14	6				
17	33	44	10	43	16	19	3				
18	9	56	8	44	21	48	11				
19	62	48	9	45	50	30	1				
20	66	14	4	46	51	42	15				
21	44	13	12	47	50	15	7				
22	26	13	5	48	48	21	4				
23	11	28	10	49	12	38	12				
24	7	43	5	50	15	56	5				
25	17	64	14	51	29	39	15				
26	41	46	6	52	54	38	4				

Table B.14. Coordinates of 11st instance

c	x	y	p	c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2	53	37	31	9
2	35	17	6	28	41	37	6	54	57	29	1
3	55	45	15	29	64	42	11	55	63	23	5
4	55	20	9	30	40	60	2	56	53	12	10
5	15	30	11	31	31	52	11				
6	25	30	10	32	35	69	10				
7	20	50	11	33	53	52	11				
8	10	43	3	34	65	55	15				
9	55	60	12	35	63	65	9				
10	30	60	6	36	2	60	6				
11	20	65	7	37	20	20	2				
12	50	35	5	38	5	5	15				
13	30	25	7	39	60	12	13				
14	15	10	4	40	40	25	15				
15	30	5	10	41	42	7	7				
16	10	20	13	42	24	12	9				
17	5	30	2	43	23	3	15				
18	20	40	9	44	11	14	12				
19	15	60	11	45	6	38	6				
20	45	65	3	46	2	48	13				
21	45	20	5	47	8	56	10				
22	45	10	10	48	13	52	15				
23	55	5	13	49	6	68	4				
24	65	35	15	50	47	47	10				
25	65	20	6	51	49	58	12				
26	45	30	3	52	27	43	4				

Table B.15. Coordinates of 10th instance

c	x	y	p	c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4	53	55	57	11
2	36	26	6	28	35	16	5	54	67	41	3
3	21	45	15	29	52	26	2	55	10	70	13
4	45	35	3	30	43	26	14	56	6	25	5
5	55	20	1	31	31	76	12				
6	33	34	10	32	22	53	12				
7	50	50	4	33	26	29	15				
8	55	45	7	34	50	40	3				
9	26	59	3	35	55	50	14				
10	40	66	13	36	54	10	14				
11	55	65	15	37	60	15	7				
12	35	51	7	38	47	66	5				
13	62	35	13	39	30	60	6				
14	62	57	12	40	30	50	7				
15	62	24	9	41	12	17	11				
16	21	36	12	42	15	14	6				
17	33	44	10	43	16	19	3				
18	9	56	8	44	21	48	11				
19	62	48	9	45	50	30	1				
20	66	14	4	46	51	42	15				
21	44	13	12	47	50	15	7				
22	26	13	5	48	48	21	4				
23	11	28	10	49	12	38	12				
24	7	43	5	50	15	56	5				
25	17	64	14	51	29	39	15				
26	41	46	6	52	54	38	4				

Table B.16. Coordinates of 9th instance

c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2
2	35	17	6	28	41	37	6
3	55	45	15	29	64	42	11
4	55	20	9	30	40	60	2
5	15	30	11	31	31	52	11
6	25	30	10	32	35	69	10
7	20	50	11	33	53	52	11
8	10	43	3	34	65	55	15
9	55	60	12	35	63	65	9
10	30	60	6	36	2	60	6
11	20	65	7	37	20	20	2
12	50	35	5	38	5	5	15
13	30	25	7	39	60	12	13
14	15	10	4	40	40	25	15
15	30	5	10	41	42	7	7
16	10	20	13	42	24	12	9
17	5	30	2	43	23	3	15
18	20	40	9	44	11	14	12
19	15	60	11	45	6	38	6
20	45	65	3	46	2	48	13
21	45	20	5	47	8	56	10
22	45	10	10	48	13	52	15
23	55	5	13	49	6	68	4
24	65	35	15	50	47	47	10
25	65	20	6	51	49	58	12
26	45	30	3				

Table B.17. Coordinates of 8th instance

c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4
2	36	26	6	28	35	16	5
3	21	45	15	29	52	26	2
4	45	35	3	30	43	26	14
5	55	20	1	31	31	76	12
6	33	34	10	32	22	53	12
7	50	50	4	33	26	29	15
8	55	45	7	34	50	40	3
9	26	59	3	35	55	50	14
10	40	66	13	36	54	10	14
11	55	65	15	37	60	15	7
12	35	51	7	38	47	66	5
13	62	35	13	39	30	60	6
14	62	57	12	40	30	50	7
15	62	24	9	41	12	17	11
16	21	36	12	42	15	14	6
17	33	44	10	43	16	19	3
18	9	56	8	44	21	48	11
19	62	48	9	45	50	30	1
20	66	14	4	46	51	42	15
21	44	13	12	47	50	15	7
22	26	13	5	48	48	21	4
23	11	28	10	49	12	38	12
24	7	43	5	50	15	56	5
25	17	64	14	51	29	39	15
26	41	46	6				

Table B.18. Coordinates of 7th instance

c	x	y	p	c	x	y	p
1	37	52	0	27	30	48	4
2	49	49	5	28	43	67	8
3	52	64	9	29	58	48	1
4	20	26	15	30	58	27	10
5	40	30	6	31	37	69	6
6	21	47	14	32	38	46	15
7	17	63	10	33	46	10	9
8	31	62	6	34	61	33	15
9	52	33	9	35	62	63	7
10	51	21	2	36	63	69	7
11	42	41	3	37	32	22	2
12	31	32	3	38	45	35	9
13	5	25	8	39	59	15	4
14	12	42	14	40	5	6	13
15	36	16	15	41	10	17	7
16	52	41	12	42	21	10	5
17	26	23	8	43	5	64	4
18	17	33	9	44	30	15	15
19	13	13	12	45	39	10	15
20	57	58	2	46	32	39	11
21	62	42	4	47	25	32	5
22	42	57	8	48	25	55	13
23	16	57	7	49	48	28	5
24	8	52	1	50	56	37	8
25	7	38	2	51	30	40	5
26	27	68	4				

Table B.19. Coordinates of 6th instance

c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2
2	35	17	6	28	41	37	6
3	55	45	15	29	64	42	11
4	55	20	9	30	40	60	2
5	15	30	11	31	31	52	11
6	25	30	10	32	35	69	10
7	20	50	11	33	53	52	11
8	10	43	3	34	65	55	15
9	55	60	12	35	63	65	9
10	30	60	6	36	2	60	6
11	20	65	7	37	20	20	2
12	50	35	5	38	5	5	15
13	30	25	7	39	60	12	13
14	15	10	4	40	40	25	15
15	30	5	10	41	42	7	7
16	10	20	13	42	24	12	9
17	5	30	2	43	23	3	15
18	20	40	9	44	11	14	12
19	15	60	11	45	6	38	6
20	45	65	3	46	2	48	13
21	45	20	5				
22	45	10	10				
23	55	5	13				
24	65	35	15				
25	65	20	6				
26	45	30	3				

Table B.20. Coordinates of 5th instance

c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4
2	36	26	6	28	35	16	5
3	21	45	15	29	52	26	2
4	45	35	3	30	43	26	14
5	55	20	1	31	31	76	12
6	33	34	10	32	22	53	12
7	50	50	4	33	26	29	15
8	55	45	7	34	50	40	3
9	26	59	3	35	55	50	14
10	40	66	13	36	54	10	14
11	55	65	15	37	60	15	7
12	35	51	7	38	47	66	5
13	62	35	13	39	30	60	6
14	62	57	12	40	30	50	7
15	62	24	9	41	12	17	11
16	21	36	12	42	15	14	6
17	33	44	10	43	16	19	3
18	9	56	8	44	21	48	11
19	62	48	9	45	50	30	1
20	66	14	4	46	51	42	15
21	44	13	12				
22	26	13	5				
23	11	28	10				
24	7	43	5				
25	17	64	14				
26	41	46	6				

Table B.21. Coordinates of 4th instance

c	x	y	p	c	x	y	p
1	37	52	0	27	30	48	4
2	49	49	5	28	43	67	8
3	52	64	9	29	58	48	1
4	20	26	15	30	58	27	10
5	40	30	6	31	37	69	6
6	21	47	14	32	38	46	15
7	17	63	10	33	46	10	9
8	31	62	6	34	61	33	15
9	52	33	9	35	62	63	7
10	51	21	2	36	63	69	7
11	42	41	3	37	32	22	2
12	31	32	3	38	45	35	9
13	5	25	8	39	59	15	4
14	12	42	14	40	5	6	13
15	36	16	15	41	10	17	7
16	52	41	12	42	21	10	5
17	26	23	8	43	5	64	4
18	17	33	9	44	30	15	15
19	13	13	12	45	39	10	15
20	57	58	2	46	32	39	11
21	62	42	4				
22	42	57	8				
23	16	57	7				
24	8	52	1				
25	7	38	2				
26	27	68	4				

Table B.22. Coordinates of 3rd instance

c	x	y	p	c	x	y	p
1	41	49	0	27	35	40	2
2	35	17	6	28	41	37	6
3	55	45	15	29	64	42	11
4	55	20	9	30	40	60	2
5	15	30	11	31	31	52	11
6	25	30	10	32	35	69	10
7	20	50	11	33	53	52	11
8	10	43	3	34	65	55	15
9	55	60	12	35	63	65	9
10	30	60	6	36	2	60	6
11	20	65	7	37	20	20	2
12	50	35	5	38	5	5	15
13	30	25	7	39	60	12	13
14	15	10	4	40	40	25	15
15	30	5	10	41	42	7	7
16	10	20	13				
17	5	30	2				
18	20	40	9				
19	15	60	11				
20	45	65	3				
21	45	20	5				
22	45	10	10				
23	55	5	13				
24	65	35	15				
25	65	20	6				
26	45	30	3				

Table B.23. Coordinates of 2nd instance

c	x	y	p	c	x	y	p
1	22	22	0	27	55	34	4
2	36	26	6	28	35	16	5
3	21	45	15	29	52	26	2
4	45	35	3	30	43	26	14
5	55	20	1	31	31	76	12
6	33	34	10	32	22	53	12
7	50	50	4	33	26	29	15
8	55	45	7	34	50	40	3
9	26	59	3	35	55	50	14
10	40	66	13	36	54	10	14
11	55	65	15	37	60	15	7
12	35	51	7	38	47	66	5
13	62	35	13	39	30	60	6
14	62	57	12	40	30	50	7
15	62	24	9	41	12	17	11
16	21	36	12				
17	33	44	10				
18	9	56	8				
19	62	48	9				
20	66	14	4				
21	44	13	12				
22	26	13	5				
23	11	28	10				
24	7	43	5				
25	17	64	14				
26	41	46	6				

Table B.24. Coordinates of 1st instance

c	x	y	p	c	x	y	p
1	37	52	0	27	30	48	4
2	49	49	5	28	43	67	8
3	52	64	9	29	58	48	1
4	20	26	15	30	58	27	10
5	40	30	6	31	37	69	6
6	21	47	14	32	38	46	15
7	17	63	10	33	46	10	9
8	31	62	6	34	61	33	15
9	52	33	9	35	62	63	7
10	51	21	2	36	63	69	7
11	42	41	3	37	32	22	2
12	31	32	3	38	45	35	9
13	5	25	8	39	59	15	4
14	12	42	14	40	5	6	13
15	36	16	15	41	10	17	7
16	52	41	12				
17	26	23	8				
18	17	33	9				
19	13	13	12				
20	57	58	2				
21	62	42	4				
22	42	57	8				
23	16	57	7				
24	8	52	1				
25	7	38	2				
26	27	68	4				

Table B.25. Optimal Tours of Instances 1-17

1	5,15,37,17,4,18,14,6,27,1,32,11,16,34,30,9,38,5
2	2,6,33,1,41,23,16,3,32,9,39,10,38,11,14,19,35,8,13,15,37,36,21,30,2
3	1,3,33,9,35,34,29,24,25,39,23,41,22,21,40,13,6,5,18,7,31,1
4	14,6,27,1,32,11,38,16,9,34,30,10,33,45,15,44,17,4,18,14
5	2,6,33,1,43,42,41,23,16,3,44,32,9,39,10,38,11,14,19,35,8,46,34,27,13, 15,37,36,21,30,2
6	1,3,33,9,35,34,29,24,25,39,23,22,41,15,43,42,14,44,16,5,18,7,31,1
7	30,34,50,16,9,38,5,46,51,32,1,48,6,14,18,4,17,44,15,45,33,10,30
8	6,2,30,48,21,47,36,37,15,13,27,34,46,8,35,19,14,11,38,10,39,9,25,50, 32,44,3,40,12,17,51,16,49,23,41,42,43,1,33,6
9	33,3,50,1,31,10,11,19,47,48,7,18,60,5,61,16,44,14,42,43,15,57,2,13,53, 58,40,21,22,56,23,39,25,55,24,29,34,35,9,51,33
10	2,6,33,1,43,42,41,56,23,49,16,51,17,12,40,3,44,32,50,25,9,39,10,38,11, 53,14,19,35,8,46,34,52,27,13,15,37,36,47,21,48,30,2
11	1,31,10,11,19,47,48,7,18,6,5,16,44,14,42,43,15,41,22,23,39,25,24,29,34, 35,9,51,33,3,50,1
12	58,13,2,57,15,43,42,14,44,16,61,5,60,18,7,48,47,19,11,62,10,31,1,50,3, 33,51,9,35,34,29,24,55,25,39,23,56,22,21,40,53,58
13	24,29,34,35,9,51,33,3,50,1,28,53,40,13,6,18,7,48,19,47,46,45,5,16,44,14, 42,43,15,41,22,56,23,39,25,55,24
14	30,48,21,47,36,37,15,57,13,27,52,34,46,8,35,19,14,53,11,38,10,58,39,9, 25,50,32,44,3,40,12,17,51,16,49,23,56,41,42,43,1,33,6,2,30
15	58,13,2,57,15,43,42,14,44,16,61,5,60,18,7,48,47,19,11,62,10,31,1,50,3,33, 51,9,35,34,29,24,55,25,39,23,56,22,21,40,53,58
16	32,44,3,40,12,17,51,16,49,23,56,41,42,43,1,33,6,68,2,30,48,47,21,69,36,71, 60,70,20,37,15,57,13,27,52,34,67,46,8,35,19,14,53,11,38,10,58,39,9,25,50,32
17	56,23,39,4,21,40,58,53,28,12,68,24,29,34,35,71,9,51,33,3,50,1,69,31,10,62, 11,19,47,48,7,18,60,5,61,16,44,14,42,43,15,57,41,22,56

Table B.26. Optimal Tours of Instances 18-25

18	69,36,47,21,74,30,2,68,75,76,67,26,12,40,17,51,6,33,73,1,43,42,41,56,23, 49,16,3,44,32,50,25,9,39,72,58,10,38,11,53,14,19,35,8,46,34,52,27,13,57, 15,37,20,70,60,71,69
19	9,51,33,3,68,76,50,1,69,31,10,62,11,19,47,48,7,18,60,5,61,16,44,14,42,43, 15,57,2,13,58,53,40,21,73,72,74,22,75,56,23,39,25,55,24,29,34,35,71,9
20	13,2,57,15,43,42,14,44,16,61,5,60,18,7,48,47,19,11,62,10,31,69,1,50,33, 81,51,9,71,35,34,78,79,3,77,76,12,80,68,29,24,55,25,39,23,56,75,22,74, 72,73,21,40,53,58,13
21	2,13,58,53,40,21,21,73,72,74,22,75,56,23,39,39,25,55,24,29,68,80,12,76, 77,3,79,78,34,35,71,9,51,81,33,50,1,69,31,10,62,11,19,47,48,82,7,18,83, 60,5,84,61,85,16,86,44,14,42,43,15,57,2
22	31,10,62,11,19,47,48,82,7,18,83,60,5,84,61,85,91,16,86,44,14,42,43,15, 57,2,87,13,58,53,40,21,73,72,74,22,75,23,39,25,55,24,29,68,80,12,76,77, 3,79,78,34,35,71,9,51,81,33,50,1,69,69,31
23	31,10,62,11,19,47,48,82,7,18,83,60,5,84,61,16,86,44,91,85,93,59,92,95, 96,6,94,13,87,42,43,15,57,2,58,53,40,21,73,72,74,22,75,56,23,39,25,55, 24,29,68,80,12,76,77,3,79,78,34,35,71,9,51,81,33,50,1,69
24	60,5,84,61,16,86,44,91,85,100,98,37,92,59,99,96,6,94,13,95,97,87,42, 43,15,57,41,22,74,75,56,23,39,4,72,73,21,40,58,53,101,28,76,77,3,68, 80,24,29,78,34,35,71,9,51,81,33,50,1,69,31,10,62,11,19,47,48,82,7,18, 83,60

REFERENCES

1. Alaykran, K. and O. Engin, "Karıncı Kolonileri Metasezgiseli Ve Gezgin Satıcı Problemleri Üzerinde Bir Uygulaması", *J. Fac. Eng. Arch. Gazi Univ.*, Vol. 20, No. 1, pp. 69-76, 2005.
2. Archetti, C., A. Hertz and M. G. Speranza, "Metaheuristics for the team orienteering problem", *Journal of Heuristics*, Vol. 13, No. 1, pp. 49-76, 2007.
3. Aron, S., J. L. Deneubourg, S. Goss, and J.M. Pasteels, "Functional selforganization illustrated by inter-nest traffic in the argentine ant *iridomyrmex humilis*", *Biological Motion*, pp 533-547, 1990.
4. Awerbuch, B., Y. Azar, A. Blum, S. Vempala, "New approximation guarantees for minimum-weight k-trees and prize collecting salesmen", *SIAM Journal on Computing*, Vol. 28, No. 1, pp. 254-262, 1998.
5. Beckers, R., J. L. Deneubourg and S. Goss, "Trails and U-turns in the Selection of the Shortest Path by the Ant *Lasius Niger*", *Journal of Theoretical Biology*, 159, pp. 397-415, 1992.
6. Bienstock, D., M. X. Goemans, D. Simchi-Levi and D. Williamson, "A note on the prize collecting travelling salesman problem", *Mathematical Programming*, Vol. 59, pp. 413-420, 1993.
7. Bonabeau, E., M. Dorigo, G. Theraulaz, "Swarm intelligence: from natural to artificial systems", *Oxford University Press US*, pp. 26, 1999.
8. Bullnheimer, R., F. Hartl and C. Strauss, "A New Rank Based Version of the Ant System: A Computational Study", *Central European Journal for Ops. Research and Economics*, vol. 7, pp. 25-38, 1999.

9. Chao, I.-M., B. L. Golden, E. A. Wasil, "A fast and effective heuristic for the orienteering problem", *European Journal of Operations Research*, Vol. 88, No. 3, pp. 475-489, 1996a.
10. Chao, I.-M., B. Golden and E. A. Wasil, "The team orienteering problem", *European Journal of Operational Research*, Vol. 88, No. 3, pp. 464-474, 1996b.
11. Christofides, N., "Worst-case analysis of a new heuristic for the travelling salesman problem", *Technical Report 388*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
12. Colomi A., M. Dorigo and V. Maniezzo, "Distributed Optimization by Ant Colonies, Proceedings of the First European Conference on Artificial Life", *Paris, France, F. Varela and P. Bourguine, Elsevier Publishing*, pp. 134-142, 1992.
13. Dell'Amico, M., F. Maffioli and A. Sciomachen, "A Lagrangian heuristic for the Prize Collecting Travelling Salesman Problem", *Annals of Operations Research*, Vol. 81, No. 0, pp. 289-306, 1998.
14. Dell'Amico, M., F. Maoli and P. Varbrand, "On Prize-collecting Tours ant The Asymmetric TSP", *International Transactions in Operational Research*, Vol. 2, No. 3, pp. 297-308, 1995.
15. Dorigo M. and Stützle T., "Ant Colony Optimization", *MIT press*, pp. 216, 2004.
16. Dorigo M. and Stützle T., "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances", *International Series in Operations Research and Management Science*, Vol. 57, pp. 250-285, 2002.
17. Dorigo M., "Optimization, Learning and Natural Algorithms", *Ph.D.Thesis, Politecnico di Milano, Italy*, 1992, in Italian.
18. Dorigo, M. and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary*

- Computation*, Vol. 1, No.1, pp. 53–66, 1997a.
19. Dorigo, M. and C. Blum, "Ant colony optimization theory: A survey", *Theoretical Computer Science*, Vol. 344, pp. 243–278, 2005.
 20. Dorigo, M. and G. Di Caro, "The ant colony optimization meta-heuristic", *New Ideas in Optimization*, McGraw Hill, London, UK, pp. 11–32, 1999.
 21. Dorigo, M., V. Maniezzo, and A. Colomi, "Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man and Cybernetics-Part B, Cybernetics*, Vol. 26, No. 1, pp. 29–41, 1996a.
 22. Dorigo, M., and L. M. Gambardella, "A study of some properties of Ant-Q, In Proceedings Fourth International Conference On Parallel Problem Solving From Nature", *Berlin: Springer-Verlag*, Vol. 1141, pp. 656–665, 1996b.
 23. Dorigo, M., L. Gambardella, "Ant colonies for the travelling salesman problem", *Biosystems*, Vol. 43, No. 2, pp. 73–81, 1997b.
 24. Dorigo, M., M. Birattari and T. Stutzle, "Ant Colony Optimization, Artificial Ants as a Computational Intelligence Technique", *IEEE Computational Intelligence Magazine*, Vol. 1, pp. 28–33, 2006.
 25. Dorigo, M., V. Maniezzo and A. Colomi, "Positive feedback as a search strategy", *Working paper, Department of Electronics, Politecnico di Milano*, pp.91–16, 1991.
 26. Dueck, G., "New optimization heuristics: The great deluge algorithm and the record-to-record travel", *Journal of Computational Physics*, Vol. 104, No. 1, pp. 86–92, 1993.
 27. Fischetti, M., J. J. Salazar González, P. Toth, "Solving the orienteering problem through branch-and-cut", *INFORMS Journal on Computing*, Vol. 10, No. 2, pp. 133–148, 1998.

28. Gambardella, L. M. and M. Dorigo, "Ant-Q: A Reinforcement Learning approach to the traveling salesman problem", *In Proceedings of the Twelfth International Conference on Machine Learning*, pp. 252-260, 1995.
29. Gendreau, M., G. Laporte, F. Semet, "A branch-and-cut algorithm for the undirected selective traveling salesman problem", *Networks*, Vol. 32, pp. 263-273, 1998.
30. Goemans, M. X., and D. P. Williamson, "A general approximation technique for constrained forest problems", *In Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, Chapter 37, pp. 307-316, 1992.
31. Golden, B., L. Levy, R. Vohra, "The Orienteering Problem", *Naval Research Logistics*, Vol. 43, pp. 307-318, 1987.
32. Hozefa M. Botee, Eric Bonabeau, "Evolving Ant Colony Optimisation", *Advances in Complex systems*, Vol. 1, pp. 149-159, 1998.
33. Keskindürk, T. and H. Söyler, "Global Karınca Kolonisi Optimizasyonu", *J. Fac. Eng. Arch. Gazi Univ*, Vol. 21, No. 4, pp. 689-698, 2006.
34. Kuehn A. A. and M. J. Hamburger, "A heuristic program for locating warehouses", *Management Science*, Vol. 9, No. 4, pp. 643-666, 1963.
35. Liang, Y.-C., A. E. Smith, "An Ant Colony Approach to the Orienteering Problem", *Journal of the Chinese Institute of Industrial Engineers*, Vol. 23, No. 5, pp. 403-414, 2006.
36. Liang, Y.-C., S. Konak, A. E. Smith, "Metaheuristics for the Orienteering Problem", *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 1, pp. 384-389, 2002.
37. Liangjun, K., C. Archetti, Z. Feng, "Ants can solve the team orienteering problem", *Computers & Computers & Industrial Engineering*, Vol. 54, No. 3, pp. 648-665, 2008.

38. Mittenenthal J. and C. F. Noon, "An Insert/ Delete Heuristic for the Travelling Salesman Subset-Tour Problem with One Additional Constraint", *The journal of the Operation Research Society*, Vol. 43, No. 3, pp. 277-283, 1992.
39. Mladenovic, N. and P. Hansen, "Variable Neighborhood Search", *Computers and Operations Research*, Vol. 24, No. 11, pp. 1097-1100, 1997.
40. Ramalhinho, H. and D. Serra, "Adaptive Approach Heuristics for the Generalized Assignment Problem", *Economic Working Paper Series, Universitat Pompeu Fabra, Dept. of Economics and Management, Barcelona, Spain*, No. 304, 1998.
41. Ramesh, R., K. M. Brown, "An efficient four-phase heuristic for the generalized orienteering problem", *Computers and Operations Research*, Vol. 18, No. 2, pp. 151-165, 1991.
42. Randal, M. and A. Lewis, "A Parallel Implementation of Ant Colony Optimization", *Journal of Parallel and Distributed Computing*, Vol. 62, pp. 1421-1432, 2002.
43. Stutzle, T. and H. H. Hoos, "MAX-MIN Ant System", *Journal of Future Generation Computer Systems*, Volume 16, pp. 889-914, 2000.
44. Tang, H., and E. Miller-Hooks, "A TABU search heuristic for the team orienteering problem", *Computers and Operations Research*, Vol. 32, pp. 1379-1407, 2005.
45. Tasgetiren, F.M. and A. E. Smith, "A genetic algorithm for the orienteering problem", *Evolutionary Computation, San Diego, CA.*, Vol. 2, pp. 910-915, 2000.
46. Tsai, C.-F., C.-W. Tsai, C.-C. Tseng, "A new hybrid heuristic approach for solving large traveling salesman problem", *Information Sciences*, Vol. 166, pp. 67-81, 2004.
47. Volgenant T. and R. Jonker, "On Some Generalizations of the Traveling-Salesman Problem", *The Journal of the Operational Research Society*, Vol. 38, No. 11, pp. 1073-1079, 1987.

48. Wade A. and S. Salhi, "An Ant System Algorithm for the Vehicle Routing Problem with Backhauls", *Proceedings of the 4th Metaheuristics International Conference, Porto, Portugal*, pp. 199–203, 2001.
49. Wang C.-X., D.-W. Cui, D.-S. Wan and L. Wang, "A Novel Genetic Algorithm Based on Gene Therapy Theory", *Transactions of the Institute of Measurement and Control*, Vol. 28, No. 3, pp. 253–262, 2006.
50. Wang, C.-X., D.-W. Cui, Z.-R. Wang, and D. Chen, "A Novel Ant Colony System Based on Minimum 1-Tree and Hybrid Mutation for TSP", *Springer Berlin / Heidelberg*, Vol. 3611, pp. 1611–3349, 2005.
51. Wang, Q., X. Sun, B. L. Golden, J. Jia, "Using artificial neural networks to solve the orienteering problem", *Annals of Operations Research*, Vol. 61, pp. 111–120, 1995.
52. Watkins, J. C. H., and P. Dayan, "Technical Note: Q-learning", *Machine Learning*, Vol. 8, No. 3–4, pp. 279–292, 1992.
53. Yang, H., L.S. Kang and Y. P. Chen, "A Gene-Based Genetic Algorithm for TSP", *Chinese Journal of Computers*, Vol. 26, No. 12, pp. 1753–1758, 2003.