ANT COLONY OPTIMIZATION FOR THE SINGLE MODEL U-TYPE ASSEMBLY LINE BALANCING PROBLEM

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Arda Alp
January, 2004

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. İhsan Sabuncuoğlu (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Erdal Erel

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Mehmet Taner

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of the Institute of Engineering and Science

ABSTRACT

ANT COLONY OPTIMIZATION FOR THE SINGLE MODEL U-TYPE ASSEMBLY LINE BALANCING PROBLEM

Arda Alp
M.S. in Industrial Engineering
Supervisor: Prof. Dr. İhsan Sabuncuoğlu
January, 2004

The assembly line is a production line in which units move continuously through a sequence of stations. The assembly line balancing problem is the allocation of tasks to an ordered sequence of stations subject to the precedence constraints with the objective of minimizing the number of stations. In a U-line the line is configured into a U-shape topology.

In this research, a new heuristic, Ant Colony Optimization (ACO) meta-heuristic, and its variants are proposed for the single model U-type assembly line balancing problem (UALBP). We develop a number of algorithms that can be grouped as: (i) direct methods, (ii) modified methods and (iii) methods in which ACO approach is augmented with some metaheuristic.

We also construct an extensive experimental study and compare the performance of the proposed algorithms against the procedures reported in the literature.

Keywords: U-type assembly line balancing problem, Ant Colony Optimization meta-heuristic.

ÖZET

TEK MODELLİ U-TİPİ MONTAJ HATTI DENGELENMESİ PROBLEMİ İÇİN KARINCA KOLONİSİ OPTİMİZASYONU

Arda Alp
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. İhsan Sabuncuoğlu
Ocak, 2004

Montaj hattı ürünlerin sıralı olarak istasyonlardan geçtiği bir üretim hattıdır. Tek modelli montaj hattı dengelenmesi problemi istasyon sayısının en küçüklenmesi amacına yönelik olarak işlerin öncüllük kısıtı dikkate alınarak sıralı istasyonlara atanmasıdır. U-tipi hatta ise üretim hattı U şeklinde düzenlenmiştir.

Bu çalışmada yeni bir sezgisel olan Karınca Kolonisi Optimizasyonu (KKO) ve çeşitleri U-tipi montaj hattı dengelenmesi problemi için önerilmiştir. Geliştirilen algoritmalar üç grup altında incelenebilir: (i) direkt metodlar, (ii) modifiye edilmiş metodlar ve (iii) KKO yaklaşımının diğer bazı sezgisellerle beraber kullanıldığı metodlar.

Ayrıca kapsamlı bir deneysel çalışma yapılmış ve önerilen algoritmaların performansı literatürdeki diğer metodlarla karşılaştırılmıştır.

Anahtar sözcükler: U-tipi montaj hattı dengelenmesi problemi, Karınca Kolonisi Optimizasyonu meta-sezgiseli.

To my family,

Acknowledgement

I would like to express my deepest gratitude to my supervisor Prof. Dr. İhsan Sabuncuoğlu for his instructive comments in the supervision of the thesis and also for all the encouragement and trust during my graduate study.

I am also indebted to Prof. Dr. Erdal Erel for his invaluable guidance, recommendations and everlasting interest for this research and for my future work.

I would like to express my special thanks and gratitude to Asst. Prof. Dr. Mehmet Taner for showing keen interest to the subject matter, for his remarks, recommendations and accepting to read and review the thesis.

I am grateful to Asst. Prof. Bahar Yetiş Kara for her understanding and her recommedations and also to Asst. Prof. Oya Karaşan for her suggestions and her guidance.

I would like to express my deepest thanks to Banu Yüksel for all her encouragement and academic support. I would like to extend my sincere thanks to Savaş Çevik, Nur Beğen, Emrah Zarifoğlu and Pınar Tan. Their endless morale support and friendship during all my desperate times, makes me to face with all the troubles.

Finally, I would like to express my gratitude to my family for their love, understanding, suggestions and their endless support. I owe so much to my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The *assembly line* is a production line in which the units move continuously through a sequence of stations where the assembly operation is performed. Typical examples of these assembly lines are car assembly, electronic appliances and computer assemblies. The first example of an assembly line is credited to Henry Ford who developed such a line and produced Ford automobiles in 1913. However after 1913, for over 40 years only trial-and-error methods were used for balancing lines. The first analytical statement of the assembly line balancing problem was made by Salveson and it was published in 1955 (Salveson, 1955). Many researchers became interested in assembly line balancing after the 1950s.

In todays business environment, demand for products fluctuates significantly, and is very difficult to forecast. It is especially difficult for the mass production line to quickly respond to the fluctuating demand. The design of such a line requires grouping of tasks into stations such that line efficiency is maximized. This problem is known as the simple assembly line balancing problem. Traditionally, these lines are arranged in a straight line. However, as a consequence of the just-in-time production principles, recently many lines are

being arranged in a U-line. Arranging the stations in a U-line has several advantages over the traditional configuration (Scholl and Klein, 1999). Demand fluctuations can be tackled easily by the U-line relative to the straight-line version due to increased search space. Thus, there are more possibilities for grouping tasks into stations on a U-line.

As stated by Scholl and Klein (1999), the traditional type of 'straight' assembly lines have some problems. These are: monotone and boring type of work, low-level skilled, unmotivated operators, inflexibility of the production system concerning failures, the sensitivity to changing demand rates, and large inventories due to rigid output rates. In order to overcome these problems, many firms nowadays incorporate JIT principle and group technology into assembly line production, and these modern assembly lines are often organized as a 'U-line'.

The U-line balancing problem considered in this thesis is the U-shaped line with constant operation times, no waiting times, and no walking times. The objective is to find a proper allocation of tasks to the stations that require minimum number of stations. Every station processes only one item in a given cycle time. Cycle time is defined as the time interval between two successive outputs. The sum of all necessary operation and processing times are intended to be equal among the stations. This is a synchronous process of items flowing through the stations and no items exist between adjacent stations. This concept is called as *a single-unit production and conveyance* ("ikko-nagashi," in Japanese).

In (JIT) production system, this concept is applied to a production line with conveyors. U-lines are balanced again when production requirements change. This is much easier with U-lines than with traditional straight lines. This requires operators to be multi-skilled to operate several different machines or processes. To accomplish this goal with a low production cost, a U-shaped layout with multi-function workers is used. As seen in Figure 1.1, this multi-function worker is

responsible from multiple tasks. In a U-shaped layout, stations are organized in such a way that the same worker can handle tasks, which are located both at the entrance side and at the exit side. First worker handles tasks both at the entrance and the exit. A new item can enter the system only after one product is completed. Thus work-in-process in the system stays constant in these systems.

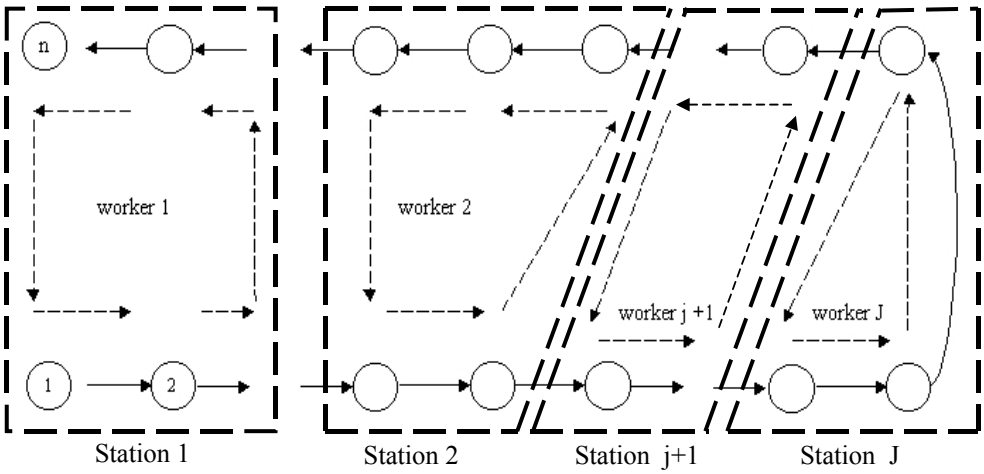


Figure 1.1: U-shaped line with multi-function workers.

When compared with traditional layout, fewer workers are allocated to machines in the U-shaped layout. Thus this type of allocation is more effective when demand fluctuates since it is easier to reallocate workers in order to balance the cycle time of workers. This is the reason why U-shaped layout allows for adapting to the changes in the circumstances more easily than the traditional layout. Miltenburg and Wijngaard (1994) note advantages of U-lines over straight lines as follows:

- Visibility and communication are improved because operators work close to each other.
- Multiskilling allows more operators to understand the relationships between operations and participate in efforts to improve the process.

- The output rate of a U-line can be adjusted by adding or removing workers. While, rebalancing on a traditional line is more difficult because of its low-skilled operators.
- The number of stations required on a U-line is less than or equal to that required on a traditional line. For a U-line there are more alternatives for grouping tasks.

The line balancing problem in a U-shaped line is more complex than balancing a traditional straight line because there are more possibilities to group the tasks while moving forward, backward or simultaneously in both directions. Based on this fact, the number of stations required on a U-line is never more than that required on a traditional line.

Scholl and Klein (1999) define the U-type assembly line balancing problem (UALBP) as an extension of the single assembly line balancing problem (SALBP) with respect to the precedence constraints. The authors describe three problem versions:

- UALBP-1: Minimize the number m of stations, given the cycle time c.
- UALBP-2: Minimize the cycle time c, given the number m of stations.
- UALBP-E: Maximize the line efficiency E for c and m being variable.

Most of the characteristics of SALBP defined by Baybars (1986) are also valid for UALBP. These are:

- A single product is manufactured in large quantities. The task $j = 1,\ldots, n$ takes deterministic operation times t_j , and t_{sum} denotes the sum of all operation times.
- The tasks are partially ordered by precedence relations. These relations defined as *precedence network* with the tasks denoted by nodes and the precedence relations denoted by directed arcs. Therefore, for an arc *(i, j)*, task i must be finished before task j can be started. In Figure 1.2, an example of a

precedence network is given. The task numbers are written on the nodes and the duration of operations are written as weight of nodes.

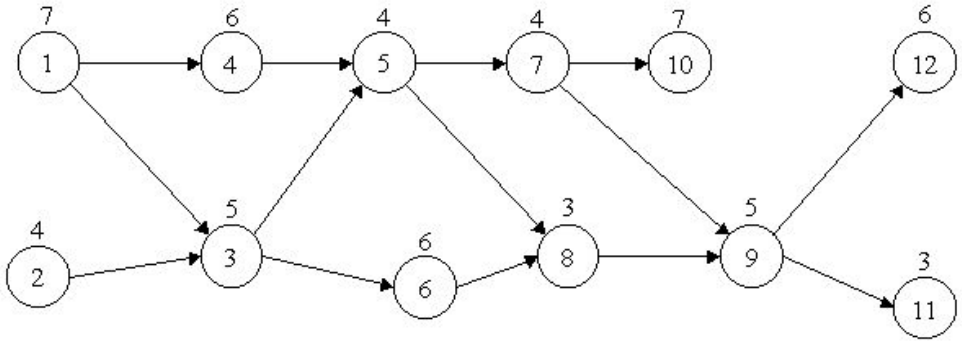


Figure 1.2: Example of a precedence graph. Scholl and Klein (1999)

- Each task can only be assigned to only one station. S_k denotes the set of tasks assigned to station $k = 1,\ldots, m$.
- $T(S_k)$ denotes the station time, the total operation time of tasks assigned to station k, and must not exceed the cycle time:

$$T(S_k) = \sum_{j \in S_k} t_j \leq c \quad k = 1,...,m \tag{1.1}$$

In SALBP, all direct or indirect predecessors of task j, performed at station k, have to be assigned to one of the stations 1,…, k. A task and its indirect predecessors or successors can share the same station only if all intermediate tasks defined with this precedence relationship are also in the same station. For example, in Figure 1.2, task 3 and task 8 can only be at the same station if tasks 5 and 6 are also assigned to the same station.

In UALBP, each task and any of its predecessors and/or successors can share the same station but it must be satisfied that, all predecessors and/or successors of a task j, performed at station k, have to be assigned to one of the stations 1,…, k. (Miltenburg and Wijngaard, 1994).

Scholl and Klein (1999) state that the optimal line efficiency of a SALBP instance is a lower bound on the optimal line efficiency, *E*, of the corresponding UALBP instance due to above given relaxed precedence constraints. Thus a higher efficiency is possible with UALBP (Line efficiency directly related with smoothness of station utilization). The line efficiency is defined with the following formulation:

$$E = \frac{t_{sum}}{m \times c} \times 100\,\% \tag{1.2}$$

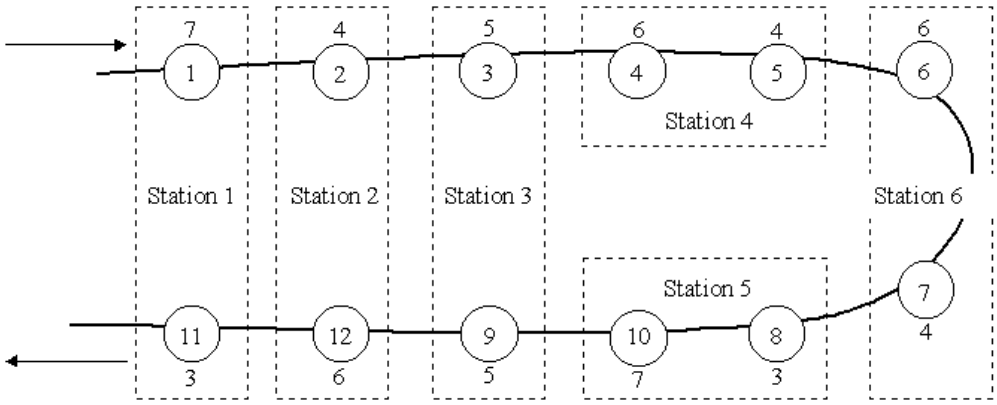


Figure 1.3: Solution of example problem for *c*=10. Scholl and Klein (1999)

In Figure 1.3 the optimal solution of UALBP-1 for c = 10, with 6 stations is given. In station 3, S_3 = {3, 9}, the tasks 3 and 9 are processed. A unit product crosses the station 3 from left to right (the task 3 is performed at that time) after its predecessors have been processed in station 1 and 2, respectively and returns to station 3 from right to left (the task 9 is performed at that time) after all the predecessors of task 9 have been processed in stations 1-6. Then the successors of task 9 are performed in stations 2 and 1, respectively.

When Scholl and Klein's (1999) optimal solution of for SALBP-1 (S_1 = {1}, S_2 = {2, 4}, S_3 = {3, 5}, S_4 = {6, 7}, S_5 = {8, 10}, S_6 = {9, 11}, S_7 = {12}; is compared with the optimal solution of UALBP-1, it is seen that the line efficiency of the U-line is 100% (t_{sum} = 60, $E = \frac{60}{6 \times 10} \times 100$) and the line efficiency of the straight line is 85.7% (t_{sum} = 60, $E = \frac{60}{7 \times 10} \times 100$).

In order to find a simple lower bound on minimal number of stations for UALBP-1 and SALBP-1 Scholl and Klein (1999) define $LB = \lceil t_{sum} / c \rceil$ where $\lceil x \rceil$ is the smallest integer larger than x. For the given example the LB is 6.

We consider the U-line balancing problem such that it is the U-shaped line with constant operation times, no waiting times, and no walking times. Our objective is to minimize the number of stations, given the cycle time c. This can be achieved by finding a proper allocation of tasks to the stations that require minimum number of stations.

In this thesis, a new heuristic, an Ant Colony Optimization (ACO) meta-heuristic, and its variants are proposed for the single model U-type assembly line balancing problem (UALBP). In fact, there are two ant algorithms proposed for the single assembly line balancing problem (Bautista and Pereira, 2002; McMullen and Tarasewich, 2003). However, this study is the first application of ACO meta-heuristic to U-shaped production lines. The work by McMullen and Tarasewich (2003) considers only six problems (ranging in size from 21 to 74 tasks) and their objective function is different from our objective function. Bautista and Pereira (2002) consider the same objective function of minimizing the number of stations given a fixed cycle time, but this model is proposed for only the single assembly line balancing problem (SALBP).

Even though several heuristics have been developed for SALBP (Erel and Sarin, 1998) for single model UALBP, there are only three heuristics in the literature. These are: Ranked Positional Weight Technique (RPWT)-based

heuristic (Miltenburg and Wijngaard, 1994), branch and bound based heuristic (Scholl and Klein, 1999) and simulated annealing based heuristic (Erel, Sabuncuoglu and Aksu, 2001).

Since the first ant algorithm developed by Dorigo and colleagues (1991), several variants of the Ant System (AS) have also been proposed in the literature. In general, ACO is an umbrella term for a number of similar metaheuristics: Ant System (AS), Ant System with Elite Strategy (AS_{elite}), Ant System with Ranking (AS_{rank}), Ant Colony System (ACS), *MAX-MIN* Ant System (MMAS) are some of these meta-heuristics.

In this research, we develop a number of algorithms that can be grouped as: (i) direct methods, (ii) modified methods and (iii) methods in which ACO approach is augmented with some metaheuristic. The first group includes algorithms such as AS, AS_{elite}, AS_{rank}, and ACS that is directly applied to UALBP. No modification is done in the structure of the algorithms. The second group includes new methods in which the structure of the algorithms in the first group is modified. The last group includes two specialized methods that ACO is augmented with simulated annealing (SA) and beam search (BS).

AS is the first algorithm used to solve UALBP. Later, we apply AS_{elite}, and ACS that perform better than AS. However, none of the algorithms give sufficient performance for UALBP. Structure of these algorithms, especially AS_{elite}, AS_{rank}, are not suitable for UALBP.

Actually this is related with the topology of the cost function. Hertz and Widmer (2003) state that the topology of the cost function should not be too flat for the heuristics for searching the optimal solution. The cost function can be considered as an altitude with mountains, valleys and plateaus. If the cost function is too flat, it is difficult for the search algorithms to escape from the large plateaus to fall into the valleys. To tackle this problem Hertz and Widmer (2003) suggest

adding a component to the cost function to discriminate the solutions with the same original cost function value.

The algorithms in the second group are modified from the first group of algorithms to tackle this problem. Their *task selection* and *pheromone trail update mechanisms* are totally modified and improved. In general, the performance of the second group is better than the first group.

The third group includes algorithms in which ACO approach is augmented with a metaheuristic. One of them is a modified version of ACS augmented with SA and the other one is a modified version of ACS augmented with beam search (BS). ACS with SA performs poor in terms of computational time. Even for the small problems (Jackson with 11 elements) the computation time ranges between 2.53 hours and 164.63 hours. Even though the structure of beam search is very suitable for ACS, its performance is poor in terms of computational time. It requires excessive amount of time to complete a single tour.

The rest of the thesis is as follows. The relevant literature on the U-type line-balancing problem and the ACO meta-heuristic are given in Chapter 2. Detailed information about the ACO meta-heuristic is given in Chapter 3. This is followed by the structure of the proposed approaches in Chapter 4. Experimental setting is explained in Chapter 5. Computational results are presented in Chapter 6. Conclusions and future research directions are given in Chapter 7.

Chapter 2

Literature Survey

2.1 Ant Colony Optimization Meta-Heuristic

Today, heuristics are widely used to solve real life problems. Especially, in the last three decades (Zanakis, Evans, Vazacopoulos, 1989), researchers have applied heuristics to produce near optimal solutions to their difficult optimization problems.

Heuristics from nature take inspiration from biology, physics, and social systems. These heuristics utilize some analogies with natural or social systems and use these analogies to develop non-deterministic heuristic methods for NP-hard combinatorial optimization problems (Glover and Greenberg, 1989; Reeves, 1993). Most heuristic algorithms use a problem specific mechanism. Such a mechanism may employs a single agent or more agents (neurons, particles, chromosomes, ants, etc). This agent may operate for a certain number of repeated trials to construct a solution or to improve a given solution. In case of multiple agents, these agents operate with a mechanism of competition-cooperation. In fact, these agents work with a cooperation, however each agent aims to find the best solution and beat the other agents. Some of these algorithms are genetic algorithms (GA), evolution strategies (ES), simulated annealing (SA), tabu search

(TS), neural nets (NN), immune networks (IN), ant colony optimization algorithms (ACO) (Reeves, 1993; Colorni, Dorigo, Maffioli, Maniezzo, Righini and Trubian, 1996; Colorni, Dorigo and Maniezzo, 1992).

The first ant algorithm is proposed by Colorni, Dorigo and Maniezzo (1991 and 1992) and it is named as *Ant System* (AS). It is a multi-agent approach, a class of distributed algorithms for combinatorial optimization. Like other ant type heuristics the main characteristics of this heuristic is simulating or imitating the behaviour of a group of ants. These ants work cooperatively by using simple communication to solve an optimization problem. First use of AS was to solve the well known Travelling Salesman Problem (TSP). Dorigo, Maniezzo and Colorni (1991a, 1991b) describe their methodology as a combination of distributed computation, positive feedback and constructive greedy heuristic. The authors apply their methodology to the classical TSP that the proposed system quickly provides very good solutions. Colorni, Dorigo and Maniezzo (1991 and 1992) state that this new approach can be used to solve any Combinatorial Optimization Problem (COP). The authors also state that a proper representation must be found as given below: (i) the problem (a graph representation which is suitable for a search by many simple agents); (ii) the autocatalytic process[1]; (iii) the heuristic rule that acts as a greedy force and allows a constructive definition of the solution, (iv) the constraint satisfaction method or tabu list. The authors apply AS to the problems such as Satisfiability (SAT), Quadratic Assignment (QAP) and Job-Shop Scheduling (JSP) by using representation rules.

Later, Dorigo, Maniezzo and Colorni (1996) apply AS to the classical TSP. They discuss the parameter selection process and compare AS with TS and SA on TSP. They also illustrate how the AS can be applied to other optimization problems (asymmetric TSP, QAP, JSP). Finally they discuss important

[1] Due to Dorigo, an autocatalytic process (ex: positive feedback) is a process that reinforces itself and causes very rapid convergence. If no limitation is given, this leads to combinatorial explosion. Dorigo, M., Maniezzo, V., Colorni, A. (1996)

characteristics of proposed methodology in terms of global data structure, distributed communication and probabilistic transitions. The authors state its main characteristics as (i) positive feedback (that contributes AS to rapidly discover better solutions), (ii) distributed computation (that provides AS to avoid early convergence), (iii) the use of constructive greedy heuristic (that helps AS to find acceptable solutions in the early stages).

In a later study, Dorigo, Caro, and Gambardella (1999) provide an overview of the recent work on ant algorithms. They give detailed information about biological findings on real ant colonies and define the ants' artificial counterpart the ACO meta-heuristic. The authors also list the applications to other combinatorial optimization problems (Table 2.1 and Table 2.2).

AS is applied to small instances of TSP with up to 75 cities. In spite of encouraging results, in general AS can not compete with state-of-the-art algorithms designed for large TSP instances. However, it has stimulated further research on its different variants that these new algorithms produce much better results on different optimization problems. A considerable amount of research has focused on ACO algorithms. In general, ACO algorithms yield better performance than AS. For that reason, the ACO is usually proposed as a common, unifying framework for the existing applications and algorithmic variants (Dorigo and Stützle, 2000). Next we focus on the variants of ACO meta-heuristic.

Several improvements and variants of the basic AS algorithm proposed in the literature. These improved versions have been mainly tested on the TSP. Indeed, these versions differ in how their search mechanism is controlled. Moreover, the best performing ACO algorithms for the TSP improve the solutions of ants using local search algorithms.

In their work; Gambardella and Dorigo (1995), Dorigo and Gambardella (1996) focus on some properties of Ant-Q (Ant-Q is an extension of AS and Q-learning algorithm, a distributed approach based on reinforcement learning. It is the first and only application of a Q-learning technique to a COP), its sensitivity to parameters, and investigation of synergistic effects when more than a single ant is used. The number of agents used makes Ant-Q different from Q-learning. Ant-Q uses a set of cooperating agents to explore the state space whereas Q-learning works with a single agent. The authors compare Ant-Q with AS, GA, evolutionary programming (EP) and SA, and they state that the covered set of problems are efficiently solved by Ant-Q. Moreover, Ant-Q outperforms AS and on the average Ant-Q is always very close to the optimal solution. Also in the comparison of average behaviour of Ant-Q with the following heuristic methods: Elastic Net (EN), SA, Self Organizing Map (SOM) and Farthest Insertion (FI), Ant-Q is almost always the best performing algorithm.

In their paper, Gambardella and Dorigo (1996) represent Ant Colony System (ACS) as an extension of AS with Q-learning. ACS finds its ground in AS and Ant-Q. The authors state ACS as a revisited version of Ant-Q where a different way of updating ants' experience is discussed. With this approach, it is aimed to improve the system performance in terms of speed and quality by using a different local updating policy. Results show that ACS finds good solutions to symmetric and asymmetric TSP.

In another study, Dorigo and Gambardella (1997a, 1997b) work on Ant Colony System (ACS) and try to understand its operation. Their work includes detailed information on ACS. Their results indicate that ACS outperforms other nature-inspired algorithms such as simulated annealing (SA) and evolutionary computation (EC). They also compare ACS-3-opt (a version of ACS augmented with a local search procedure based on the 3-opt neighborhood) with some of the

best performing algorithms for symmetric and asymmetric TSP. The authors show that ACS is an interesting approach to parallel stochastic optimization of TSP. Also it looks like a very good constructive heuristic to provide a starting solution for the local optimizers. They also compare the performance of ACS with the performance of other naturally inspired meta-heuristics: SA, GA, NN, EC, EP, EN, SOM, FI, and some of their combinations.

Botee and Bonabeau (1998) apply ACO to the TSP and use a GA in order to find the best set of parameters for ACO. They also analyze how the parameters scale with problem size and tour length. Botee and Bonabeau (1998) report that tuning the parameters of the ACO algorithms with an automated search results in better solutions and savings from the computation time.

In another study, Stützle (1998) develop an ACO method, *MAX-MIN* Ant System, for the Flow Shop Problem (FSP). In their previous research, Stutzle and Hoos (1997) and Dorigo and Gambardella (1997) determine that a local search procedure can improve the solution of each ant. Stützle (1998) use a fast local search procedure and show that this approach yields high quality solutions to the FSPs in a short time. This approach performs better or at least comparable to other state-of-art algorithms proposed for the FSP.

Bullnheimer, Kotsis, and Strauss (1998) find the structure of AS highly suitable for parallel implementation of the algorithm. They develop two parallelization strategies and analyze the factors that have influence on computational complexity. They also dwell upon the design parameters and compare the performance of their parallelization strategies.

Later, Stützle and Dorigo (1999) give an overview of the ACO algorithms available for the TSP. They outline how ACO algorithms can be applied to that problem, present the available applications for the TSP, and briefly discuss local search applications to the TSP. Stützle and Dorigo (1999) work with *MAX-MIN*

Ant System (MMAS is one of the improved versions of AS). MMAS can find high quality solutions for all instances and it yields better average solutions than the iterated local search algorithm (ILS is one of the best algorithms for the TSP).

Bullnheimer, Hartl and Strauss (1999) introduce a new rank based version of the AS, called as AS_{rank}. The authors compare this new version with AS, AS_{elite}, SA and GA on several TSP instances. Their results indicate that AS compete well with the two meta-heuristics and it outperforms the other methods for large problems in terms of average and especially the worst case behaviour.

Besten, Stutzle and Dorigo (2000a, 2000b) present an application of the ACO metaheuristic to the single machine total weighted tardiness problem. The authors introduce a simple but very effective local search and combine it with the constructive phase of ACO. Thus they obtain a new ACO algorithm that uses a heterogeneous colony of ants. The authors state that this new algorithm is highly effective in finding the optimal or the best-known solutions on all instances of benchmark problems in ORLIB within reasonable computation times.

In another study, Fenet and Hassas (2000) propose a new problem-solving framework, A.N.T. This method employs mobile reactive agents for distributed problem solving (on different machines) and remote control. This distributed mechanism leads to improvement in the complex collective behaviour based on local interactions of ants.

In their overview Maniezzo and Carbonaro, (2001) compare the ACO approach with other metaheuristics (SA, TS, GA and GRASP) for COP. They focus on ANTS metaheuristic, which is an extension of AS. The ANTS tested on the quadratic assignment and frequency assignment problem (QAP and FAP, respectively). The results indicate that ANTS is the best performer among the algorithms, both considering the best and the average quality of the solutions

proposed for QAP. Also, ANTS is competitive with the best approaches developed for FAP.

In another study, Gagné, Gravel and Price (2001) add a look-ahead mechanism to the ACO algorithm and test their method on an industrial scheduling problem. The look-ahead mechanism allows combining information on the expected decisions, which are beyond the immediate choice horizon. The results indicate that the look-ahead information improves the solution quality, but increases the computation time.

Montgomery and Randall (2002) work on alternative pheromone representations for ACO. They propose three different alternatives for structuring and using pheromone. Their results on TSP indicate that memory requirements decrease but these alternatives are not as effective as ACO. Montgomery and Randall (2002) state that if pheromone representation matches closely with the problem representation, then better results are obtained.

Blum and Samples (2002) deal with the FOP Shop scheduling problem (First Order Parallel Shop scheduling; a general scheduling problem including Job Shop scheduling, Open Shop scheduling and Mixed Shop scheduling). They compare different pheromone representations taken from the literature with a new pheromone representation for ACO to solve the FOP Shop scheduling problem. The new pheromone representation results in a clearly improved performance when compared to the known pheromone representations.

In a recent work, Middendorf, Reischle and Schmeck (2002) propose multi colony ant algorithms (MCAA). In MCAA, several colonies of ants cooperatively work to find the better solutions for a given problem by exchanging information about good solutions at certain time steps. The authors state that if the amount of exchanged information is not too large, then MCAA can easily parallelized by placing colonies on different processors. The authors study the behavior of

MCAA by using different kinds of information exchange between the colonies; and they also compare the behavior of multi colony ant algorithms to a single colony ant algorithm. TSP and QAP are the test problems. The authors observe that for the TSP the multi colony approach with a moderate number of colonies is better than a single colony. For QAP, multi colony ant approach is not better but at least not much worse than having only one large colony. The important part is the exchange of information related with good solutions between the colonies.

T'kindt, Monmarche, Tercinet and Laugt (2002) consider the 2-machine flowshop scheduling problem with the objective of minimizing the total completion time and the makespan criteria. The proposed method (SACO) is an ACO approach with additional feature of SA search and local search. Their results indicate that SACO heuristic is effective and yield better results when compared to existing heuristics. Especially for large problems (problems having more than 200 jobs) the SACO is the most efficient heuristic.

Bautisca and Pereira (2002) work on the simple assembly line balancing and generalized assembly line balancing problems and try to minimize the number of stations given a fixed cycle time. The problem is solved using the ACO metaheuristic with different features. In their research, the authors study several trail information management policies and trail information reading techniques. Also new ideas (solving the direct and reverse instance of a problem concurrently) and priority rules are used together. The authors state that after a long computation time the results obtained with proposed algorithms are better than the results obtained with the exact procedure and also these results are very close to the known problem bound.

McMullen and Tarasewich (2003) present a heuristic, based on ant techniques. This heuristic uses concepts derived from ACO techniques. They state that their approach effectively address the assembly line balancing problem with

complicating factors (parallel workstations, stochastic task times, and mixed models). The assembly line layouts obtained by the proposed heuristic are used for simulated production runs in order to collect some output performance measures. These output performance measures are compared to output performance measures obtained from several other heuristics such as SA. The results indicate that their proposed approach is competitive with the other heuristic methods.

Finally, Krishnaiyer and Cheraghi (2002) present an overview of ant algorithms in their paper and they propose a review of ant applications for real life problems faced in business and industrial environments. The applications of ACO algorithms to static and dynamic combinatorial optimization problems are given in Tables 2.1 and Table 2.2.

Table 2.1: List of applications of ACO algorithms to static combinatorial optimization problems. This table is adapted from Dorigo and Stützle (2000), Krishnan and Cheraghi (2002)

Problem Name	**Authors**	**Year**	**Algorithm Name**
Traveling salesman	Dorigo, Maniezzo & Colorni	1991	AS
	Gambardella & Dorigo	1995	Ant-Q
	Dorigo & Gambardella	1996	Ant-Q
	Dorigo & Gambardella	1996	ACS & ACS-3-opt
	Gambardella & Dorigo	1996	ACS
	Stützle & Hoos	1997	MMAS
	Bullnheimer, Hartl & Strauss	1997	AS_{rank}
	Bullnheimer, Kotsis & Strauss	1998	AS & Parallelization
	Botee & Bonabeau	1998	Evolving ACO
	Stützle & Dorigo	1999	AS, ACS, MMAS, ANT_{elite}, ANT_{rank}
	Cordon, et al.	2000	BWAS
	Middendorf , Reischle & Schmeck	2002	Multi Colony Ant Algorithms
	Montgomery & Randall	2002	Alternative pheromone representations for ACS
Quadratic assignment	Maniezzo, Colorni & Dorigo	1994	AS-QAP
	Gambardella, Taillard & Dorigo	1997	HAS-QAP[a]
	Stützle & Hoos	1998	MMAS-QAP
	Maniezzo & Colorni	1998	AS-QAP[b]
	Maniezzo	1998	ANTS-QAP
	Maniezzo & Carbonaro	2001	ANTS-QAP
	Middendorf , Reischle & Schmeck	2002	Multi Colony Ant Algorithms
Scheduling problems	Colorni, Dorigo & Maniezzo	1994	AS-JSP
	Forsyth & Wren	1997	AS
	Stützle	1998	AS-FSP
	Bauer et al.	1999	ACS-SMTTP
	den Besten, Stützle & Dorigo	1999	ACO-SMTWTP
	den Besten, Stützle & Dorigo	2000	ACO-SMTWTP
	Merkle, Middendorf & Schmeck	2000	ACO-RCPS
	Gagne, Gravel & Price	2001	fACO
	Blum & Sampels	2002	ACO-FOP Shop
	T'kindt, Monmarche, Tercinet & Laugt	2002	ACO
Vehicle routing	Bullnheimer, Hartl & Strauss	1997	AS-VRP
	Gambardella, Taillard & Agazzi	1999	HAS-VRP
Sequential ordering	Gambardella & Dorigo	1997	HAS-SOP
Graph coloring	Costa & Hertz	1997	ANTCOL
Shortest common supersequence	Michel & Middendorf	1998	AS-SCS
Frequency assignment	Maniezzo & Carbonaro	1998	ANTS-FAP
	Maniezzo & Carbonaro	2001	ANTS-FAP
Generalized assignment	Ramalhinho Lourenço & Serra	1998	MMAS & GAP

[a] HAS-QAP is an ant algorithm but does not follow all the aspects of the ACO meta-heuristic.
[b] This is version of the original AS-QAP.

Table 2.1: (Cont'd)

Flow manufacturing	Stützle	1998	ACO
Multiple knapsack	Leguizamon & Michalewicz	1999	AS-MKP
Redundancy allocation	Liang & Smith	1999	ACO-RAP
Layout of facilities	Bland	1999	AS (TS)
Space-planning	Bland	1999	ACO
Constrain satisfaction	Solnon	2000	Ant-P-solver
Image segmentation - Pattern reorganization	Ramos, & Almeida	2000	Cognitive map model
Digital Art	Tzafestas	2000	Painter Ants
Numeric Optimization	Monmarche, Venturini & Slimane	2000	API
Structural Design Problem	Bland	2001	ACO
Bioreactors Optimization	Jayaraman, Kulkarni & Gupta	2001	ACO
Pickup and delivery problems	Doerner, Hartl, & Reimann	2001	ACO
Full truck load transportation problems	Doerner, Hartl, & Reimann	2001	ACO
Bus stop allocation problem	Jong & Wiering	2001	Multiple Ant Colony Systems
Peer-to-peer (P2P) networks	Baboglu, Meling, & Montresor	2001	Anthill
Shop floor routing	Cicirello	2001	AC^2
Assembly Line Balancing	Bautista & Pereira McMullen & Tarasewich	2002 2003	ACO ANT1, ANT2, ANT3, ANT4
Distributed Problem Solving	Fenet & Hassas	2000	A.N.T.

Table 2.2: List of applications of ACO algorithms to dynamic combinatorial optimization problems. This table is adapted from Dorigo and Stützle (2000), Krishnan and Cheraghi (2002)

Problem Name	**Authors**	**Year**	**Algorithm Name**
Connection-oriented network routing	Schoonderwoerd, Holland, Bruten & Rothkrantz	1996	ABC
	White, Pagurek & Oppacher	1998	ASGA
	Di Caro & Dorigo	1998	AntNet-FS
	Bonabeau, Henaux, Guerin, Snyers, Kuntz & Theraulaz	1998	ABC-smart ants
Connection-less network routing	Di Caro & Dorigo	1997	AntNet & AntNet-FA
	Subramanian, Druschel & Chen	1997	Regular ants
	Heusse, Guerin, Snyers & Kuntz	1998	CAF
	van der Put & Rothkrantz	1998	ABC-backward
Optical networks routing	Navarro Varela & Sinclair	1999	ACO-VWP
Dynamic routing in telecommunication networks	Zhou & Liu	1999	Intelligent Ant algorithm

2.2 U-Type Line Balancing

U- type line balancing (ULB) concept is a new and promising topic in the assembly line balancing literature. The literature in this area is accumulated since Monden (1993) first introduced the U-type configuration to the attention of the scientific community. Erel, Sabuncuoglu and Aksu (2001) classify the research into two categories: line balancing and production flow lines. The first group includes the studies for balancing the U-type assembly lines to minimize the number of stations for a given cycle time or minimize the cycle time for a given number of stations. The second group includes the studies to identify the importance of design factors, and their effects on the performance of U-type flow lines. Since this research focuses on the line balancing problem, we refer the reader to the following papers: Nakade *et al.* (1997), Nakade and Ohno (1997, 1999, 2003), Miltenburg (2000, 2001a, 2001b).

The first study in this area is due to Miltenburg and Wijgaard (1994) who analyze the U-line line balancing problem and develop solution procedures. They also show how a solution method (developed for the traditional line-balancing problem) can be adopted to the U-line. They work on twelve well-known sets of line balancing problems taken from the literature. Each problem consists of a number of tasks, task completion times, precedence constraints, and a number of cycle times. Thus each cycle time corresponds to a new problem. (Indeed this data set consists of 61 problems). In order to obtain an optimal balance for the U-line and traditional line, Miltenburg and Wijgaard (1994) propose a dynamic programming (DP) formulation, and solve 21 relatively small problems (up to 11 tasks). The authors also develop a heuristic which is based on maximum ranked positional weight heuristic (RPWT) for the large size problems. They also use standard version of this heuristic to obtain the optimal traditional line balances.

Later, Miltenburg and Sparling (1995) develop three exact algorithms for the ULB problem: a reaching dynamic programming algorithm, a breadth-first branch and bound algorithm, and a depth-first branch and bound algorithm. The authors solve 180 problem instances (up to 40 tasks). The computational experiments indicate that the breadth-first and depth-first B&B algorithms are more efficient than the DP approach.

In another study, Urban (1998) develops an integer programming (IP) formulation to determine the optimal solution for ULB problem. Standard test problems of Miltenburg and Wijngaard (1994) are solved using this model. Urban (1998) considers only the "larger" problems (21 or more tasks). Problems of up to 45 tasks can be solved using proposed model.

To solve large size problems encountered in practice Scholl and Klein (1999) propose a new branch and bound procedure, ULINO, which is adapted from SALOME (previously developed algorithm for traditional straight line balancing problem). SALOME is chosen as the basis of the ULINO, because it has been shown to be very effective in several computational tests (Scholl and Klein, 1999). ULINO is a branch and bound procedure that performs a depth-first search by considering bounds and some dominance rules. Computational experiences with their method are presented for 256 instances (complete data set), problems with up to 297 tasks. The results indicate that the proposed method yields very good results for Type-1 problem, (UALBP-1; minimizing the number of stations given the cycle time) and Type-2 problem (UALBP-2; minimizing the cycle time given the number of stations) in limited computation time. For the most general problem type (UALBP-E; maximizing the line efficiency for variable cycle time and number of stations). Scholl and Klein (1999) suggest a further research to find more efficient solution procedures.

Erel, Sabuncuoglu and Aksu (2001) propose a simulated annealing (SA)-based algorithm for the UALB problem. Their algorithm employs with an intelligent mechanism to search a large solution space effectively. The performance of the proposed method is tested on a large number of benchmark problems from the literature. Their computational results indicate that the proposed method is quite effective and its computational requirements are not as high as expected. The computational success of this SA-based heuristic can be attributed to the intelligent way of searching a larger search space.

Miltenburg (2001) presents a detailed study on the theory and practice of U-shaped production lines. He gives the related history of U-shaped production lines, describes the JIT production environment, the layout and operation of the U-line in an JIT production environment. He also provides useful information for designers to design and manage the U-lines.

Some researchers focus on the mixed-model U-line balancing. In recent years, manufacturers change their production lines from single product or batch production to mixed-model production lines. This is an expected reason related with their objective: implementing the just-in-time principles. Different products or models are produced on the same line in mixed-model production and manufacturers are able to respond their customers with a variety of products in a timely and cost-effective manner. As Sparling and Miltenburg (1998) state, U-lines are widely used for the mixed model production.

Sparling and Miltenburg (1998) study on the mixed model production and the mixed-model U-line balancing (MMULB) problem. MMULB assigns the tasks required producing all models to a minimum number of stations on a U-shaped line. They develop an approximate algorithm to solve the problem. Their algorithm transforms the multi-model problem into an `equivalent' single-model problem and finds the optimal balance to this problem using a branch and bound

algorithm from the literature. The proposed algorithm is capable of solving the single-model problem with up to 25 tasks.

Miltenburg (1998) work on the problem of balancing and rebalancing U-line facilities. A U-line facility is defined as a unit that consists of numerous U-lines connected by multi-line stations. The objective is to assign tasks to a minimum number of stations while satisfying cycle time, precedence, location, and station-type constraints when balancing such a facility. A secondary objective is to concentrate the idle time in a single station. A reaching DP algorithm is proposed to determine optimal balances for the facilities with any number of U-lines. Miltenburg state that this reaching DP algorithm is effective for balancing and rebalancing facilities with any number of U-lines, providing that individual U-lines do not have more than 22 tasks and do not have wide, sparse precedence graphs.

Sparling (1998) also work on U-line facilities. He introduces the concept of a JIT production unit, where a number of U-lines produce and assemble parts for the same production line. Balancing a JIT production unit problem is considered as the N U-line balancing problem. For both cases (for the case where U-line locations are not fixed and for the fixed case) problems are modelled and heuristic algorithms are developed. This heuristic algorithm solves problems with up to nine individual U-lines that each of them having tasks up to 18.

Kim *et al.* (2000) work on balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. They develop a new approach using an artificial intelligence search technique, called as cooperative co-evolutionary algorithm. It is possible to solve the line balancing and the model sequencing problems at the same time with this approach. In order to promote the population diversity and the search efficiency authors adopt some strategies (these strategies are localized evolution and steady-state reproduction) and develop some methods (selection of

environmental individuals and evaluation of fitness). They also provide efficient genetic representations and operator schemes. Based on the experimental results the authors indicate that the proposed algorithm is much better than the other two cooperative co-evolutionary algorithms and the traditional hierarchical approach. They also state that with a little modification, the proposed algorithm can be applied to many variants of the problem.

Also some of the researches work on different but interested subjects to investigate some properties of the U-shaped production lines.

Nakade and Ohno (1999) deal with a U-shaped production line with multiple-function workers. They consider an optimization problem of finding an allocation of workers to the production line. The objective is to maximize the overall cycle time under the minimum number of workers while satisfying the demand.

Miltenburg (2000) investigates the effect of the U-shape of the line on the production line's effectiveness when breakdowns occur. The author finds that the effectiveness of the U-line is greater than or equal to the effectiveness of the straight line when buffer inventories are located between stations.

Miltenburg (2001) works on one-piece flow production system on U-shaped production lines and examines the research literature on one-piece flow manufacturing. Miltenburg state that if implemented carefully, in a situation where it is appropriate impressive results are obtained. In this research the author dwells upon the decision rules that determine when one-piece flow is appropriate, and unique elements of this production system. Also he examines the mathematical models used to design one-piece flow system.

Nakade and Ohno (2003) consider a U-shaped production line with multiple workers. Each worker is a multi-function worker and takes charge of multiple machines. They consider two types of allocations of workers to

machines; a separate allocation and a carousel allocation. By using these allocations they derive some upper and lower bounds for the expected overall cycle times in U-shaped production line and they propose an approximate expressions for the expected overall cycle times. The authors show that when the processing, operation, and walking times are constant, the overall cycle time in the carousel allocation is less than or equal to that in the separate allocation. In their numerical study Nakade and Ohno (2003) compare these allocations and they discuss the performance of their approximation.

Summary of the work done on the U-line line balancing problem is given in Table 2.3.

Table 2.3: Summary of work done on the U-type assembly line problems.

Authors	Methodology	Problem type	Solved problems	Objectives
Miltenburg and Wijngaard (1994)	DP formulation RPWT-based heuristic	single model	up to 11 tasks up to 111 tasks	number of stations
Miltenburg and Sparling (1995)	DP-based exact algorithm Depth-first and breadth-first B&B	single model	up to 40 tasks	number of stations
Sparling and Miltenburg (1998)	Heuristic	mixed model	up to 25 tasks	number of stations
Urban (1998)	IP formulation	single model	up to 45 tasks	number of stations
Miltenburg (1998)	DP-based exact algorithm	U-line facility with several individual U-lines	individual U-lines with up to 22 tasks	number of stations and idle time in a single station
Sparling (1998)	Heuristic	U-line facility with several individual U-lines	up to 9 U-lines with up to 18 tasks in each U-line	number of stations
Scholl and Klein (1999)	B&B-based heuristic	single model	up to 297 tasks	number of stations, cycle time, and both
Kim *et al.* (2000)	Co-evolutionary algorithm	mixed model	up to 111 tasks	number of stations
Erel, Sabuncuoglu and Aksu (2001)	SA-based heuristic	single model	up to 297 tasks	number of stations

Chapter 3

Ant Algorithms and Applications

This chapter focuses on pure ant algorithms. Most parts are writen with the guide of Dorigo, Caro, and Gambardella's (1999) paper (This paper overviews the recent work on ant algorithms, gives detailed information about the biological findings on the real ant colonies and defines ants' artificial counterpart the ACO meta-heuristic. Their paper mainly focuses on the most important aspects of the ACO meta-heuristic. It is a very detailed study including every aspect of ant algorithms and can be stated as a sort of handbook).

3.1 Introduction

Ant heuristics are proposed as a new approach to combinatorial optimization in the literature (Dorigo, Maniezzo and Colorni, 1991a, 1996). Their main characteristics are: *positive feedback* (that helps to discover better solutions), *distributed computation* (that avoids early convergence), and the use of *constructive greedy heuristic* (that helps ants to find acceptable solutions in the early stages.

This new heuristic attracts the attention of the scientific community because it is *versatile* (which is applicable to similar versions of the same problem), *robust* (with only minimal changes it can be applied to other combinatorial optimization problems), and *a population based approach* (which allows the exploitation of positive feedback as a search mechanism). This last property makes these systems suitable for parallel implementation. Detailed information on the parallelization strategies for the Ant System can be found in Bullnheimer, Kotsis, and Strauss (1998).

The first ant algorithm is developed by Dorigo and colleagues (Dorigo, Maniezzo, Colorni, 1991) to difficult combinatorial optimization problems, such as the travelling salesman problem (TSP) and the quadratic assignment problem (QAP). Since then many researchers work on ant-based algorithms to apply ant algorithms to various discrete optimization problems. These applications and other details of ant colony optimization algorithms (ACO) are summarized in Table 2.1 and Table 2.2.

3.2 Biological Fundamentals

Similar to other nature based heuristics (such as genetic algorithms, evolution strategies, simulated annealing, tabu search, neural nets, immune networks) ant algorithms are 'derived' from nature and inspiration comes from the observation of real ant colonies. The behaviour of ants in the colony constitutes the main power of the ant algorithms. Members of the colony do not act selfishly, only by focusing on their individual needs. On the contrary, everyone acts cooperatively, as a whole in order to provide the survival of the colony. Ants have no chance to survive when they are alone but when they form a group, then they are more structural and their colonies can survive. This surviving mechanism and their foraging behaviour have captured the attention of many scientists. Foraging behaviour is an important and interesting behaviour. It is an ability to find the shortest paths between the food sources and their nest.

On the way between the food source and the nest, ants deposit on the ground a substance called *pheromone* which forms a trail of pheromone. Ants have ability to detect (smell) the pheromone and chose their way according to their detection level. This means the probability of choosing one of the possible ways depends on the level of pheromone that is deposit on this way. By using this pheromone trails the ants find their way back to the nest or the food source and also other member of the colony detects the location of the food sources by following the trails of the previous ants. Many paths are available between the nest and the food source. When this pheromone trail following behaviour is exploited by ants then a group of ants are able to find the shortest path between the nest and the food source by following the previously laid pheromone trails of the individual ants.

Deneubourg et al. (1990) study the ants' foraging behaviour by using an interesting experiment, 'the binary bridge experiment'. This experiment is done

under controlled conditions. As seen in Figure 3.1a, a double bridge in which each branch has the same length is put to separate the nest of a colony of ants and a food source. Then, the ants are left free to move between the nest and the food source. During the experiment the percentage of ants which choose one or the other of the two branches is observed over time. The result is given in Figure 3.1b. After an initial transition phase, ants tend to converge on the same path. It is stated that arising of some oscillations during the transition state is normal.

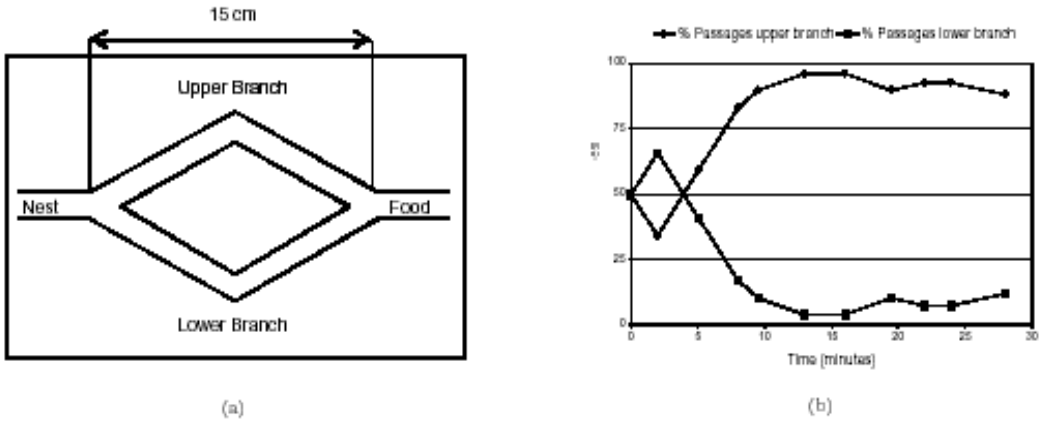


Figure 3.1: Single Bridge Experiment. a)Experiment setup and b) Results for a typical single trial; showing the percentage of passages on each of the two branches per unit of time as a function of time. After an initial short transition phase, the upper branch becomes the most used. Deneubourg et al., (1990)

In the first stage of the experiment the selection probability of any branch is equal to each other because initially no pheromone is left. During the transition phase some fluctuations occur because the amount of trail accumulated is not enough yet to direct the ants. After an initial transition phase the number of ants that randomly select one passage (let it be the upper passage) increases and more pheromone accumulates on this passage over the other. As more pheromone accumulates for the upper passage then there is more chance for an ant to choose that way. This attitude continues as a vicious cycle.

Dorigo, Caro, and Gambardella (1999) describe this phenomenon with a probabilistic model. The authors state that this model closely matches with the

experimental observation. The amount of pheromone on a branch is proportional to the number of ants that used that branch in the past and the pheromone evaporation is not taken into account. In this model, the probability of choosing a branch at a certain time depends on the total amount of pheromone accumulated on the branch. Thus the probability is proportional to the number of ants that used the branch until that time. Let U_m and L_m be the numbers of ants that have used the upper and lower branch after m ants have crossed the bridge, with $U_m + L_m = m$. The probability $P_U(m)$ which the $(m+1)^{th}$ ant chooses the upper branch is given as:

$$P_U(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h} \tag{3.1}$$

and the probability $P_L(m)$ that ant chooses the lower branch is $P_L(m) = 1 - P_U(m)$. This form of the probability is obtained from experiments on trail-following. The parameters h and k allow the user to fit the model to experimental data. Then U_{m+1} is updated as follows: $U_{m+1} = U_m + 1$, if $\psi \leq P_U$, $U_{m+1} = U_m$ otherwise, where ψ is a random variable uniformly distributed over the interval (0,1). L_m is also updated by the same way.

Another experiment is done by preserving the same pheromone laying mechanism and using the bridge with branches of different lengths. Mostly the shortest branch is selected in this case. This result can be explained by the pheromone laying mechanism: the first group of ants on the way to the food source do not have chance to use the pheromone advantage and they choose the shortest way arbitrary. But on the way to the nest those ants that took the shortest branch will cause much pheromone to be accumulated on the short branch when compared with the long branch. For a same unit of time more ant pass through the short branch and cause more pheromone to be laid on the ground. This mechanism and the result of the double bridge experiment with branches of different length is given in Figure 3.2.

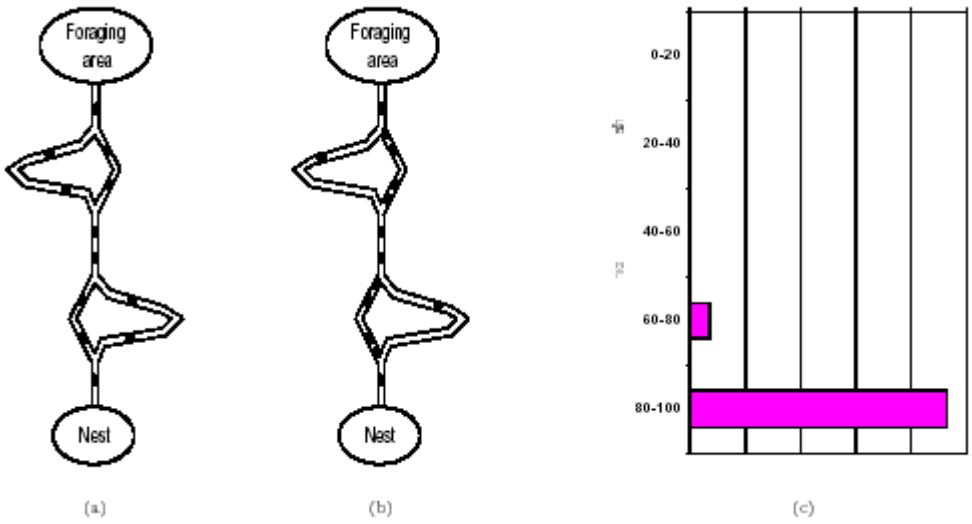


Figure 3.2: Double Bridge Experiment.
a) Ants start exploiting the double bridge, b) As a result most of the ants choose the shortest path.
c) Distribution of the percentage of ants that selected the shorter path. Goss et al. (1989)

Dorigo, Caro and Gambardella (1999) explain this process as a 'distributed optimization mechanism'. A single ant is capable of finding a path between the nest and the food source. Nevertheless, each single ant gives only a very small contribution. On the contrary, with the corporation of ants only a colony can present the "shortest path finding" behaviour. The ants perform this specific behaviour by using a simple form of indirect communication with the help of pheromone trails. This is known as *stigmergy*[1].

Grasse (1946) defines stigmergy as follows: "stimulation of workers[2] by the performance they have achieved."

Stigmergy is a different style of communication because the information released by the communicating ants is a physical substance. This released

[1] Dorigo, Caro and Gambardella (1999) state that the term stigmergy not only used to explain the behavior of termite societies, the same term is also used to describe indirect communication mechanism observed in other social insects.

[2] There are castes is termite colonies and workers are one of the castes in these termite colonies.

information belonging to a specific path is locally attainable for only the ants which use that specific path or only the neighbourhood of that path.

Dorigo, Caro and Gambardella (1999) state that if appropriate state variables are associated to problem states then the characteristics of stigmergy can easily be extended to the ants. In order to explain this statement, a comparison of real word with the imaginary model may be very helpful. The stigmergetic communication of real ants is accomplished by using the pheromone laid on ground. Similarly artificial agents update the proper pheromone variables that representing the problem states that they have visited. Artificial agents have only local access to these pheromone variables according to the stigmergetic communication model.

Another important aspect of real ants' behaviour is the coupling between the *autocatalytic (positive feedback)* mechanism and the *implicit evaluation of solutions*. The shorter paths (for artificial agents, the solutions with lower cost) will be completed earlier than the longer ones. As a result, more pheromone will be accumulated on these paths. This fact explains the 'implicit evaluation of solutions'. Dorigo, Caro and Gambardella (1999) state that the implicit solution evaluation combined with the autocatalysis can be very effective. This is because of the chain effect that arises spontaneously. The shorter paths will receive pheromone quickly. As more pheromone is deposited on these paths, more ants chose these paths and so on. If properly used the autocatalysis is stated as a powerful mechanism for the population-based optimization algorithms. This mechanism gives extra importance to the best individuals and these individuals direct the search process. However, sometimes the misdirection of the search may be dangerous. A misdirected search may result with premature convergence (stagnation). *Stagnation* is a contingent situation in which some individuals that are not very good take over the population and they dominate the search process.

This situation prevents the further exploration of the search space. There are many reasons of stagnation. Not getting out of a local optima results in giving extra incorrect emphasis to this local solution and stagnation; or initial random fluctuations may cause a not very good individual to be selected more than the other individuals. Thus, this individual becomes much better than all others.

3.3 The Ant Colony Optimization Approach

In the ant colony optimization (ACO) meta-heuristic[3] a group (colony) of agents cooperatively work to find good solutions to difficult optimization problems. Cooperation is the most important part of the ACO algorithms. All the computational resources are divided into many parts and allocated to simple agents. The only way to use that divided computational resource is the indirect communication caused by stigmergy. Thus, good solutions are exposed by the agents using this cooperative interaction.

When the agents and the real ants are compared there are some similarities and dissimilarities. Mostly the agents are the abstraction of the real ants and they have most of the abilities of the real ones, like shortest path finding behaviour. However, they have some extra capabilities that the real ants do not have. These extra capabilities (The agents have some memory. They are not completely blind and they live in an environment where the time is discrete) are given to artificial ants in order to make them more efficient and effective.

[3] Dorigo, Caro and Gambardella (1999) use ACO meta-heuristic to refer to the general procedure. The term ACO algorithm is used to refer any generic instance of ACO meta-heuristic. There are also some algorithms called as Ant algorithms that not necessarily follow all the aspects of the ACO meta-heuristic. So, all ACO algorithms are also ant algorithms, where as vice versa is not true.

3.3.1 Similarities and Difference with Real Ants

Dorigo, Caro, and Gambardella (1999) state the ideas that are inspired from real ants as follows:

(i) Usage of a colony of communicating and cooperating individuals,

(ii) Usage of an artificial pheromone trail for the stigmergetic communication,

(iii) Usage of a sequence of local moves to find the shortest paths,

(iv) Usage of a local information for the stochastic decision policy.

We explain these characteristics as follows.

Colony of communicating and cooperating individuals: Like real ant colonies, the agents are part of a population, which they globally and asynchronously cooperate at the same time to find a better solution. Although a single ant can construct a feasible solution (and a single real ant can also find the path between nest and food source), better solutions are the result of the cooperation between each single individual. The power of this cooperation comes from their communication, namely the exploitation of the information that they read or write on the problem's visited states.

Pheromone trail and stigmergy: Similar to real ants pheromone laying attitude, the agents apply the same behaviour by changing some numeric values of the problem's state that they have visited. With this information, previous ants' performances are stored. This information will be ready for the new agents at the next stage for re-reading or for updating. This numeric information belonging to the problems state variables acts like *artificial* pheromone trail. Thus, communication among the ants can only be proved by this artificial pheromone trails and collective knowledge is evaluated and restructured by this way. This knowledge supplies information about the problem landscape. In fact, this knowledge acts like a function of whole history of the ant colony. Generally the updating mechanism for this information is not only obtained by the accumulation

of feedback coming from the ants but also there is an evaporation mechanism inspired from the nature. By evaporating some amount of the pheromone over time, the ant colony slowly forgets its history and this mechanism helps directing the search to new areas of the search space. Thus, stacking to a specific point or a neighbourhood of a point can be avoided by this way.

Finding the solutions with minimum cost and local moves: The objective of real ants is to find the shortest path joining the nest to the food source. They accomplish this goal by moving on the ground from one point to another, namely they move from one adjacent area to another one. The agents also have a common objective. They try to find the solution with the minimum cost. The agents move through the 'adjacent states' of the problem. They start from an origin and move step by step to a destination point.

Stochastic and myopic state transition policy: The agents construct their solutions by applying a probabilistic decision policy. The decision policy is used while moving from one state to another state. The local information gained from artificial pheromone trails obtains the primary information for this decision policy. There is no lookahead information to predict the future condition of the state (Although, simple forms of ACO algorithms do not use lookahead information, there are some researchers working on this topic and there exist a special version of ant algorithms using the lookahead mechanism). In fact, the transition probability is totally local. The policy is a function of both the a priori information represented by problem specifications and the local modifications in the artificial pheromone trails induced by previous ants.

Some characteristics of agents do not correspond with the characteristics of their counterpart, real ants. These are:

- Agents do not move continuously. In fact, they move from one discrete state to another discrete state. So their environment is a discrete habitat.
- Agents have a kind of memory that stores the information of the ant's past actions.
- Artificial pheromone is a function of the quality of the solution found.[4]
- Agents' pheromone laying mechanism is a little bit different from the real life. The timing in pheromone laying mechanism is problem dependent and generally does not correspond with real ants' behaviour. In most of the ant algorithms the agents update the pheromone trails just after generating a solution.
- Agents may have extra abilities and some special mechanisms can be applied like parallelization (or parallel implementation), local optimization, hybridization, lookahead, backtracking to improve the system effectiveness and efficiency.

3.4 The ACO Meta-heuristic

This section gives information about how the agents are used in an algorithmic framework so that the ACO algorithms can be applied to combinatorial optimization problems.

In ACO algorithms a finite number of artificial ants cooperate and move in the search space to find the good solutions to the optimization problem under some predefined conditions. Each single ant constructs a full solution or only a part of it starting from a predefined initial state. This state is determined by taking problem specific conditions into consideration. As the ants move from one state to

[4] Dorigo, Caro and Gambardella (1999) state that in real life some ants have a similar behaviour. They deposit more pheromone if they found richer food sources.

another they construct their own solution and also they gather some important knowledge about the problem characteristics and their own performance. They use this information to modify the problem-related variables (like trail values). All individual ants move alone by making random choices. Indeed, all ants operate collectively. This is not a direct communication between two or more ant. In fact, it is the common usage of this information, or more literally stigmergetic paradigm as explained before.

Under consideration of problem's constraints, a desired solution is the one with the minimum cost. Although some of the ants may individually find poor solutions, better solutions (or the best) are obtained from a group of many alternative solutions as a result of the global cooperation of these individuals.

Every ant constructs its solution by moving from one state to another adjacent state by applying a stochastic local search policy. Definition of the neighborhood, every ant's memory, gathered pheromone trail information and problem-specific local information direct this search policy.

Every ant's own memory carries information about the history of that ant. This information is used to compute the value of the solutions, the contribution of each move generated by an ant and the most important one, to check the feasibility of the constructed solutions. By using ant's memory and the related information about the local state, some moves that take the ant from a feasible solution to an infeasible one can be avoided.

Both the knowledge of pheromone trails and the problem specific heuristic information constitute the public information. The pheromone information is a global and time dependent information that has an influence on ants' decisions. It is a shared local long-term memory. The quantity and the frequency of pheromone depositing depend on the problem characteristic and the structure of the algorithm. Each ant can lay pheromone just after making a move (updating step-by-step) or

after constructing the whole solution (delayed updating). Also it is possible to use both of these policies at the same time. Generally the step-by-step updating and the delayed updating are stated as mutually exclusive. Generally the amount of pheromone deposed by an ant is proportional to the quality of the solution generated by this ant. Thus, the contribution of better moves to the global information will be higher than the contribution of the worse moves.

By combining the trail information and heuristic values and also list of tabu moves, every ant constitutes its own *decision table* that directs its search. This decision table includes the probability of each possible move for each ant. Stochasticity in the decision mechanism and the evaporation of trail values prevents the ants converging to a same solution or a certain part of the solution space. Of course there is a balance between the exploitation of accumulated knowledge and the exploration of new points in the search space depending on how the stochasticity and the evaporation mechanisms are used. Also the ACO meta-heuristic can be used to combine the ants' decision policy with some extra components. These components are optional and implementation dependent. For instance, one method may be the collecting of some extra information from a global perspective and using this information to deposit additional pheromone information. Other method may be the usage of a problem specific local optimization procedure. Main activities, the stochastic decision policy, the pheromone evaporation and also some extra components need well synchronization.

In Figure 3.3, a general description of the ACO meta-heuristic is given as a pseudo-code. Some parts are optional like extra activities. Also, some parts and the sequence of some activities may differ depending on the problem type and implementation.

```
MAIN PROGRAM  ACO Meta-heuristic
   BEGIN {For MAIN PROGRAM}
     WHILE  (termination criteria is not satisfied) DO
          BEGIN {For WHILE}
               Initialize the system;
             Proceed ants and construct solutions;
               Pheromone evaporation and update;
               Extra activities; {This part is optional and implementation dependent}
           END; {For WHILE}
   END; {For MAIN PROGRAM}


PROCEDURE Proceed ants and construct solutions  {Life cycle of an ant}
  BEGIN {For PROCEDURE}
     Create new ants;
     Initialization of each ant;
     Mem = Call the memory of each ant;
       WHILE  (a complete solution is not reached for predefined number of ants) DO
            BEGIN {For WHILE}
                 Give initial position;
              NonTabu = Read the memory and determine non-tabu tasks;
                 Prob = Compute transition probabilities (Mem, NonTabu, Problem
                                                          Constraints);
                 Next = Determine where to move due to decision policy (Prob, Problem
                                                                  Constraints);
               Move to next state (Next);
                    IF (step-by-step pheromone update policy is used) THEN
                       BEGIN {For IF}
                          Update pheromone level on the visited arc;
                            Update ant's memory;
                       END; {For IF}
                 Update other related values;
            END; {For WHILE}
  END; {For PROCEDURE}


PROCEDURE Pheromone evaporation and update {Delayed pheromone update}
  BEGIN {For PROCEDURE}
     IF (delayed pheromone update policy is used) THEN
          BEGIN {For IF}
             Evaluate each ant's solution;
           Update pheromone level on all visited arcs;
        END; {For IF}
  END; {For PROCEDURE}
```

Figure 3.3: A general description of ACO meta-heuristic. Comments, explanations are given in braces. Necessary actions for related line and conditions for related IF THEN statements are given in parentheses.

3.5 Some Applications of ACO Algorithms

In literature there are many successful ACO applications to different combinatorial optimization problems. Dorigo, Caro and Gambardella (1999) classify these applications into two groups: those applied to static combinatorial optimization problems and those applied to dynamic ones.

In static problems the definition and the initial characteristics of the problem are given once and they do not change over time. Classic travelling salesman problem is a good example. The location of cities and their relative distances are given in the problem definition and they do not change during the solution process. Quadratic assignment problem (QAP), Job-shop scheduling problem (JSP), Vehicle routing problem (VRP), Shortest common supersequence problem (SCS), Graph coloring problem (GCP), Sequential ordering problem (SOP), are also some examples of the static problems. In the dynamic case, the problem is defined as a function of some quantities and the values of these quantities depend on the dynamics of the system. As the system changes overtime the algorithm must also adapt itself to this changing environment. Some of the examples of this type of problem are network routing, connection-oriented network routing, connection-less network routing.

Once the mapping of problem (allows incremental construction of a solution), the structure of neighbourhood and the stochastic state transition rule is defined, then the ACO meta-heuristic can be applied directly to a static combinatorial optimization problem. Many early applications of the ACO algorithms in the literature are inspired by Ant System (AS) and most of these applications are the relatively direct application of AS to the problem.

Applications of ACO to the dynamic combinatorial optimization problems and the related research on these applications are focused on the communications network. Because the characteristics communications network match with the properties of ACO meta-heuristic.

Chapter 4

Proposed Approach: Ant Colony Optimization

4.1 Overview of the Proposed Approach

In this chapter we present a new heuristic, an Ant Colony Optimization (ACO) meta-heuristic, and its variants to solve the U-type assembly line balancing problem (UALBP). The UALBP is defined in Chapter 1 and the literature survey on the U-lines and the ACO meta-heuristic is given in Chapter 2. Detailed information about the ACO meta-heuristic can be found in Chapter 3.

Section 4.1 covers the fundamentals of ant type algorithms that do not change from one method to another. Section 4.2 gives detailed information about proposed methods. Especially the task selection rules and the pheromone trail update mechanisms will be explained in details.

4.1.1 Motivation

As stated in Chapter 2 there are some exact algorithms for the U-line problem even though these algorithms can handle only limited size problems. Both UALBP and SALBP are NP-hard problem (Baybars, 1986; Miltenburg and

Sparling, 1995; Miltenburg, 1998; Sparling and Miltenburg, 1998). Obviously large size realistic problems require heuristic approaches.

Even though many heuristics have been developed for SALBP (Erel and Sarin, 1998), for the single model UALBP there are only three heuristics. These are: RPWT based heuristic (Miltenburg and Wijngaard, 1994), branch and bound based heuristic (Scholl and Klein, 1999) and SA based heuristic (Erel, Sabuncuoglu and Aksu, 2001).

In this research, a new heuristic, Ant Colony Optimization (ACO) meta-heuristic, and its variants are proposed for UALBP. There are two ant approaches proposed for SALBP (Bautista and Pereira, 2002; McMullen and Tarasewich, 2003). However this study is the first application of ACO meta-heuristic to U-shaped production lines. The work by McMullen and Tarasewich (2003) considers only six problems (ranging in size from 21 to 74 task) and their objective function is different from our objective function. Bautista and Pereira (2002) consider the same objective function, minimizing the number of stations given a fixed cycle time, but this model is proposed for only SALBP.

Since the first ant algorithm developed by Dorigo and colleagues (1991), several variants of the AS have been proposed in the literature. In general, ACO is an umbrella term for a number of similar metaheuristics: Ant System (AS), Ant System with Elite Strategy (AS_{elite}), Ant System with Ranking (AS_{rank}), Ant Colony System (ACS), *MAX-MIN* Ant System (MMAS) are some of these meta-heuristics. Details can be found in Table 2.1 and Table 2.2 (see also Chapter 3).

4.1.2 Fundamentals

Although there are some aspects that makes these variants of the ACO meta-heuristics differ from each other, the fundamentals of ant type algorithms do not change from one method to another. These are as follows:

1. Problem Representation (UALBP must be represented as a graph that can be searched by many simple agents);
2. Autocatalytic Feedback Process (it is necessary to find an efficient representation for artificial trail values);
3. Solution Generator (heuristic also called as "*greedy force"*. It is very important to define a proper heuristic function for UALBP);
4. Constraint Satisfaction (a proper tabu list that will satisfy the constraints of the problem consideration).

Dorigo, Maniezzo, and Colorni (1991, 1996), state that the most difficult of these is to find an appropriate graph representation for the problem and the greedy force. These issues are discussed next.

4.1.2.1 The graph representation of the problem

For many real life problems, it is possible to describe the structure of the problem using a graph. The key feature of any ant type algorithm is the definition of the links between the elements of a solution (Montgomery and Randall,2002). For example, for the Travelling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP), the key feature is the links between cities/customers respectively. For the Quadratic Assignment Problem (QAP), the links represent couplings of facilities and locations.

UALPB can be defined as a graph, where the nodes represent tasks, the arcs represent precedence relations between tasks and the operation times are given as node weights. The *(i,j)* arc between nodes *i* and *j* represents the precedence relations between task *i* and task *j*. This precedence graph forms a basis for construction of solutions and generation of tabu lists.

4.1.2.2 The autocatalytic process

For the proposed algorithms, the collective behavior of the ants emerges as a form of autocatalytic process (the *autocatalytic process* is a process that reinforces itself) where more ants follow a trail, the trail becomes more attractive for being followed. Here, the amount of pheromone trail represents the attractiveness of assignment of a task to a specific station. It is a global piece of information. Thus, if a task is relatively assigned to a specific station many times, then it is more attractive to assign the same task to the same station next time. The process is thus characterized by a positive feedback loop, in which the probability of assigning a task to a specific station increases with the number of ants that previously assigned the same task to that station. If no control mechanism exists this situation misleads the ants and may cause a very rapid convergence. This is called *stagnation* (Dorigo, Maniezzo and Colorni 1991,1996). For detailed information see Chapter 2.

4.1.2.3 The greedy force

The greedy force is basically defined as the available heuristic value that gives a prior knowledge on attractiveness of a solution component. Thus a solution component with a high heuristic value is more desirable. In fact, the heuristic value represents a priori information about the problem instance definition or run-time information provided by a source different from the ants (Dorigo and Stützle, 2000). It is a local piece information. An ant algorithm only using the heuristic information in the task selection process (no contribution of trail value) is a stochastic greedy algorithm with multiple starting points.

We took inspiration from Miltenburg and Wijngaard's (1994) task priority function, $p(k)$ (this function is also called *U-line maximum ranked positional weight*).

Following Miltenburg and Wijngaard's (1994) definition, let U_k^p is given as the set of tasks which must precede task *k*, and U_k^s is given as the set of tasks which must succeed task *k*. Then at any instant, the set of assignable tasks given as $V = \{k \mid all\ i \in U_k^p\ or\ all\ j \in U_k^s$ have already been assigned$\}$.

A forward/backward priority of a task is defined as a priority function, $p^f(k)$ / $p^b(k)$, and is called *forward/backward U-line ranked positional weight*. This priority function constitutes our heuristic value, and is defined for each task *k* as:

$$p^f(k) = \left\{ t(k) + \sum_{i \in U_k^s} t(i) \right\} \qquad \text{for } k = 1,..,N \qquad (4.1)$$

$$p^b(k) = \left\{ t(k) + \sum_{j \in U_k^p} t(j) \right\} \qquad \text{for } k = 1,..,N \qquad (4.2)$$

Thus, the priority of each task is either the time required to complete both that task and all the tasks that must succeed or precede it. In this way, for tasks whose successors/precedessors require a long time to complete are strongly forward/backward assignable and it is better to assign them as soon as possible.

In order to illustrate the "forward" and "backward" positional weights, consider the well-known Jackson problem, with 11 tasks. Task numbers are written in nodes and duration of operations is written as weight of nodes.

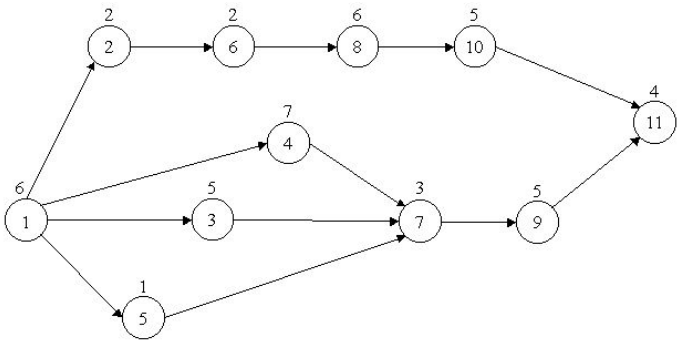


Figure 4.1: Jackson problem with 11 tasks

For task 1, the forward positional weight is the duration of task 1 and sum of task times of all the tasks that succeed it.

$P^f(1) = t(1)+ t(2)+ t(5)+ t(3)+ t(4)+ t(6)+ t(7)+ t(8)+ t(9)+ t(10)+ t(11) = 46$

For task 1, the backward positional weight is the duration of task 1 and sum of task times of all the tasks that precede it.

$P^b(1) = t(1) = 6$

For the Jackson problem, forward and backward positional weights are given in Table 4.1.

Table 4.1: Forward and backward positional weights for Jackson problem.

Task Number	Forward positional weight	Backward positional weight
1	46	6
2	19	8
3	17	11
4	19	13
5	13	7
6	17	10
7	12	22
8	15	16
9	9	27
10	9	21
11	4	46

We will explain allocation of tasks to the stations in Section 4.1.3.

4.1.2.4 The constraint satisfaction

Each ant has its own tabu list that stores the state of each task. During a tour, assigned tasks are marked as tabu and can not be assigned once more. The tabu list[1] is used in order to determine feasible and assignable tasks to calculate the selection probabilities of each.

[1] The term ‘tabu list’ is used to indicate a simple memory that contains the set of already assigned tasks, and has no relation with tabu search.

4.1.3 Generation of a solution

Initially, very small amount of pheromone deposition for each task is given as an initial input to the system. Based on given constraints, precedence relations and limiting cycle time, the possible feasible solutions are constructed and a search list is created. This constitutes the initial solution phase of the solution system. Then, a number of ants move through the neighborhood of the solution in order to find the alternative solutions, hopefully an optimal solution. As they travel from one node to another feasible one, the memory is updated instantly or later depending on the quality of the solution. This iterative process continues until the best solution is obtained or the respecified termination condition (either on the number of cycles or on the computation time of the problem) is satisfied. For our algorithms, the termination criterion is a limit on the number of tours (cycles).

For each ant, completion of one tour (or one cycle) means that all nodes are visited and all tasks are allocated. During a tour for a graph with n nodes (i.e., a problem with n tasks), m ants find m solutions, each of them visiting n nodes (i.e., each of them allocating n tasks). Briefly, during a tour, m ants find m solutions each of them allocating n tasks, and a tour includes n iterations for each ant (During an iteration each ant allocates one task). Generalized flow chart of proposed methods is given in Figure 4.2. Next, we give a numerical example to better explain the main features of the ant algorithms.

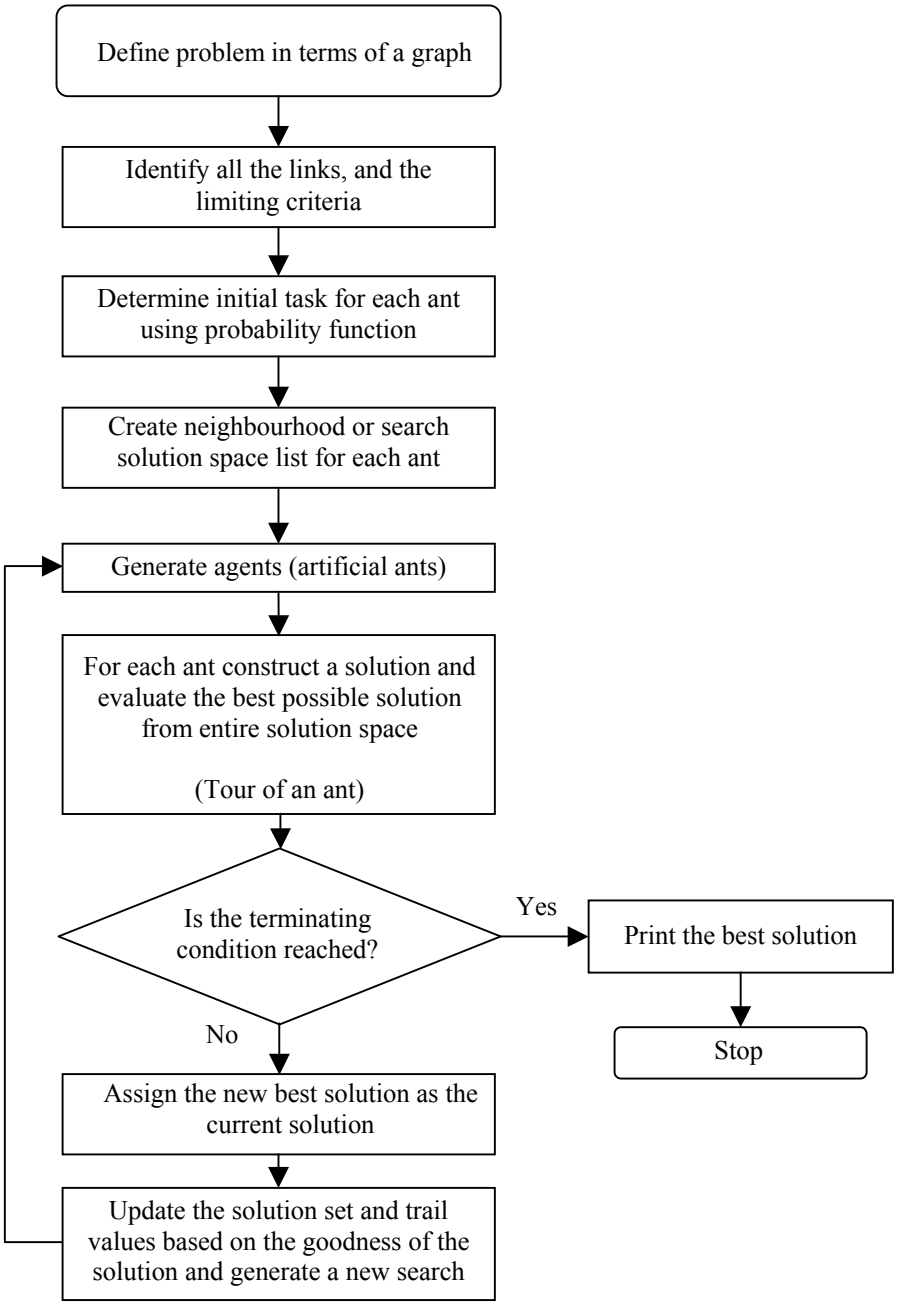


Figure 4.2: A flowchart of the proposed algorithms. This flowchart is adapted from Krishnaiyer and Cheraghi's (2002) generalized flow chart.

Numerical Example:

For illustrative purposes, consider the Jackson problem with a cycle time of 10. The construction of a solution during one tour for a single ant is given below. General form of the probability function, probability of assignment of task *i* to station *j,* is defined as follows:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left[T_{ij}(t)\right]^{\alpha}\cdot\left[\eta_i\right]^{\beta}}{\sum_{k\in allowed_k}\left[T_{kj}(t)\right]^{\alpha}\cdot\left[\eta_k\right]^{\beta}} & if\ i\in allowed_k\\ 0 & otherwise\end{cases} \tag{4.3}$$

Here T_{ij} represents the trail value of assigning task *i* to station *j* and η_i represents the heuristic value called *visibility*. If a task is forward assignable, than its heuristic value is the forward positional weight of that task and if a task is backward assignable, then heuristic value is the backward positional weight of that task. Parameters α and β allow a user to control the relative importance of trail versus visibility. Time counter *t* represents the iteration number. So, the transition probability is a tradeoff between the visibility and the trail intensity. For simplicity, T_{ij} , α and β are taken as 1.

In tabu list; 0's represent unavailable and unassigned tasks, -1's represent assigned tasks, 1's represent forward assignable tasks, and 2's represent backward assignable tasks.

<u>*Iteration 1*</u>*:*

Station number: 1, Station time: 0
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	1	0	0	0	0	0	0	0	0	0	2

Forward available and feasible tasks: 1
Backward available and feasible tasks: 11

$$P_{1,1}(1)=\left\{\frac{\left[T_{1,1}(1)\right]^{1}\cdot\left[\eta_1\right]^{1}}{\left[T_{1,1}(1)\right]^{1}\cdot\left[\eta_1\right]^{1}+\left[T_{11,1}(1)\right]^{1}\cdot\left[\eta_{11}\right]^{1}}\right\}=\frac{1\times 46}{1\times 46+1\times 46}=0.5\,;$$

$$P_{11,1}(1)=\left\{\frac{[T_{11,1}(1)]^{l}\cdot[\eta_{11}]^{l}}{[T_{1,1}(1)]^{l}\cdot[\eta_{1}]^{l}+[T_{11,1}(1)]^{l}\cdot[\eta_{11}]^{l}}\right\}=\frac{1\times 46}{1\times 46+1\times 46}=0.5;$$

Random number, *RN(0,1)* = 0.040; task 1 is forward assigned[2]. New station time is 6.

Iteration 2:

Station number: 1, Station time: 6
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	1	1	1	1	0	0	0	0	0	2

Forward available and feasible tasks: 2, 5
Backward available and feasible tasks: 11

Here task 3 and 4 are also available for forward assignment but their task times are greater than the remaining idle time for station 1. Thus, they are not feasible tasks for iteration 2.

$P_{2,1}(2)=0.244$, $P_{5,1}(2)=0.166$, $P_{11,1}(2)=0.590$

Random number, *RN(0,1)* = 0.494; task 11 is backward assigned[2]. New station time is 10.

Iteration 3:

No available task exists for station 1, thus we open a new station.

Station number: 2, Station time: 0
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	1	1	1	1	0	0	0	2	2	-1

Forward available and feasible tasks: 2, 3, 4, 5
Backward available and feasible tasks: 9, 10

$P_{2,2}(3)=0.164$, $P_{3,2}(3)=0.147$, $P_{4,2}(3)=0.164$, $P_{5,2}(3)=0.112$,

$P_{9,2}(3)=0.232$, $P_{10,2}(3)=0.181$

[2] Task selection: (i) Choose a random number, *RN(O,1).* (ii) Start from the smallest task number. In a non-decreasing order of task numbers, add together the probability $P_{ij}(t)$ of all feasible and assignable tasks (one at a time). Stop immediately when the sum is greater than or equal to *RN(O,1).* The last individual added is the selected task for the assignment.

Random number, *RN(0,1)* = 0.637; task 9 is backward assigned. New station time is 5.

Iteration 4:

Station number: 2, Station time: 5
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	1	1	1	1	0	2	0	-1	2	-1

Forward available and feasible tasks: 2, 3, 5
Backward available and feasible tasks: 7, 10

$P_{2,2}(4) = 0.207$, $P_{3,2}(4) = 0.185$, $P_{5,2}(4) = 0.141$, $P_{7,2}(4) = 0.239$ $P_{10,2}(4) = 0.228$

Random number, *RN(0,1)* = 0.339; task 3 is forward assigned. New station time is 10.

Iteration 5:

No available task exists for station 2, thus we open a new station.

Station number: 3, Station time: 0
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	1	-1	1	1	0	2	0	-1	2	-1

Forward available and feasible tasks: 2, 4, 5
Backward available and feasible tasks: 7, 10

$P_{2,3}(5) = 0.202$, $P_{4,3}(5) = 0.202$, $P_{5,3}(5) = 0.138$, $P_{7,3}(5) = 0.234$, $P_{10,3}(5) = 0.223$

Random number, *RN(0,1)* = 0.172; task 2 is forward assigned. New station time is 2.

Iteration 6:

Station number: 3, Station time: 2
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	1	1	1	2	0	-1	2	-1

Forward available and feasible tasks: 4, 5, 6
Backward available and feasible tasks: 7, 10

$P_{4,3}(6) = 0.207$, $P_{5,3}(6) = 0.141$, $P_{6,3}(6) = 0.185$, $P_{7,3}(6) = 0.239$, $P_{10,3}(6) = 0.228$

Random number, *RN(0,1)* = 0.256; task 5 is forward assigned. New station time is 3.

Iteration 7:

Station number: 3, Station time: 3
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	1	-1	1	2	0	-1	2	-1

Forward available and feasible tasks: 4, 6
Backward available and feasible tasks: 7, 10

$P_{4,3}(7) = 0.241,\ P_{6,3}(7) = 0.215, P_{7,3}(7) = 0.278, P_{10,3}(7) = 0.266$

Random number, *RN(0,1)* = 0.891; task 10 is backward assigned. New station time is 8.

Iteration 8:

Station number: 3, Station time: 8
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	1	-1	1	2	2	-1	-1	-1

Forward available and feasible tasks: 6
Backward available and feasible tasks: -

$P_{6,3}(8) = 1$

Random number, *RN(0,1)* = 0.595; task 6 is forward assigned. New station time is 10.

Iteration 9:

No available task exists for station 3, thus we open a new station.

Station number: 4, Station time: 0
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	1	-1	-1	2	2	-1	-1	-1

Forward available and feasible tasks: 4
Backward available and feasible tasks: 7, 8

$P_{4,4}(9) = 0.333,\ P_{7,4}(9) = 0.386, P_{8,4}(9) = 0.281$

Random number, *RN(0,1)* = 0.194; task 4 is forward assigned. New station time is 7.

Iteration 10:

Station number: 4, Station time: 7
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	-1	-1	-1	2	2	-1	-1	-1

Forward available and feasible tasks: -
Backward available and feasible tasks: 7

$P_{7,4}(10) = 1$

Random number, *RN(0,1)* = 0.076; task 7 is backward assigned. New station time is 10.

Iteration 11:

No available task exists for station 4, thus we open a new station.

Station number: 5, Station time: 0
Tabu list:

Tasks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Tabu values	-1	-1	-1	-1	-1	-1	-1	2	-1	-1	-1

Forward available and feasible tasks: -
Backward available and feasible tasks: 8

$P_{8,5}(11) = 1$

Random number, *RN(0,1)* = 0.582; task 8 is backward assigned. New station time is 6. In Figure 4.3 the task allocation is given.

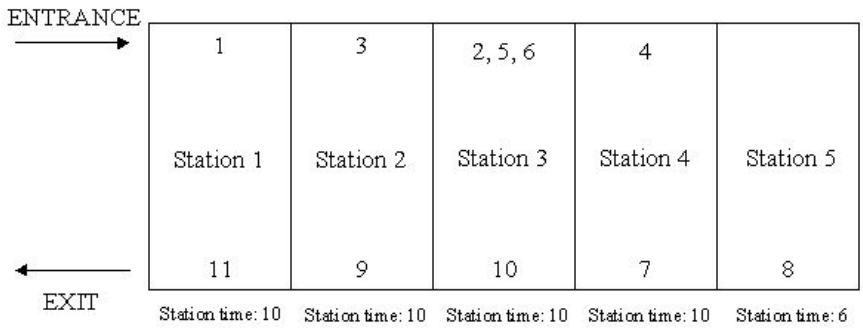


Figure 4.3: The task allocation for Jackson problem, *c* = 10.

4.2 Proposed Methods

This section provides more detailed information on the proposed methods for UALBP. Especially the task selection rules and the pheromone trail update mechanisms will be explained in detail. We consider these methods in three groups: (i) directly applied methods, (ii) modified methods, and (iii) methods in which ACO approach is augmented with a metaheuristic. First group includes methods, AS, AS_{elite}, AS_{rank}, and ACS that is directly applied to UALBP. No modification is done in the structure of the algorithms. Second group includes the new methods that the structure of the algorithms is modified. Also, performance of the algorithms in this group is much better than that in the first group. Last group includes two specialized methods that the ACO approach is augmented with simulated annealing (SA) and beam search (BS).

4.2.1 Ant System (AS)

AS is the first example of an ACO heuristic in the literature and its importance resides in being the prototype of many ant algorithms. Therefore, we consider the AS as our starting point. Indeed, AS is a set of three algorithms called *ant-cycle*, *ant-density*, and *ant-quantity* (Dorigo, Maniezzo, Colorni, 1996). These three versions differ in the way that the trail is updated.

While in *ant-density* and *ant-quantity* the ants update the trail directly just after they allocate a task to a station, in *ant-cycle* the trail update is done only once, after all ants finished a tour and construct a full solution (when all tasks are allocated to stations). The amount of pheromone deposited by each ant is set to be a function of its solution quality. For all methods the solution quality is represented by the number of stations. Ant-cycle is reported (Dorigo, Maniezzo, Colorni, 1996) to perform better than the other two variants and these two

algorithms are no longer studied. Thus, we only consider *ant-cycle*. Generally *ant-cycle* is known as AS.

In AS each ant starts a tour with an empty station. After the initialization phase of the system, depending on each ant's memory, first available tasks are determined and then the tabu list is used to determine the feasible tasks. If there are available tasks but none of them are feasible, then a new station is opened. Starting from first task, an ant iteratively moves from task to task and allocates them to the stations. For iteration *t,* ant *k* chooses a feasible task *i* to assigns to station *j* with a probability given by:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left[T_{ij}(t)\right]^{\alpha}\cdot\left[\eta_i\right]^{\beta}}{\sum_{k\in allowed_k}\left(\left[T_{kj}(t)\right]^{\alpha}\cdot\left[\eta_k\right]^{\beta}\right)} & if\ i\in allowed_k \\ 0 & otherwise\end{cases} \tag{4.3}$$

where allowed_k is the set of available and feasible tasks for ant k. This state transition rule is called as *random-proportional rule.* Parameter η_i is the priory available heuristic information and in Equation 4.3 it is defined as the positional weight of task *i* (see section 4.1.2.2 and section 4.1.2.3). Parameters α and β determine the relative influence of the pheromone trail and the heuristic information. These parameters affect the behavior of the algorithm. If $\alpha = 0$, the selection probabilities are proportional to $\left[\eta_i\right]^{\beta}$ and tasks with high positional weight are more likely to be selected. In this case, AS corresponds to a stochastic greedy algorithm with multiple starts. If $\beta = 0$, only pheromone information affects the selection probabilities, and if no control mechanism exists, this situation misleads the ants. This may cause a very rapid convergence, leading to a *stagnation* situation (Dorigo, Maniezzo, Colorni, 1996). In this situation, all the ants follow the same path and construct the same solutions, which are strongly suboptimal (see Section 4.1.2.2 and Section 4.1.2.3). Thus, there is a trade-off between the trail intensity and the heuristic value.

After each ant has completed a tour, the solution construction ends. Next, the pheromone trails are updated by first lowering the trail values by a constant factor (pheromone evaporation) and then allowing each ant to deposit an amount of pheromone on the related trail value. For example, at time *t* if ant *k* allocates task 3 to station 5, then the related trail value for this solution component is $\Delta T_{3,5}^{k}(t)$. Trail values are updated as follows:

$$T_{ij}(t) = (1-\rho)\cdot T_{ij}(t-1) + \sum_{k=1}^{m} \Delta T_{ij}^{k}(t) \qquad \forall (i,j) \tag{4.4}$$

where *m* is the number of ants and ρ is the pheromone trail evaporation rate ($0 < \rho < 1$). Rate ρ enables the algorithm to forget the previously made bad selections. Thus, unlimited accumulation of the pheromone trails is avoided. The amount of pheromone trail, $T_{ij}(t)$ represents the learned desirability of allocating task *i* to station *j*. The trail values that are not updated will decrease exponentially with the number of tours. $\Delta T_{i,j}^{k}(t)$ is the amount of pheromone deposit by ant *k* if task *i* is allocated to station *j* at tour *t* and it is defined as:

$$\Delta T_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f^{k}(t)} & \textit{if task i is allocated to station j by ant k} \\ 0 & \textit{otherwise} \end{cases} \tag{4.5}$$

where $f^{k}(t)$ is the station number found by ant *k* and *Q* is a constant. $\Delta T_{i,j}^{k}(t)$ depends on how well the ant has performed. Equation 4.5 satisfies more pheromone to be deposit for better solutions (allocations with less number of stations). The tasks which are allocated to some specific stations by more ants and which are the part of a solution with less number of stations will receive more pheromone. Therefore, these tasks are more likely to be allocated to same stations in future tours of the algorithm. This choice helps to direct the search towards the better solutions.

During tours, changing pheromone trail information reflects the experience acquired by ants.

The initial amount of pheromone $T_{i,j}(0)$ is set to a very small positive constant value, τ_0 and the total number of ants m, is set equal to the total number of tasks, n. Values of α, β, ρ and Q are found after fine-tuning of these parameters. Further information will be given in Chapter 5. Flowchart of the AS algorithm is given in Figure 4.4.

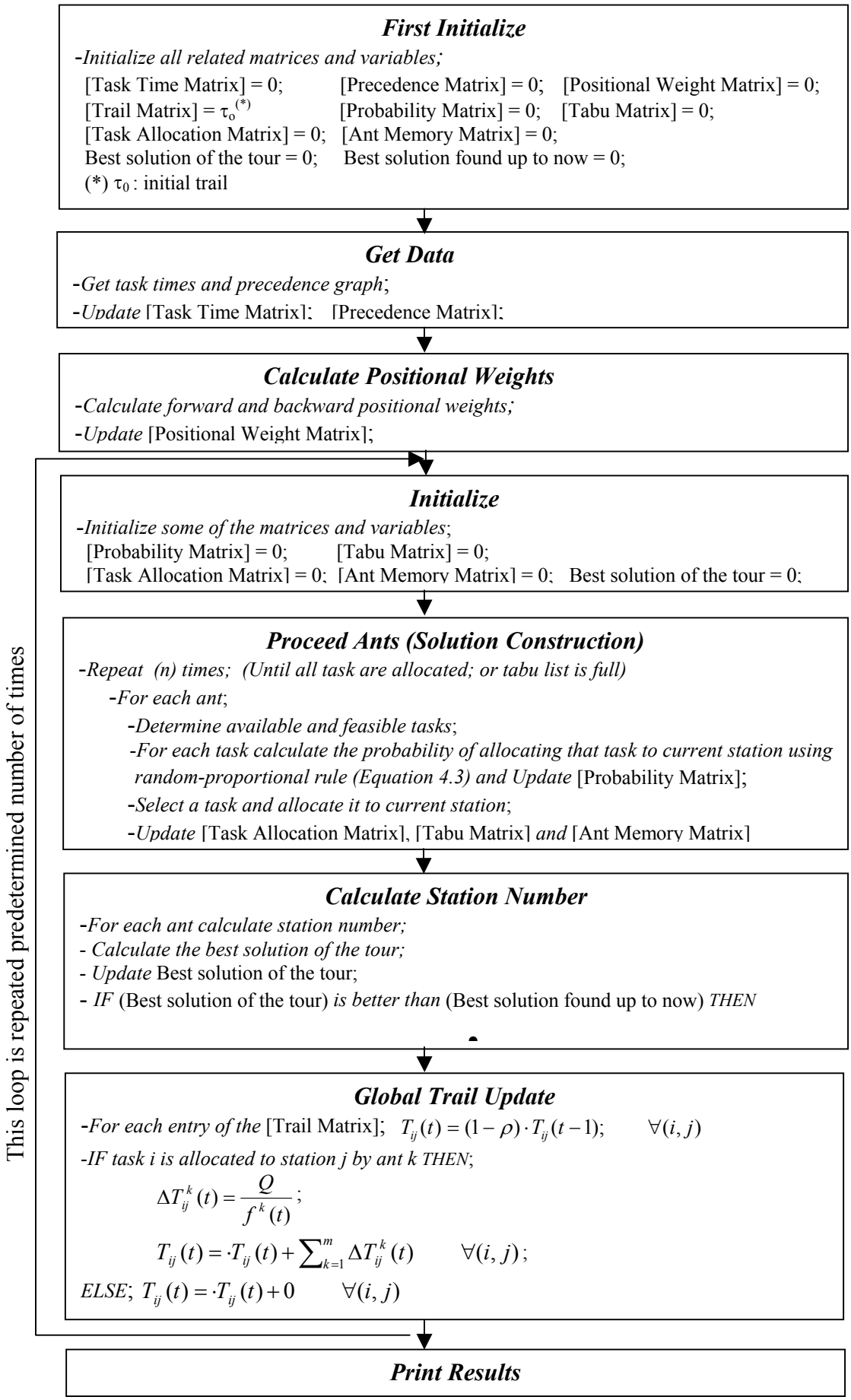


Figure 4.4: Flowchart of Ant System

4.2.2 Ant System with Elitist Strategy (AS_{elite})

The idea of the elitist strategy is giving an extra emphasis to the best solution found after every tour (Bullnheimer, Hartl and Strauss, 1997). In the AS the trail values are updated based on the solution quality of all ants. The level of contribution depends on the solution quality. A shortcoming of this procedure is stated by Bullnheimer, Hartl and Strauss (1997) as follows: after a period if the overall solution quality rises and the difference between the solutions decrease, the effect of emphasizing better solutions will diminish. Thus the difference in trail values and also the selection probabilities decrease. Therefore, the exploitation of the solution space will not be as high as desired. One possible alternative to solve this problem is *elitist strategy*.

After the trail values are updated, the best solution of the tour is treated as if a certain number of elite ants had found that solution. Because some parts of this solution may belong to the optimal solution. Thus the idea is to guide the search in succeeding tours. The trail updating mechanism is as follows:

$$T_{ij}(t) = (1-\rho)\cdot T_{ij}(t-1) + \sum_{k=1}^{m} \Delta T_{ij}^{k}(t) + \Delta T_{ij}^{(*)}(t) \qquad \forall (i,j) \quad (4.6)$$

where
$$\Delta T_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f^{k}(t)} & \text{if task } i \text{ is allocated to station } j \text{ by ant } k \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

$$\Delta T_{ij}^{(*)}(t) = \begin{cases} \sigma \dfrac{Q}{f^{(*)}(t)} & \text{if } (i,j) \text{ is part of the best solution} \\ 0 & \text{otherwise} \end{cases} \quad (4.7)$$

$\Delta T_{ij}^{(*)}(t)$ is the increase of trail level for allocation of task i to station j caused by the elite ants. σ is the number of elite ants and $f^{(*)}(t)$ is the station number of the best solution found. Bullnheimer, Hartl and Strauss's (1997) experiment results for the TSP are very good when $\sigma = \alpha = \beta$. Nevertheless AS_{elite} does not give good results for UALBP. After a period, almost all ants find the best solution of the tour. Although the allocation of each ant is different from

each other, the station number of these different allocations is the same. Thus, the elitist strategy is applied for all of them and the extra emphasis is given for all of their solutions. This does not help with the distinction of the actual best solution.

Actually this is related with the topology of the cost function. Hertz and Widmer (2003) state that the topology of the cost function should not be too flat for the heuristics to get the optimal point. The cost function can be considered as an altitude with mountains, valleys and plateaus. If the cost function is too flat, it is difficult for the search mechanism to escape from the large plateaus to fall into the valleys. To tackle this problem, Hertz and Widmer (2003) suggest adding a component to the cost function to discriminate the solutions with the same original cost function value.

4.2.3 Ant System with Ranking (AS_{rank})

The concept of ranking is similar to the elitist strategy. After all m ants generate a tour, these ants are sorted according to their solution quality. ($f_1 \leq f_2 \leq \ldots \leq f_m$). The contribution of an ant to the trail level update is weighted according to the rank of that ant (Bullnheimer, Hartl and Strauss,1997). Only w best ants are considered and this prevents the over-emphasis of the pheromone trails when there are too many ants.

Based on computational results, Bullnheimer, Hartl and Strauss (1997) state that the exploitation as well as the exploration is considerably high and well balanced. Nevertheless AS_{rank} is not proper for UALBP. In UALBP, after a period almost all ants find the best solution of the tour. Although the allocation of each ant is different from each other, the station number of these different allocations is the same. Therefore, when ants are sorted by their solution quality (the number of stations) most of them will have an equal rank and their contribution to the trail

level update will be equally weighted. There will be no distinction among the solutions and this will not help the exploitation of the better solutions.

4.2.4 Ant Colony System (ACS)

Ant Colony System is proposed by Dorigo and Gambardella (1997), Gambardella and Dorigo (1996) to improve the performance of the AS. ACS is based on an earlier algorithm proposed by the same authors, called Ant-Q.

ACS differ from the AS in three main aspects: (i) ACS uses a more greedy action choice rule than that of AS, (ii) the global pheromone trail update rule is only applied for the global-best solution (after a tour), (iii) while the ants construct a solution, a *local pheromone updating rule* is applied (during a tour). In the following, these modifications will be explained in more detail.

Tour Construction. In ACS the state transition rule is defined as follows: an ant allocates a task s to station j by applying the rule given in Equation (4.8)

$$s = \begin{cases} \arg \max_{u \in allowed_u} \left\{ [T_{uj}(t)] \cdot [\eta_u]^{\beta} \right\} & if\ q \le q_0 \quad (\text{exp}loitation) \\ S & otherwise \quad (biased\ \text{exp}loration) \end{cases} \tag{4.8}$$

where q is a uniformly distributed random number in [0,1], q_0 is a parameter ($0 \le q_0 \le 1$), and S is a random variable selected according to the random-proportional rule given in Equation (4.3). The state transition rule is the combination of Equations (4.3, 4.8) and called *pseudo-random-proportional rule.*

This state transition rule favors allocations with a large amount of pheromone. Parameter q_0 controls the relative importance of exploitation versus exploration. If $q \le q_0$ then the best task is chosen and this selection is a kind of greedy behavior which favors the exploitation of the search space. Otherwise, a task is chosen according to Equation (4.3) and that favors the exploration of the search space.

Global Pheromone Trail Update. In ACS only the globally best ant is allowed to deposit pheromone. This ant is the one, which finds the best solution from the beginning. Dorigo and Gambardella (1997) state that the global updating rule together with the use of the pseudo-random-proportional rule makes the search more directed. The global updating is performed after all ants have constructed their solutions (when a tour is completed) according to Equation (4.9):

$$T_{ij}(t) = (1-\rho_1)\cdot T_{ij}(t-1) + \rho_1 \cdot \Delta T_{ij}^{(gb)}(t) \tag{4.9}$$

where

$$\Delta T_{ij}^{(gb)}(t) = \begin{cases} \dfrac{1}{f^{(gb)}(t)} & \text{if } (i,j) \text{ is part of the global best solution} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

$0 \le \rho_1 \le 1$ is pheromone evaporation parameter, and $f^{(gb)}(t)$ is the station number of the globally best solution. As in AS, the global updating favors the better solutions. However, this time only those trail values belonging to the globally best solution receive reinforcement.

Local Pheromone Trail Update. In ACS, additional to the global updating rule, the ants use a local update rule. That is just after they allocate task *i* to station *j* they change the related pheromone level by using Equation (4.11):

$$T_{ij}(t) = (1-\rho_2)\cdot T_{ij}(t-1) + \rho_2 \cdot \tau_0 \tag{4.11}$$

where ρ_2 , $0 \le \rho_2 \le 1$, and τ_o is the initial pheromone level. The effect of the local updating is to make the desirability of tasks change dynamically. The local updating rule makes the already chosen task less desirable for the following ants. In this way, the exploration of unallocated tasks is increased and the ants make a better use of pheromone information.

For UALBP, ACS performed better than AS, AS_{elite} and yield better results. ACS is able to find the optimal solution for some small and medium size instances that AS and AS_{elite} can not find the optimal solution. Nevertheless,

ACS's performance is poor for some large size problems. ACS finds such allocations having one station more than the optimal allocation. Details are given in Chapter 5. When we study the behavior of ACS, our investigations make us focus on pheromone trail accumulation mechanism and the structure of the trail matrix. When the problem size increases, it gets difficult to find the proper task allocation that gives the optimal solution. While searching for the optimal allocation it is very hard to find the exact place of a task (that may lead to an optimal allocation) by using only a random selection mechanism. Therefore, to increase the probability of allocating a task to a correct place, the related pheromone level must be comparatively high. Also structure of the trail matrix must be suitable in order to allow this pheromone accumulation. Consider the Jackson problem given in Figure 4.5. The optimal task allocation is given in Figure 4.5.a and Figure 4.5.b with different representations.

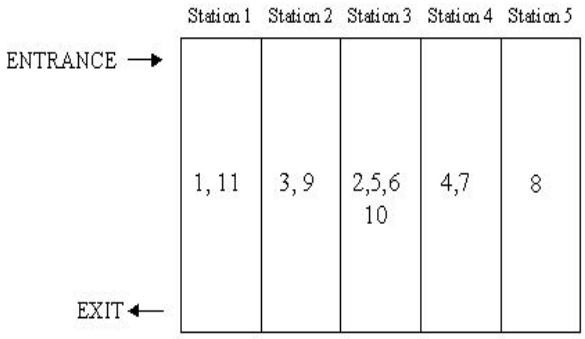


Figure 4.5.a: Optimal task allocation for the Jackson problem, c=10. Location of tasks is not given.

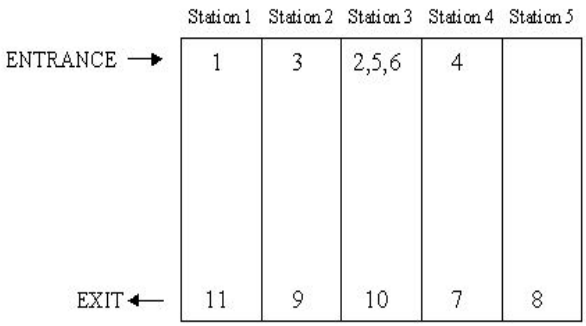
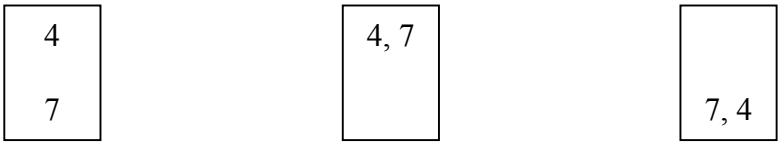


Figure 4.5.b: Optimal task allocation for the Jackson problem, c=10. Location of tasks is given.

When the location information is not given, it is not clear where the tasks are located in a station and how they are allocated (forward or backward). Sometimes knowing only to which station a task is assigned will not be enough. Consider station 4 in Figure 4.5.a. We only know that tasks 4 and 7 are allocated to station 4. However, the location information is unknown. With this information, it is possible to define three different feasible allocation alternatives:

Alternative 1: Station 4; *Alternative 2*: Station 4; *Alternative 3*: Station 4;



However, only Alternative 1 is the part of the given optimal allocation.

In order to emphasize the location information, the structure of the trail representation is modified to consider the location values. With this new modification, for a given tour number *t*, $T_{ij(upper)}(t)$ represents the desirability of allocating task *i* to the upper part of station *j*; and $T_{ij(lower)}(t)$ represents the desirability of allocating task *i* to the lower part of station *j*. For station 4, only trail values $T_{4,4(upper)}(t)$ and $T_{7,4(lower)}(t)$ will receive reinforcement. However in the previous version of ACS, when $T_{4,4}(t)$ value is reinforced, at the same time $T_{4,4(upper)}(t)$ and $T_{4,4(lower)}(t)$ receive reinforcement (Note that, $T_{ij}(t) = T_{ij(upper)}(t) + T_{ij(upper)}(t)$) but only the $T_{4,4(lower)}(t)$ characterizes the real situation.

With this new pheromone structure, proposed algorithm performs better than the previous version. It is able to find the optimal allocations that the previous version could not find. For the problems that optimal solution is found, this modified version finds the optimal solution faster than the previous version.

Flowchart of the ACS algorithm is given in Figure 4.6 (The pheromone structure modification is not included in the figure).

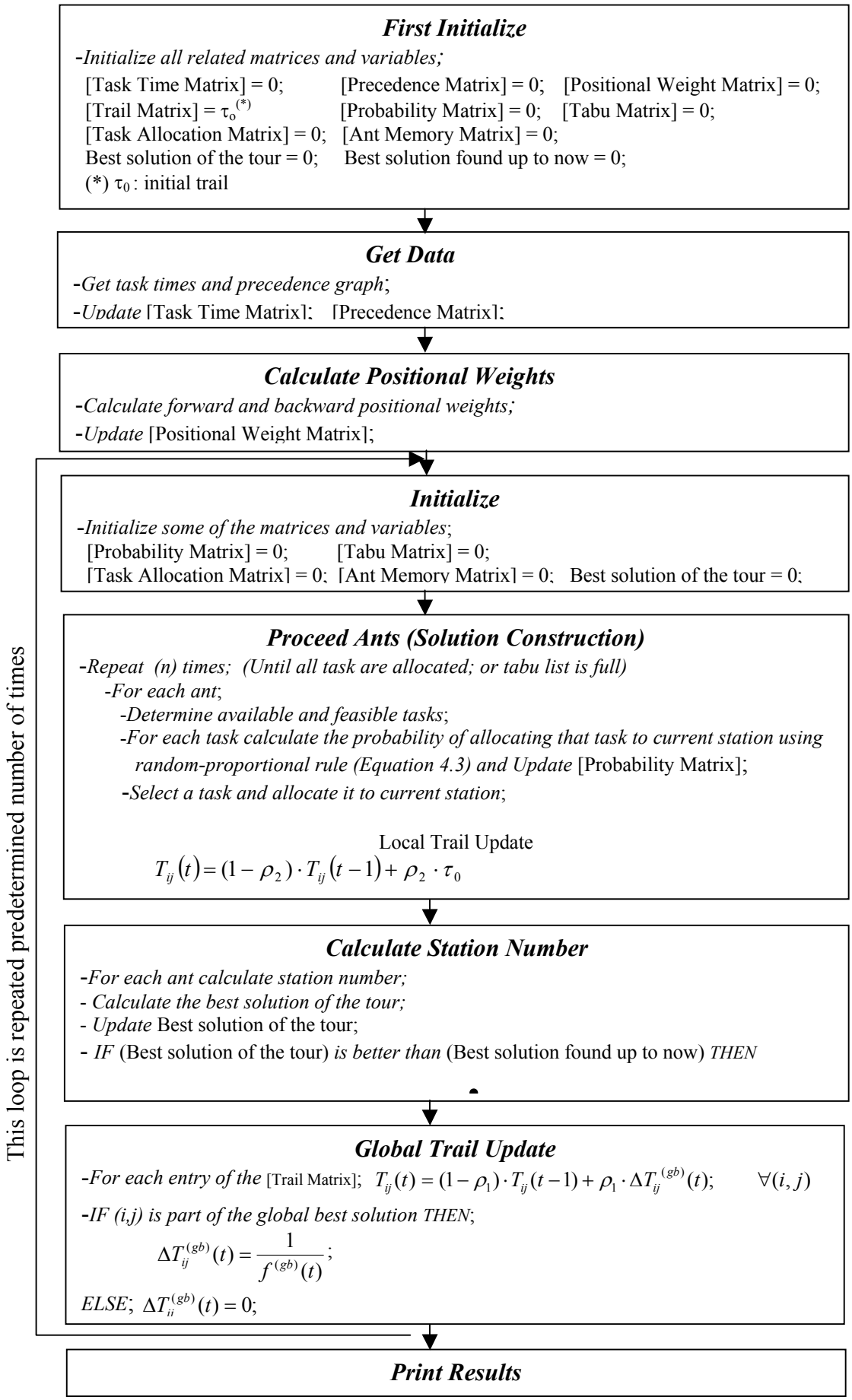


Figure 4.6: Flowchart of Ant Colony System

4.2.5 Modified Ant Colony System (ACS) with Random Search

With the new pheromone structure, ACS algorithm performs better than the previous versions. It is able to find the optimal allocations that the previous version could not find (Details are given in Chapter 6). This new pheromone structure allows the exploitation of some new and important information. This fact leads us to investigate the task allocations and accumulation of the trail values for problems that the optimal solution is not found. We focused on the two smallest size problems (Buxey with cycle time of 36, and Gunther with cycle time of 69) that AS, AS_{elite}, and ACS could not find the optimal solution; none of the modifications and parameter fine-tuning succeed with these problems.

In order to explain our modifications, consider Gunther problem with 35 tasks and cycle time of 69. We run ACS algorithms for 1 replication until 1000 tours have been completed and store the trail matrix data to investigate the accumulation of pheromone (Trail matrix for tour number 10, 100, 500, 1000 is given in Appendix A). After a short period, most of the entries of the trail matrix converge to zero and the remaining entries have a value, which is very close to zero. Somehow the valuable information intended to gain by pheromone accumulation is lost. The difference between the very small values of trail matrix and the values of positional weight matrix is very high. Therefore, the influence of pheromone trail on the selection probabilities is very low. In this situation, most of the information for selection probabilities is coming from the heuristic value. The selection probabilities are mostly proportional to $[\eta_i]^\beta$ and the tasks with high positional weight are more likely to be selected. In this case, algorithm behaves like a stochastic greedy algorithm. This also explains the fact that why small and medium size problems are easily solved with ACS. For most of these problems, the positional weight information is sufficient to search the solution space.

However for harder and larger size instances it is insufficient. Therefore, we need to magnify the effect of trail values and we need to satisfy the pheromone accumulation in an effective way.

Modification Step 1. Our first method is simple and works cooperatively with ACS. We use a *secondary pheromone trail accumulation mechanism* that directly collects the task allocation information. After all ants finished a tour and constructed a full solution (when all tasks are allocated to stations), a secondary trail update mechanism works in parallel with the global trail update. For each globally best ant *k*, if a task *i* is allocated to location *j'*(now each station is divided into two locations) this secondary trail update mechanism updates the related entry of a secondary trail matrix by adding only 1. We call this secondary trail matrix as *T2,* and it is updated as follows:

$$T2_{ij'}(t) = T2_{ij'}(t) + \sum_{k=1}^{m} \Delta T2_{ij'}^{k}\ (t) \qquad \forall (i,j) \tag{4.12}$$

where

$$\Delta T2_{ij'}^{k}(t) = \begin{cases} 1 & \textit{if task i is allocated to location } j' \textit{ by globally best ant } k \\ 0 & \textit{otherwise} \end{cases} \tag{4.13}$$

First we collect the *T2* matrix for a single and very long replication. By letting the algorithm to work for a long replication, we give an opportunity for some critical entries of the *T2* matrix to be emphasized more. In their experiments, Dorigo, Maniezzo, Colorni (1991, 1996) let the algorithm run until 2500 tours have been completed. Similarly, we have done two experiments: (i) the algorithm is allowed to run until 5000 tours have been completed, (ii) the algorithm is allowed to run until 10000 tours have been completed (the *T2* matrix for tour number 5000 and 10000 is given in Appendix B).

Scholl and Klein's (1999) optimal task allocation for Gunther problem is given in Figure 4.7. In Table 4.2, we list the ranking of most possible location alternatives for each task depending on the *T2* matrix.

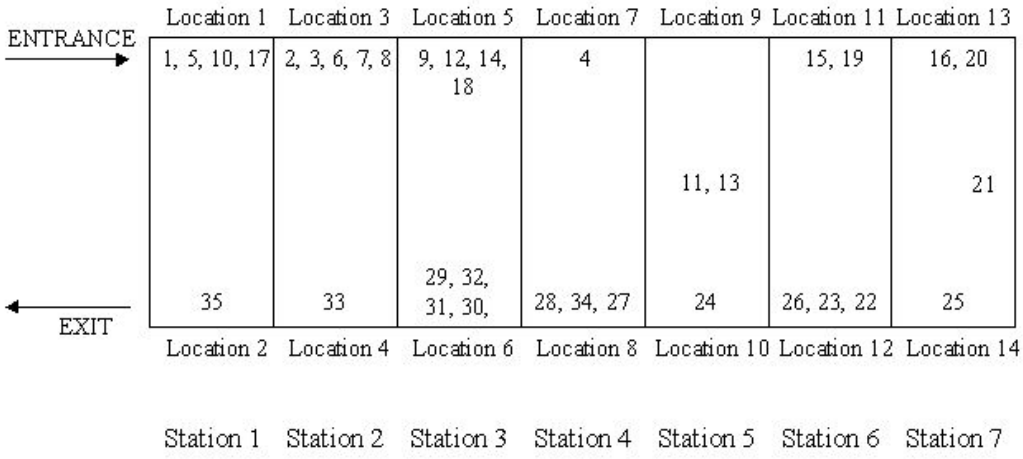


Figure 4.7: Scholl and Klein's (1999) optimal task allocation for Gunther problem

Table 4.2: Ranking of most possible location alternatives for each task depending on *T2* matrix

Alternatives / Task	Experiment with 5000 tours					Experiment with 10000 tours				
	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
1	1	3	5	7	13	1	3	5	7	13
2	1	3	5	7	9	1	3	5	7	9
3	3	5	7	1	9	3	5	7	1	9
4	7	9	5	11	3	7	9	11	5	3
5	3	1	5	7	9	3	1	5	7	9
6	3	7	5	9	1	3	7	5	9	1
7	3	7	5	9	1	3	7	5	9	1
8	7	3	9	5	1	7	3	5	9	1
9	9	11	7	5	13	7	11	9	5	3
10	7	3	9	5	1	3	7	9	5	1
11	14	10	12	8	5	12	14	10	8	7
12	7	3	9	5	1	3	7	9	1	11
13	10	8	14	12	6	12	10	6	14	7
14	9	11	5	7	3	9	11	5	7	3
15	11	13	9	7	5	11	13	9	7	5
16	13	14	11	12	9	13	11	14	12	9
17	1	3	5	7	9	3	1	5	7	9
18	9	11	5	7	3	9	11	5	7	3
19	11	9	13	7	5	11	13	9	7	5
20	13	14	11	12	9	13	14	11	9	10
21	12	14	10	13	11	14	12	10	13	11
22	12	14	10	8	6	12	14	10	8	6
23	10	12	8	6	14	12	10	6	8	14
24	10	8	12	4	6	10	4	8	14	12
25	10	8	6	12	4	10	8	6	12	4
26	8	6	10	12	4	8	6	10	4	12
27	4	8	6	10	12	4	8	6	20	14
28	4	2	6	8	14	4	2	6	8	12
29	2	4	6	8	10	2	4	6	8	10
30	2	8	10	12	6	2	8	10	12	6
31	2	6	8	10	12	2	6	8	10	4
32	2	6	4	8	10	2	6	4	8	10
33	2	4	6	8	10	2	6	4	8	10
34	2	4	6	8	10	2	4	6	8	10
35	2	4	6	8	10	2	4	6	8	10
Total matching	**14**	**10**	**3**	**5**	**2**	**14**	**8**	**5**	**3**	**3**

When we compare our alternatives with Scholl and Klein's (1999) optimal task allocation, the results are very encouraging. In Table 4.2 shaded cells represents alternatives matching with Scholl and Klein's (1999) optimal task allocation. For most of the tasks the first or the second alternative gives a matching.

Modification Step 2.

Observation 1. While collecting data for the *T2* matrix we take a single long run. However, for a particular replication, consecutive tours are not independent form each other. Therefore the observations are neither independent nor identically distributed. There is a bias and we can not statistically trust our data. Thus we modify out method. In order to collect reliable data for the *T2* matrix we make 100 independent replications (runs) of length 100 tours and for each replication (run) different random numbers are used.

Observation 2. Since there is no pheromone evaporation mechanism for the *T2* matrix, algorithm does not forget the previously done bad selections and there is a risk of unlimited accumulation of the trail values. Therefore, during a single replication at time *t*, if there is an improvement and globally best solution is updated then all the entries of the *T2* matrix is cleared and new update is done for only new globally best solution (Because we do not need pheromone accumulation for previous globally best solution any more).

Observation 3. During a single replication some ants (assume *k* ants) can find the same globally best solution. When the *T2* matrix is updated for each ant indeed it is updated *k* times only for the same single solution. This causes over-emphasis of that globally best solution. To handle this problem, during a single run, every solution must be kept in the memory, every new globally best solution must be compared with the previous globally best solutions and the *T2* matrix

must be updated for only the new globally best solutions. Unfortunately it is a very hard job to keep every globally best solution in the memory. (Assume that every tour, *k* ants find a new solution. For a single replication of length *m*, there exist *m x k* solutions. Moreover each solution is an allocation matrix of *n x 2n* dimension; where *n* is the task number and *2n* is the maximum limit allowed)

Solution Labeling Mechanism. We propose a very practical and effective solution labeling mechanism to overcome this memory problem. Each globally best solution is characterized with two labels and the algorithm uses these labels while checking if this solution is previously found or not.

Consider two similar optimal allocation for the Jackson problem with cycle time of 10. Allocation A and Allocation B are given in Figure 4.8.

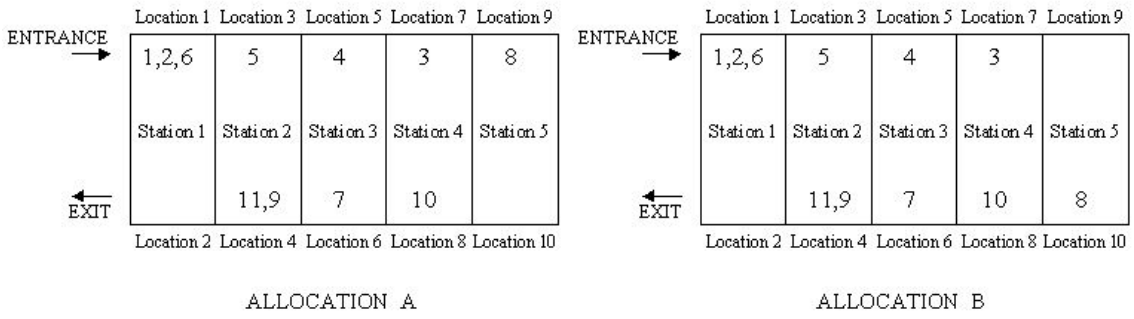


Figure 4.8: Two similar optimal allocation for Jackson problem, *c* =10.

Label 1 is calculated as follows:

$$Label\ 1 = \sum_{j'=1}^{J'} \left(j' \cdot \sum_{i \in I(j')} number\ of\ task(i) \right) \tag{4.14}$$

Label 2 is calculated as follows:

$$Label\ 2 = \sum_{j'=1}^{J'} \left(j' \cdot \prod_{i \in I(j')} number\ of\ task(i) \right) \tag{4.15}$$

where, *J'* is the total number of locations; *I(j')* is the set of tasks allocated to location *j'*.

For *Allocation A*;

Label 1 = 1(1+2+6) + 3(5) + 4(11+9) + 5(4) + 6(7) + 7(3) + 8(10) + 9(8) = 339;

Label 2 = 1(1·2·6) + 3(5) + 4(11·9) + 5(4) + 6(7) + 7(3) + 8(10) + 9(8) = 658.

For *Allocation B*;

Label 1 = 1(1+2+6) + 3(5) + 4(11+9) + 5(4) + 6(7) + 7(3) + 8(10) + 10(8) = 347;

Label 2 = 1(1·2·6) + 3(5) + 4(11·9) + 5(4) + 6(7) + 7(3) + 8(10) + 10(8) = 666.

Observation 4: There can be different allocations that their *Label 1* values are equal.

Proof: Assume that for *Allocation 1* tasks 7 and 5 are allocated to location *n* and for *Allocation 2* tasks 10 and 2 are allocated to location *n.*

For *Allocation 1*, *Label 1* = n(7+5) = 12n; *Allocation 2*, *Label 1* = n(10+2) = 12n

For *Allocation 1*, *Label 2* = n(7·5) = 35n; *Allocation 2*, Label 2 = n(10·2) = 20n

Observation 5: There can be different allocations that their *Label 2* values are equal.

Proof: Assume that for *Allocation 1* tasks 2 and 6 are allocated to location *n* and for *Allocation 2* tasks 3 and 4 are allocated to location *n.*

For *Allocation 1*, *Label 1* = n(2+6) = 8n; *Allocation 2*, *Label 1* = n(3+4) = 7n

For *Allocation 1*, *Label 2* = n(2·6) = 12n; *Allocation 2*, *Label 2* = n(3·4) = 12n

Observation 6: There can not be different allocations that their *Label 1* and *Label 2* values are equal.

Proof: Let for *Allocation 1* tasks x_1 and y_1 are allocated to location *n* and for Allocation *2* tasks x_2 and y_2 are allocated to location *n.*

Assume that *Allocation 1* and *Allocation 2* are different and their *Label 1* and *Label 2* values are equal. By using labeling definition we can write:

(1) $n(x_1 + y_1) = n(x_2 + y_2)$; (Equality of *Label 1* values)

(2) $n(x_1 \cdot y_1) = n(x_2 \cdot y_2)$; (Equality of *Label 2* values)

By using *(2)* we can write: *(3)* $y_1 = \frac{x_2 \cdot y_2}{x_1}$ and *(4)* $y_2 = \frac{x_1 \cdot y_1}{x_2}$;

By using *(1)*; $x_1 + y_1 = x_2 + y_2$

$x_1 - y_2 = x_2 - y_1$

By using *(4)*; $x_1 - \frac{x_1 y_1}{x_2} = x_2 - y_1$ or By using *(3)*; $x_1 - y_2 = x_2 - \frac{x_2 y_2}{x_1}$

$$\frac{x_1 x_2}{x_2} - \frac{x_1 y_1}{x_2} = \frac{x_2^2}{x_2} - \frac{y_1 x_2}{x_2} \qquad \frac{x_1^2}{x_1} - \frac{x_1 y_2}{x_1} = \frac{x_1 x_2}{x_1} - \frac{x_2 y_2}{x_1}$$

$$x_1 x_2 - x_1 y_1 = x_2^2 - y_1 x_2 \qquad x_1^2 - x_1 y_2 = x_1 x_2 - x_2 y_2$$

$$x_1(x_2 - y_1) = x_2(x_2 - y_1) \qquad x_1(x_1 - y_2) = x_2(x_1 - x_2)$$

$x_1 = x_2 \rightarrow y_1 = y_2$ $\qquad$ $x_1 = x_2 \rightarrow y_1 = y_2$

If $x_1 = x_2$ and $y_1 = y_2$ then *Allocation 1* is equal to *Allocation 2* and this contradicts with our assumption. So there can not be and different allocations that their *Label 1* and *Label 2* values are equal.

Random Search with Multiple Starts. The *T2* matrix is collected in an effective and statistically correct way after the modifications given in Observations 1-3. After gathering *T2* matrix the ants are used to make a random search by only using the *T2* matrix. This time algorithm is modified to use only the secondary trail matrix, *T2*. There is no primary trail update mechanism (no

local or global pheromone update) or pheromone evaporation. Indeed, tours are independent from each other. We can call this second stage as *a pure random search with multiple starts*.

We test this method with medium size problems that AS, AS_{elite} and ACS never find the optimal solution. These problems are Buxey with 29 tasks (C=36), Gunther with 35 tasks (C=69), Warnecke with 58 tasks (C=60, C=78, C=82, C=86, C=92, C=97, C=104, C=111), Lutz2 with 89 tasks (C=11, C=12, C=13, C=14, C=17).

This random search method finds the optimal allocation for problems Buxey (C=36), Gunther (C=69), Warnecke (C=60, C=111). It is an important development that for the first time we are able to find the optimal solution for these problems.

However, there is a disadvantage of this method. For large size problems it requires extensive amount of time to collect the *T2* matrix. We test the random search method on large size problems but we can not find the optimum solution. Therefore, we need a more efficient mechanism to search the solution space.

Our observations and investigations on the structure of the *T2* matrix led us to develop the New Ant Colony Optimization method.

4.2.6 A New Ant Colony Optimization (ACO) Method

This new method takes inspiration from the ACS and the secondary trail accumulation mechanism proposed in the previous section. For ACS, after a short period most of the entries of the trail matrix converge to zero and the remaining entries have a value, which is very close to zero. Somehow the valuable information intended to gain by pheromone accumulation is lost. The secondary pheromone trail accumulation mechanism particularly overcomes this situation. However, this method is a two-stage method and it requires extensive amount of

time to collect the *T2* matrix. Furthermore, only random search is not enough for the large size problems. Therefore, this new method is designed to overcome these problems.

Version 1.

Modification of Local Pheromone Update Mechanism. We modify the local pheromone update mechanism of ACS to prevent losing valuable information gained by the pheromone accumulation. The new local pheromone update mechanism is defined as follows:

$$T_{ij}(t) = (\rho_2) \cdot T_{ij}(t-1) + 1 \tag{4.16}$$

where ρ_2 , $0 \leq \rho_2 \leq 1$. Instead of adding a very small value $\rho_2 \cdot \tau_o$ (Equation 4.11), we reinforce the trail matrix by adding up 1 as we did in the previous method while collecting the *T2* matrix. Thus, emphasize of the related trail values will be more and their effect will be magnified by this way. For the pheromone evaporation we use only ρ_2 as a multiplier instead of $(1-\rho_2)$.

Modification of Global Pheromone Update Mechanism. We modify the global pheromone update mechanism of ACS to increase the effect of better solution on trail reinforcement. The new global pheromone update mechanism is defined as follows:

$$T_{ij}(t) = (1-\rho_1) \cdot T_{ij}(t-1) + \Delta T_{ij}^{(gb)}(t) \tag{4.17}$$

where

$$\Delta T_{ij}^{(gb)}(t) = \begin{cases} \dfrac{Optimal}{f^{(gb)}(t)} & if\ (i,j)\ is\ part\ of\ the\ global\ best\ solution \\ 0 & otherwise \end{cases} \tag{4.18}$$

$0 \leq \rho_1 \leq 1$ is pheromone evaporation parameter, *Optimal* is the optimal station number of the considered problem, $f^{(gb)}(t)$ is the station number of the globally best solution and $\dfrac{Optimal}{f^{(gb)}(t)}$ is a *reward function* that satisfies more reinforcement for the solutions which have less station number (a solution with less station

number is more close to the optimal solution and the reward value will be more for this solution).

Modification of Pseudo-Random-Proportional Rule. When we investigate the Scholl and Klein's (1999) optimal task allocations we note that, the idle time of almost all stations is zero, namely it is not possible to allocate an another task to these stations. These stations are called *full-loaded stations*. This is a very interesting and important characteristic of the optimal solutions.

We have to guide the search mechanism to obtain such kind of task selection and allocation, which gives a full-load. Duration of a task is the key point. Thus, the task selection process (state transition rule) must also consider the task times. On the other hand, it is an important decision to consider which tasks first; the tasks with low duration or high duration?

Consider an empty station. At the beginning no task is allocated and the idle time is equal to the cycle time. Assume that first the tasks with low duration are allocated to that station. As the low duration tasks assigned, the remaining idle time decreases and only the tasks with high duration remain unassigned. Later on, there will be only limited idle time left which is not enough for the assignment of the remaining tasks with high duration. On the contrary, if the tasks with high duration are allocated to that station first, later the remaining tasks with low duration would not be a problem. It will be easy to allocate them to this station or to an another available station. Therefore, we modify the task selection mechanism (state transition rule) to emphasize the tasks with high duration. The new random-proportional rule is defined as follows:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{\left[T_{ij}(t)\right]^{\alpha}\cdot\left[\eta_i\right]^{\beta}\cdot\left[t_i\right]^{\beta}}{\sum_{k\in allowed_k}\left(\left[T_{kj}(t)\right]^{\alpha}\cdot\left[\eta_k\right]^{\beta}\cdot\left[t_i\right]^{\beta}\right)} & if\ i\in allowed_k \\ 0 & otherwise\end{cases} \tag{4.19}$$

where $allowed_k$ is the set of available and feasible tasks for ant k.

An ant allocates task *s* to station *j* by applying the rule given by Equation (4.20):

$$s = \begin{cases} \arg\max_{u \in allowed_u} \left\{ \left[T_{uj}(t) \right] \cdot \left[\eta_u \right]^{\beta} \cdot \left[t_u \right]^{\beta} \right\} & if \ q \le q_0 \\ S & otherwise \end{cases} \tag{4.20}$$

where q is a uniformly distributed random number in [0,1], q_0 is a parameter ($0 \le q_0 \le 1$), and S is a random variable selected according to the new random-proportional rule given in Equation (4.19).

When we test the new method on medium and large size test problems (Buxey with 29 tasks, Gunther with 35 tasks and Mukherje with 94 tasks), we investigate that these modifications are effective and the trail values are accumulated in a consistent way. For illustrative purpose consider Buxey problem with cycle time of 36. In Figure 4.9, the trail accumulation for the ACS and the new method is given.

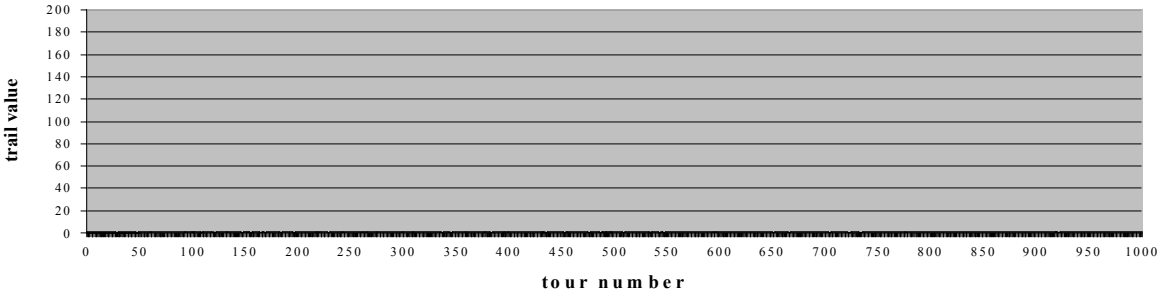


Figure 4.9.a: Trail accumulation for ACS, α =1, β =1, $\rho_1 = \rho_2$ = 0.4, q_0 =0.2, *initial trail* =1

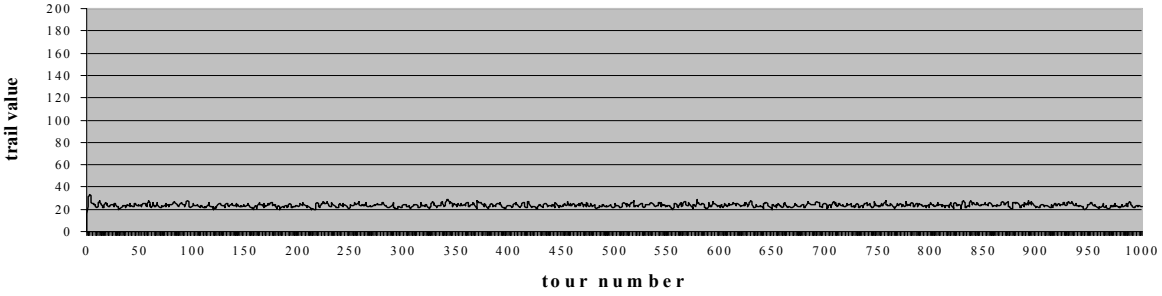


Figure 4.9.b: Trail accumulation for the new method, α =1, β =1, $\rho_1 = \rho_2$ =0.9, q_0 =0.3, *initial trail* =1

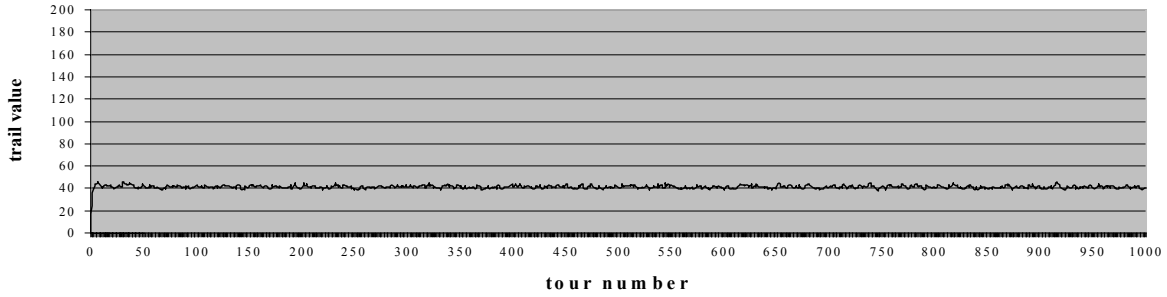


Figure 4.9.c: Trail accumulation for the new method, α =1, β =1, $\rho_1 = \rho_2$ =0.95, q_0 =0.3, *initial trail* =1

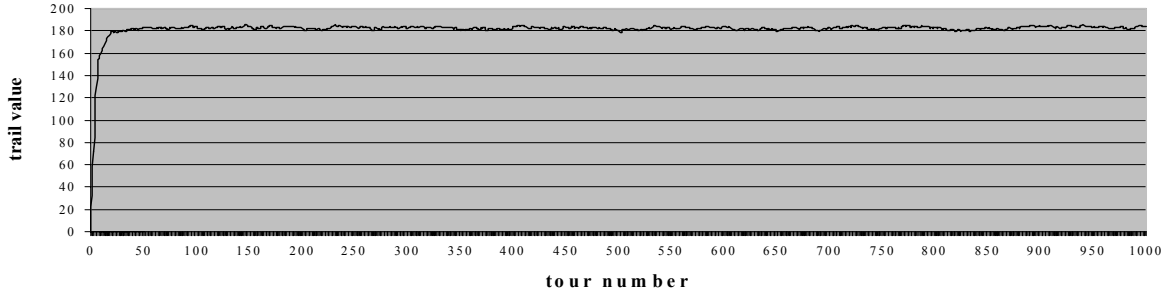


Figure 4.9.d: Trail accumulation for the new method, α =1, β =1, $\rho_1 = \rho_2$ =0.99, q_0 =0.3, *initial trail* =1

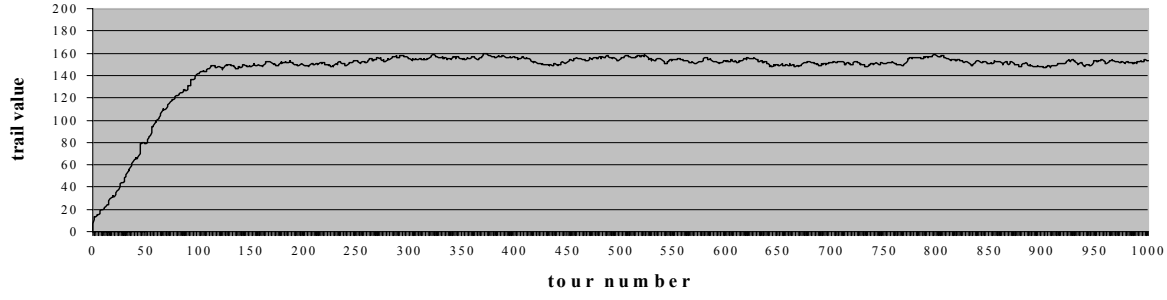


Figure 4.9.e: Trail accumulation for the new method, α =1, β =1, $\rho_1 = \rho_2$ =0.99, q_0 =0.8, *initial trail* =1

A little change in parameters ρ_1, ρ_2, *and* q_0 effects the trail accumulation. Trail values are not too small as they were in ACS and they accumulate more consistently. Also, the test results of this new method on medium size and large size problems indicates that defining locations for each station is not necessary, namely when the stations are not divided into locations, the algorithm performs better and finds the optimal solutions faster.

After a short period, the trail values are stabilized and the trail matrix reaches a steady-state. Short number of tours is necessary to emphasize the trail matrix. For test problems (Buxey with cycle time of 36, and Gunther with cycle time of 69) it is possible to find many matchings with Scholl and Klein's (1999) optimal task allocation.

Consider Buxey problem with 29 tasks and cycle time of 36. We have done three experiments; run the algorithm until 100, 250 and 1000 tours have been completed and store the trail matrices (The trail matrix for tour number 100, 250, 1000 is given in Appendix C).

Scholl and Klein's (1999) optimal task allocation for Buxey problem is given in Figure 4.10. In Table 4.3 we list the ranking of most possible station alternatives for each task depending on the trail matrices.

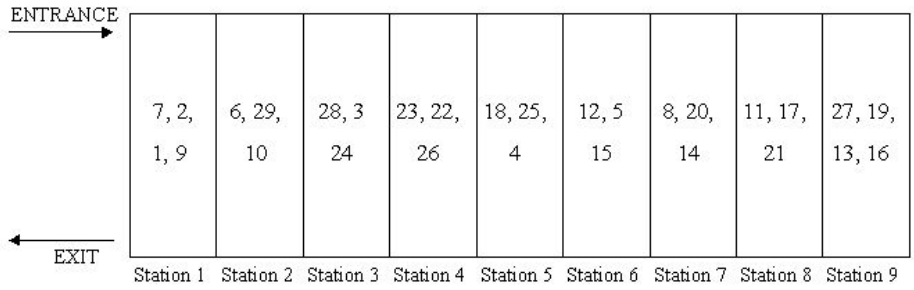


Figure 4.10: Scholl and Klein's (1999) optimal task allocation for Buxey problem

Table 4.3: Ranking of most possible location alternatives for each task depending on trail matrix.

Alternatives / Task	Experiment with 100 tours					Experiment with 250 tours					Experiment with 1000 tours				
	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
1	1	2	4	6	8	1	2	4	3	5	1	2	3	4	5
2	2	1	9	8	7	2	1	8	4	5	2	1	4	8	5
3	3	4	2	6	7	3	2	4	5	8	3	4	2	5	7
4	3	4	6	7	8	3	4	5	6	8	3	4	5	6	7
5	5	8	6	4	7	5	6	4	7	8	6	5	4	7	8
6	3	2	9	8	1	3	2	4	1	9	3	2	1	4	5
7	1	3	2	4	5	1	3	2	4	5	1	3	2	4	5
8	8	6	9	7	5	6	7	8	9	5	6	7	8	9	5
9	3	2	1	4	5	3	2	4	1	5	3	2	4	1	5
10	4	5	6	3	2	5	4	6	3	7	5	4	6	7	8
11	7	6	8	9	5	7	8	6	9	-	7	8	6	9	-
12	4	8	6	3	5	4	5	3	8	6	4	6	5	8	7
13	9	7	8	5	6	9	8	7	6	5	9	8	7	6	5
14	4	5	7	8	6	7	6	5	8	9	7	6	8	9	5
15	9	7	8	6	5	9	8	7	6	4	9	8	7	6	5
16	7	8	6	9	5	8	9	6	7	5	9	8	6	7	5
17	5	6	8	9	7	6	7	5	8	9	5	6	7	8	9
18	5	6	7	8	9	5	6	8	7	9	6	5	8	7	9
19	9	8	7	6	5	9	8	7	6	5	9	8	7	6	5
20	5	4	8	9	6	5	4	6	8	7	5	4	6	7	8
21	4	5	6	7	8	4	5	6	7	8	4	5	6	7	8
22	4	5	7	6	8	4	5	6	7	3	4	5	6	3	7
23	3	4	5	7	8	4	3	5	9	7	3	4	5	8	9
24	1	2	3	9	5	1	3	2	4	9	1	2	3	4	9
25	9	8	7	4	6	9	8	7	4	6	9	8	7	6	3
26	2	4	3	5	6	2	3	4	5	6	2	3	4	5	6
27	9	8	7	4	5	9	8	6	7	5	9	8	3	4	5
28	2	1	3	4	9	2	1	3	4	9	2	1	3	4	9
29	1	2	3	4	9	1	2	3	1	-	1	2	3	4	-
Total matching	**8**	**5**	**8**	**3**	**2**	**10**	**8**	**3**	**3**	**3**	**10**	**8**	**4**	**4**	**1**

When we compare our alternatives with Scholl and Klein's (1999) optimal task allocation, the results are encouraging. In Table 4.3 shaded cells represents the alternatives matching with Scholl and Klein's (1999) optimal task allocation. For most of the tasks, the first or the second alternative gives a matching, however there are many instances that the third or the fourth alternative also gives a matching.

This method is able to find optimal allocation for small, medium and large size problems that non-of the previous methods succeed. The performance of this method is very encouraging. However, for the two largest problems (Barthol2 problem with 148 tasks, and Scholl problem with 297 tasks) we need improvement.

Version 2.

Version 1 succeed almost all problems but it's performance is insufficient for the two largest problems.

Accumulation of the trail values is the reason of this situation. When the trail matrices are investigated (The trail matrices gathered for tour number 100, 250, 1000 are given in Appendix C) most of the entities in a column are so close to each other, namely for a station the trail values of the task alternatives are very close to each other. Also, when the task alternatives are too many (when the problem size is very large) it is very hard to find the proper task allocation.

Version 2 is the modified style of *Version 1* joined up with *a secondary global pheromone trail update mechanism* and works as follows:

$$T_{ij}(t) = T_{ij}(t) + \Delta T_{ij}^{(gb-full\ load)}(t) \tag{4.21}$$

where

$$\Delta T_{ij}^{(gb-full\ load)}(t) = \begin{cases} Q_2 & \text{if } (i,j) \text{ is part of the global best solution} \\ & \text{and } j \text{ is a full-loaded station} \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

After regular global pheromone update (as defined in Equation 4.17 and Equation 4.18), a secondary pheromone update is done for the globally best ants. The trail values of the tasks belonging to a full-loaded stations are reinforced with a very high number. Thus, after a very short time the trail values of these tasks will be distinguished easily. This situation can be easily observed for the trail matrices given in Appendix D.

We consider Buxey problem with cycle time of 36. We have done three experiments and run the algorithm until 100, 250 and 1000 tours have been completed and store the trail matrices (Trail matrix for tour number 100, 250, 1000 is given in Appendix D). Scholl and Klein's (1999) optimal task allocation for Buxey problem is given in Figure 4.10. In Table 4.4 we list the ranking of most possible station alternatives for each task depending on the trail matrices.

For *Version 2* matching rates of the first and the second alternatives are slightly higher than *Version 1*.

Version 2 succeeds for small, medium and large size problems and able to find the optimal allocation. Only there are a few instances that it finds such an allocation with one station more than the optimal. *Versions 2* also solves the Barthol2 problem with 148 tasks optimally. Only the largest problem Scholl with 297 tasks is a handicap and *Version 2* finds such an allocation with one station more than optimal.

Table 4.4: Ranking of most possible location alternatives for each task depending on trail matrix

Alternatives / Task	Experiment with 100 tours					Experiment with 250 tours					Experiment with 1000 tours				
	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th	1st	2nd	3rd	4th	5th
1	1	-	-	-	-	1	-	-	-	-	1	-	-	-	-
2	1	-	-	-	-	1	-	-	-	-	1	-	-	-	-
3	3	5	2	8	4	3	4	8	-	-	3	4	8	-	-
4	3	5	2	6	7	4	6	5	7	8	4	3	6	5	8
5	6	7	5	9	8	4	6	5	8	7	6	5	4	7	9
6	1	2	3	7	9	1	2	5	4	7	1	2	5	8	9
7	1	3	7	5	6	1	5	4	7	3	1	5	7	4	3
8	7	9	8	6	5	5	7	9	8	6	5	7	9	8	-
9	1	4	3	7	6	1	4	5	6	13	1	5	6	7	4
10	5	2	3	7	4	5	2	7	6	9	2	5	7	8	6
11	7	8	10	-	-	8	9	7	-	-	8	9	-	-	-
12	5	7	3	6	13	5	2	7	4	6	5	7	6	4	9
13	7	6	8	9	5	4	7	6	9	8	6	4	8	7	9
14	4	5	7	6	3	5	7	4	6	2	5	4	7	6	9
15	6	7	8	9	-	5	8	9	7	-	9	7	8	6	-
16	6	9	7	8	5	7	6	9	8	5	6	7	9	8	5
17	5	7	6	9	8	6	8	7	9	-	6	7	9	8	-
18	6	5	9	8	7	6	7	9	8	-	6	8	7	9	-
19	7	6	5	9	8	7	6	5	9	8	7	6	8	9	5
20	5	7	9	8	-	5	7	9	8	-	5	7	9	8	-
21	5	8	7	6	9	5	7	9	6	8	5	6	8	7	9
22	4	5	6	7	9	5	7	6	9	8	5	6	8	7	9
23	4	9	5	7	-	9	4	6	-	-	4	6	7	9	-
24	2	3	4	9	-	2	3	4	9	-	2	3	9	7	-
25	4	5	9	-	-	3	2	9	7	-	3	9	8	-	-
26	2	3	4	5	7	2	5	4	6	3	2	4	3	8	5
27	7	3	6	8	4	5	7	6	9	8	5	7	6	8	9
28	4	3	5	-	-	3	4	8	1	-	3	4	7	9	-
29	2	3	-	-	-	2	8	3	-	-	2	-	-	-	-
Total Matching	**11**	**11**	**2**	**3**	**1**	**8**	**9**	**3**	**3**	**2**	**11**	**5**	**4**	**4**	**2**

For *Version 2* matching rates of the first and the second alternatives are slightly higher than *Version 1*. Especially matching rates of the trail matrix gathered for 100 tour is higher than *Version 1*. These high matching rates indicates that the trail reinforcement mechanism of Version 2 is effective and less number of tours are enough to collect the trail matrix.

4.2.7 Ant Colony System Augmented with Simulated Annealing (ACS with SA)

We propose a modified version of ACS augmented with SA in order to improve the performance of ants. We intend to use SA as a support mechanism that works cooperatively with each ant. When a tour is completed, namely after an ant constructs a full solution, SA based mechanism efforts to improve the ants solution with *swap* (swapping of two tasks located in different stations), *insert* (inserting a task to an another station), and *repair* (instead of rejecting infeasible alternatives and generating a new alternative, repairing module repairs infeasible alternatives and makes them feasible, thus computational effort is saved) modules.

ACS with SA is only tested on very small size problems however it's performance is poor. Even for the second smallest problem Jackson with 11 elements, the computation time ranges between 2.53 hours and 164.63 hours.

4.2.8 Ant Colony System Augmented with Beam Search (ACS with BS)

A modified version of ACS augmented with beam search (BS) is proposed to direct the search in an intelligent way. As stated in Section 4.2.5 for ACS after a short period most of the entries of the trail matrix converge to zero and the remaining entries have a value, which is very close to zero. Somehow the valuable information intended to gain by the pheromone accumulation is lost. The

difference between very small values of the trail matrix and the positional weight values is very high. In this situation most of the information for the selection probabilities is coming from the heuristic value. Thus, the selection probabilities are mostly proportional to $[\eta_i]^\beta$ and the tasks with high positional weight are more likely to be selected. Another disturbing point is the structure of heuristic function. Heuristic function (positional weights) is static and it is not adapting itself (where as trail matrix does) as the search proceeds.

Instead of using positional weights we propose beam search to calculate the value of the heuristic function. Consider an ant *k;* just before allocating a task to station *j,* for each task *s* ($s \in \text{allowed}_s$) BS continues an imaginary allocation process. By using a very simple allocation mechanism, BS allocates the remaining tasks and calculates the station number of the imaginary allocation if that task *s* would have been selected for allocation. Instead of using positional weights, beam search supplies an estimate of the next move. Heuristic function values of the promising tasks will be higher than non-promising task. Therefore the selection probability of these promising tasks are expected to be high.

ACS with BS is tested on small and medium size problems. Structure of the beam search is very suitable for ACO however the performance of the algorithm is poor in terms of computational time. It requires excessive amount of time to complete a single tour.

Chapter 5

Experimental Setting

In this chapter, we present the parameters that we use in our numerical study and explain how we determine these parameters.

Cycle time (C), *task processing times (t)* and *number of tasks (n)* are the inputs of the algorithm. The other parameters such as *number of ants (m)*, α, β, ρ, ρ_1, ρ_2, Q, Q_2, q_0, τ_0 are also used in the algorithm.

In the following section we briefly explain how to set up and fine tune these parameters for various versions of the proposed ant algorithms.

The performances of the proposed algorithms are tested by using the benchmark problems available in the literature. In the U-type assembly line literature, there are a number of test problems used by several researchers (Scholl and Klein, 1999 classify these problems into three data sets: Talbot et al., 1986; Hoffmann, 1990, 1992; Scholl, 1993). These data sets are available at:

(i) http://www.assembly-line-balancing.de/

(ii) http://www.wiwi.uni-jena.de/Entscheidung/alb/

(iii) http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb/

The first data set considers 64 instances with varying problem sizes ranging from 8 to 111 tasks. The second data set considers 50 instances with varying problem sizes ranging from 30 to 111 tasks. 13 of these instances are also

contained in Set 1. The last data set is relatively new and most complex data set such that it considers 168 instances with varying problem sizes ranging from 25 to 297 tasks.

We use proposed algorithms (*Ant System*, *Ant System with Elitist Strategy*, *Ant Colony System*, *New Ant Colony Optimization Approach; Version 1* and *Version 2*) to solve 190 instances from these data sets and compare the results with ULINO (Scholl and Klein, 1999) and simulated annealing based algorithm of Erel, Sabuncuoglu and Aksu (2001). The computational results of SA based algorithm are provided only for 190 problem instances. Therefore we restrict ourselves only to those problems instead of whole data set. These data sets that we consider are given as follows:

Data Set 1:

Bowman (8 tasks, $C = 20$);

Mansoor (11 tasks, $C = 48, 62, 94$);

Jackson (11 tasks, $C = 7, 9, 10, 13, 14, 21$);

Mitchell (21 tasks, $C = 14, 15, 21, 26, 35, 39$);

Arcus2 (111 tasks, $C = 5755, 8847, 10027, 10743, 11378, 17067$);

Data Set 3:

Roszieg (25 tasks, $C = 14, 16, 18, 21, 25, 32$);

Buxey (29 tasks, $C = 27, 30, 33, 36, 41, 47, 54$);

Lutz1 (32 tasks, $C = 1414, 1572, 1768, 2020, 2357, 2828$);

Gunther (35 tasks, $C = 41, 44, 49, 54, 61, 69, 81$);

Hahn (53 tasks, $C = 2004, 2338, 2806, 3507, 4676$);

Warnecke (58 tasks, $C = 54, 56, 58, 60, 62, 65, 68, 71, 74, 78, 82, 86, 92, 97, 104,111$);

Wee-mag (75 tasks, $C = 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 45, 46, 47, 49, 50, 52, 54, 56$);

Lutz2 (89 tasks, $C = 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21$);

Lutz3 (89 tasks, *C = 75, 79, 83, 87, 92, 97, 103, 110, 118, 127, 137, 150*);

Mukherje (94 tasks, *C = 176, 183, 192, 201, 211, 222, 234, 248, 263, 281, 301, 324, 351*);

Arcus2 (111 tasks, *C = 5755, 8847, 10027, 10743, 11378, 17067*);

Barthold (148 tasks, *C = 403, 434, 470, 513, 564, 626, 705, 805*);

Barthol2 (148 tasks, *C = 84, 85, 87, 89, 91, 93, 95, 97, 99, 101, 104, 106, 109, 112, 115, 118, 121, 125, 129, 133, 137, 142, 146, 152, 157, 163, 170*)

Scholl (297 tasks, *C = 1394, 1422, 1452, 1483, 1515, 1548, 1584, 1620, 1659, 1699, 1742, 1787, 1834, 1883, 1935, 1991, 2049, 2111, 2177, 2247, 2322, 2402, 2488, 2580, 2680, 2787*)

Task Processing Time (t) is the duration to complete a task. In this research task processing times are assumed to be deterministic.

Station Time (ST) is the sum of the processing times of tasks that are performed at the same station. The station time should not exceed the cycle time.

Cycle Time (C) is the time interval between two successive outputs of a station. The cycle time of a worker is defined as the time interval between his consecutive arrivals at his first task, and consists of operation times, walking time between tasks which are located at the entrance side and at the exit side. In this research walking times are assumed to be zero. A new item can only enter a station only after a product is completed. Cycle time of a station is bounded by the maximum task time of the tasks allocated to that station and total duration of all tasks ($\max_{i \in S_k} t_i \le C \le \sum_{i=1}^{n} t_i$; S_k is the set of assigned to station k = 1,..,*K*). Also total operation time of tasks assigned to a station *k*, station time, should not exceed the cycle time ($\sum_{i \in S_k} t_i \le C \quad k = 1,..,K$).

Scholl and Klein (1999) report the optimal station number for most of the problem instances, however some of them are given in the interval. A simple lower bound on the minimal number of stations is equal to $LB = \left\lceil \sum_{i=1}^{n} t_i / C \right\rceil$, where $\lceil x \rceil$ is the smallest integer larger than x (Scholl and Klein, 1999; Erel, Sabuncuoglu and Aksu, 2001). Besides the simple lower bound *LB,* Scholl and Klein (1999) use three additional bound arguments. The upper bound on the minimal number of stations is proposed to be the required number of stations after all the tasks are assigned and a feasible solution is found.

Number of Tasks (n) for each problem is given at Scholl and Klein's web site. All related data (number of tasks, task processing times and precedence relations) is given as a compressed file. This data file can be downloaded from this web address (http://www.wiwi.uni-jena.de/Entscheidung/alb/albdata.zip).

Number of Ants (m): Each ant is a problem-solving agent. Cooperation between ants is one of the important characteristics of the ACO metaheuristics. In fact, although a single ant is capable of construct a solution, better solutions are found when a colony of ants are used. Good solutions are exposed when these agents interact with each other (by using trail values) and work in cooperation. Dorigo and Stütze (2000) suggest that ACO algorithms perform better when the number of ants is set to a value $m > 1$. Dorigo, Maniezzo and Colorni (1991, 1996); Colorni, Dorigo and Maniezzo (1991, 1992) also suggest that the optimal number of ants should be taken close to the number of cities ($m \approx n$) for TSP. Our test results support this fact. It is better to take number of ants m equal to number of tasks n.

We have carried out a set of experiments in order to test the effect of the number of ants on the performance of the proposed algorithm. A small size (Jackson with 11 tasks and cycle time of 10), a medium size (Gunther with 35 tasks and cycle time of 54) and a large size problem (Barthold with 148 tasks and

cycle time of 805) are chosen as the test problems. The proposed algorithms perform better when the number of ants is taken equal to the number of tasks. We evaluate the performance of proposed algorithms varying number *m* of ants from 1 to 2*n*, given (100 replications • 100 ant tours) for Jackson and Gunther problem and (1 replications • 100 ant tours) for Barthold problem. The results are given in the Section 5.1.

Parameters α *and* β allow a user to control the relative importance of pheromone trail versus heuristic information (visibility). These parameters affect the behaviour of the algorithm. If α = 0, the selection probabilities are proportional to $[\eta_i]^\beta$ and the tasks with high positional weight are more likely to be selected. In this case AS corresponds to a stochastic greedy algorithm with multiple starts. If β = 0, only the trail information effects the selection probabilities and if no control mechanism exists this situation misleads the ants. This may cause a very rapid convergence, leading to a *stagnation* situation (Dorigo, Maniezzo, Colorni, 1996). In this situation all the ants follow the same path and construct the same solutions; strongly suboptimal solutions (see also section 4.1.2.1 and section 4.1.2.2). Thus, there is a trade-off between the trail intensity and the heuristic value.

Parameters ρ, ρ_1 *and* ρ_2 are called as *pheromone trail evaporation rate* ($0 < \rho, \rho_1, \rho_2 < 1$). Evaporation enables the algorithm to forget the previously done bad selections and unlimited accumulation of the pheromone trails is avoided by this way.

Parameter q_0 is used in *Ant Colony System*, *Modified Ant Colony System with Random Search*, *New Ant Colony Optimization Approach; Version 1* and *Version 2.* It controls the relative importance of exploitation versus exploration. If $q \leq q_0$ then the best task is chosen and this selection is a kind of greedy behaviour which favours the exploitation of the search space. Otherwise a task is chosen

according to *random-proportional rule* and that favours the exploration of the search space.

Parameter Q and Q_2 are the constants related to the quantity of trail laid by ants. The amount of reinforcement that the related trail value receives is controlled by these parameters.

Parameter τ_0 is the amount of pheromone reinforcement that the related trail values receives during the local pheromone update in *Ant Colony System*, *Modified Ant Colony System with Random Search*, *New Ant Colony Optimization Approach; Version 1* and *Version 2.*

We have done a second set of experiments in which we study the performance of the proposed algorithms with respect to α, β, ρ, ρ_1, ρ_2 and q_0 using the previous test problems. The number of ants *m,* is taken to be equal to the number of tasks *n*. We test several values for each parameter. The values being tested are: $\alpha \in \{0, 1, 2, 5\}$, $\beta \in \{0, 1, 2, 5\}$, ρ (also ρ_1 and ρ_2) $\in \{0.1, 0.4, 0.7, 0.9, 0.99\}$ and $q_0 \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. A small size (Jackson with 11 tasks and cycle time of 10), a medium size (Gunther with 35 tasks and cycle time of 54) and a large size problem (Barthold with 148 tasks and cycle time of 805) are chosen as the test problems. We evaluate the performance of proposed algorithms given a (100 replications • 100 ant tours) for Jackson and Gunther problem and (1 replications • 100 ant tours) for Barthold problem.

The results are given in Tables 5.1-5.6. In each cell the first element represents the results of Jackson problem, the second element represents the results of Gunther problem and the third element represents the results of Barthold problem. The results are reported like $X^{(Y)}$; where *X* represents the number of stations found after *(Y)* number of replications completed (tours for Barthold problem). When the optimal number of stations is obtained, *(Y)* is not reported.

5.1 Experimental Setting for AS

5.1.1 Number of Ants

The effect of the number of ants on the efficiency of the Ant System is given in Figure 5.1. The abscissa represents the total number of ants used in each set of replication and the ordinate represents the number of replications (tours for Barthold problem) required to obtain the optimum.

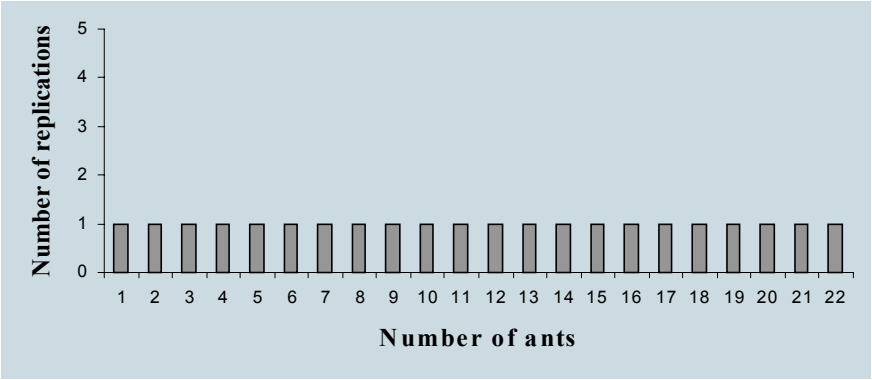


Figure 5.1.a: Number of replications required to obtain the optimum number of stations. Jackson problem with 11 tasks, *C*=10. The experiment has been carried out for (100 replications • 100 ant tours).

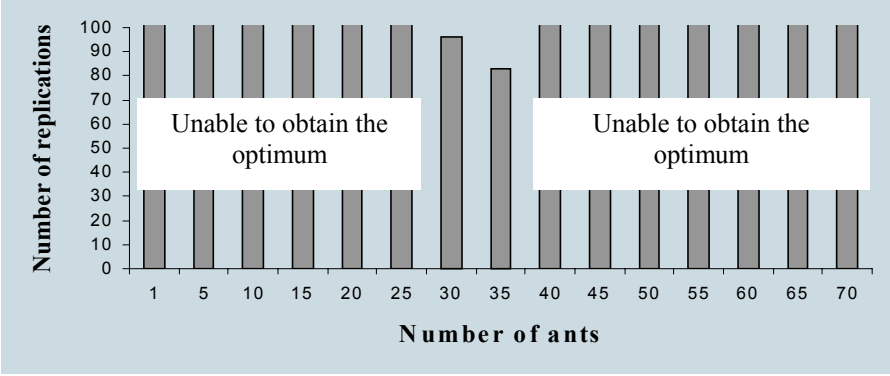


Figure 5.1.b: Number of replications required to obtain the optimum number of stations. Gunther problem with 35 tasks, *C*=54. The experiment has been carried out for (100 replications • 100 ant tours).

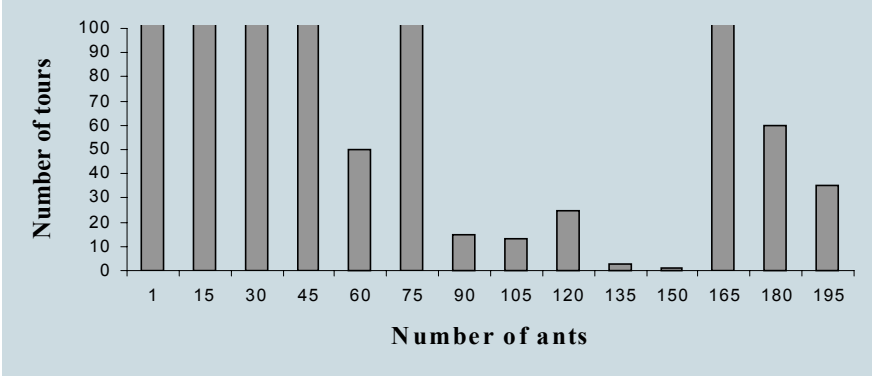


Figure 5.1.c: Number of tours required to obtain the optimum number of stations. Barthold problem with 148 tasks, *C*=805. The experiment has been carried out for (1 replications • 100 ant tours).

For Jackson problem, AS finds the optimal station number for any number of ants at the first replication. For Gunther problem, the optimum is obtained at replications 96 and 83 with 30 and 35 ants, respectively. For Barthold problem, the minimum number of tours to obtain the optimum is 1 with 150 ants. Depending on the results we take the number of ants equal to the number of tasks.

5.1.2 Parameters Setting

The performance of AS with respect to the parameters α, β, and ρ is given in Table 5.1. We take the number of ants m, equal to the number of tasks n. The results indicate that the optimal solution is found at the first replication (tour for Barthold problem) when the parameters are taken as: $\alpha=2$, $\beta=2$ and $\rho=0.7$. The shaded cell in Table 5.1 indicates the best set of these parameters.

Table 5.1: Fine tune-up of the parameters α, β, and ρ for AS.

$\rho = 0.1$				
β \ α	**0**	**1**	**2**	**5**
0	$5^{(1)}$ / $9^{(23)}$ / $7^{(15)}$	$5^{(1)}$ / $9^{(12)}$ / $7^{(21)}$	$5^{(1)}$ / 10 / $7^{(33)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(12)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(62)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(2)}$
2	$5^{(1)}$ / $9^{(4)}$ / $7^{(8)}$	$5^{(1)}$ / $9^{(57)}$ / $7^{(10)}$	$5^{(1)}$ / 10 / $7^{(23)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(5)}$	$5^{(1)}$ / 10 / 8
$\rho = 0.4$				
0	$5^{(1)}$ / $9^{(7)}$ / $7^{(6)}$	$5^{(1)}$ / 10 / $7^{(18)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(3)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(29)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(48)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(6)}$	$5^{(1)}$ / 10 / $7^{(6)}$	$5^{(1)}$ / $9^{(75)}$ / $7^{(3)}$
$\rho = 0.7$				
0	$5^{(1)}$ / $9^{(3)}$ / $7^{(11)}$	$5^{(1)}$ / $9^{(11)}$ / $7^{(8)}$	$5^{(1)}$ / 10 / $7^{(23)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(18)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(30)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / 8
5	$5^{(1)}$ / 10 / $7^{(6)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / $9^{(8)}$ / $7^{(5)}$	$5^{(1)}$ / 10 / 8
$\rho = 0.9$				
0	$5^{(1)}$ / $9^{(5)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(81)}$	$5^{(1)}$ / $9^{(9)}$ / 8	$5^{(1)}$ / $9^{(1)}$ / 8
1	$5^{(1)}$ / $9^{(17)}$ / $7^{(5)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(14)}$ / $7^{(9)}$	$5^{(1)}$ / 10 / $7^{(7)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(5)}$	$5^{(1)}$ / 10 / 8
$\rho = 0.99$				
0	$5^{(1)}$ / $9^{(26)}$ / $7^{(9)}$	$5^{(1)}$ / 10 / $7^{(23)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(61)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(47)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(2)}$

5.2 Experimental Setting for AS_{elite}

5.2.1 Number of Ants

The effect of the number of ants on the efficiency of the Ant System with elitist strategy is given in Figure 5.2. The abscissa represents the total number of ants used in each set of replication and the ordinate represents the number of replications (tours for Barthold problem) required to obtain the optimum.

For Jackson problem, AS_{elite} finds the optimal solution for any number of ants at the first replication. For Gunther problem, minimum number of replications to reach the optimum is 18 with 35 ants. For Barthold problem, the minimum number of tours to obtain the optimum is 4 with 150 ants. Fewer number of replications (tours for Barthold problem) are required to obtain the optimal station number, when the number of ants is nearly equal to the number of tasks.

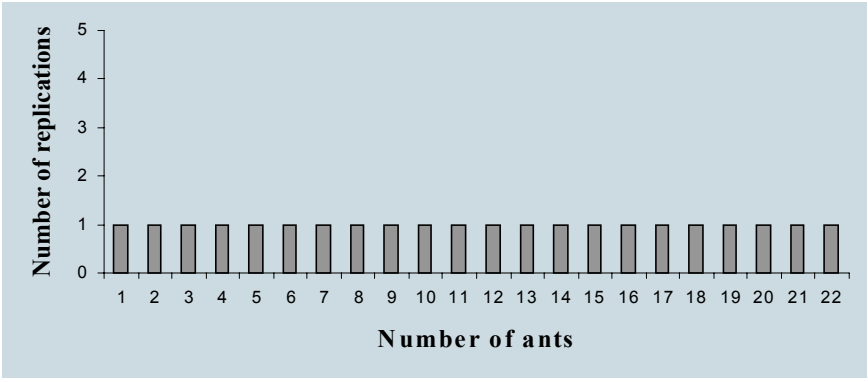


Figure 5.2.a: Number of replications required to obtain the optimum number of stations. Jackson problem with 11 tasks, *C*=10. The experiment has been carried out for (100 replications • 100 ant tours).

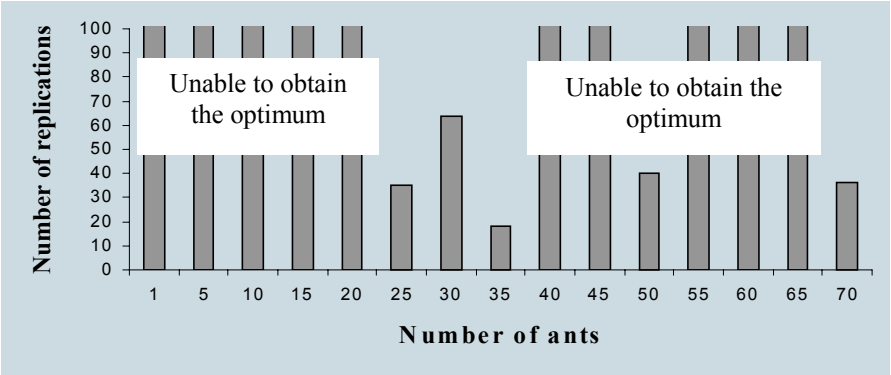


Figure 5.2.b: Number of replications required to obtain the optimum number of stations. Gunther problem with 35 tasks, *C*=54. The experiment has been carried out for (100 replications • 100 ant tours).

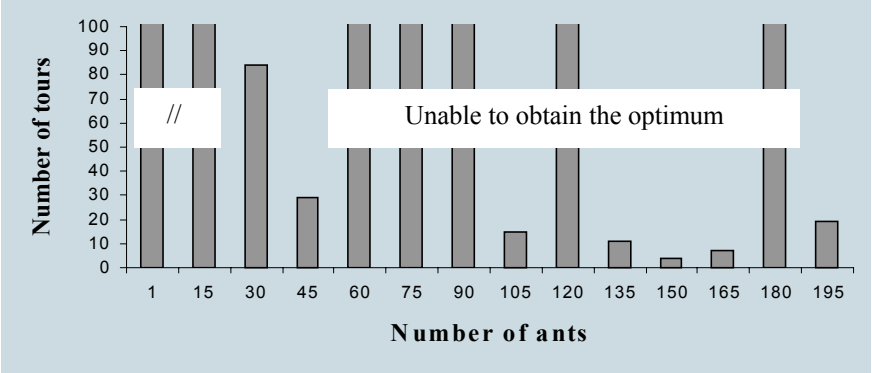


Figure 5.2.c: Number of tours required to obtain the optimum number of stations. Barthold problem with 148 tasks, *C*=805. The expeiment has been carried out for (1 replications • 100 ant tours).

5.2.2 Parameters Setting

The performance of AS_{elite} with respect to the parameters α, β, and ρ is given in Table 5.2. We take the number of ants m, equal to the number of tasks n. The results indicate that when the parameters are taken as α=2, β=1 and ρ=0.9 the optimum solution is found at the first replication (tour for Barthold problem). In Table 5.2 the shaded cell indicates the best set of parameters.

Table 5.2: Fine tune-up of the parameters α, β, and ρ for AS_{elite}.

$\rho = 0.1$				
β \ α	**0**	**1**	**2**	**5**
0	$5^{(1)}$ / $9^{(13)}$ / $7^{(52)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / $7^{(45)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(3)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(11)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(17)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(76)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(3)}$
$\rho = 0.4$				
0	$5^{(1)}$ / $9^{(2)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(49)}$	$5^{(1)}$ / 10 / $7^{(6)}$	$5^{(1)}$ / $9^{(50)}$ / 8
1	$5^{(1)}$ / $9^{(21)}$ / $7^{(12)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / $9^{(52)}$ / $7^{(13)}$	$5^{(1)}$ / $9^{(1)}$ / 8
2	$5^{(1)}$ / $9^{(14)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(5)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / 8
$\rho = 0.7$				
0	$5^{(1)}$ / $9^{(9)}$ / $7^{(7)}$	$5^{(1)}$ / $9^{(30)}$ / $7^{(56)}$	$5^{(1)}$ / $9^{(1)}$ / 8	$5^{(1)}$ / $9^{(1)}$ / 8
1	$5^{(1)}$ / $9^{(15)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$
2	$5^{(1)}$ / $9^{(28)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(73)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$
5	$5^{(1)}$ / 10 / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(26)}$
$\rho = 0.9$				
0	$5^{(1)}$ / $9^{(12)}$ / $7^{(9)}$	$5^{(1)}$ / 10 / $7^{(58)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(4)}$ / $7^{(11)}$	$5^{(1)}$ / $9^{(30)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / 8
2	$5^{(1)}$ / $9^{(75)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(7)}$	$5^{(1)}$ / 10 / $8^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / 8
$\rho = 0.99$				
0	$5^{(1)}$ / $9^{(54)}$ / $7^{(22)}$	$5^{(1)}$ / $9^{(66)}$ / $7^{(78)}$	$5^{(1)}$ / 10 / $7^{(12)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(7)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(21)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(6)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / 8	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(87)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8

5.3 Experimental Setting for ACS

5.3.1 Number of Ants

The effect of the number of ants on the efficiency of the ACS is given in Figure 5.3. The abscissa represents the total number of ants used in each set of replication and the ordinate represents the number of replications (tours for Barthold problem) required to obtain the optimum.

For Jackson problem, ACS finds the optimal station number for any number of ants at the first replication. For Gunther problem, the optimum is obtained with minimum number of replications when more than 25 ants are used. It is possible to obtain the optimum with minimum number of replications for 55, 60, 65, 70 ants. However, using fewer ants (30 and 35 ants) reduces the computational effort. For Barthold problem, the optimum is obtained with minimum number of tours when more than 25 ants are used (In this experiment, it is not possible to test ACS with 165, 180 and 195 ants due to memory requirements). Fewer number of replications (tours for Barthold problem) are required to obtain the optimal station number, when the number of ants is nearly equal to the number of tasks.

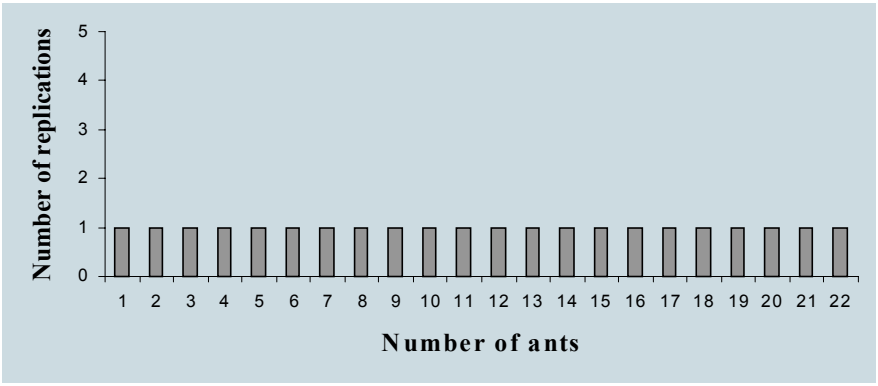


Figure 5.3.a: Number of replications required to obtain the optimum number of stations. Jackson problem with 11 tasks, C=10. The experiment has been carried out for (100 replications • 100 ant tours).

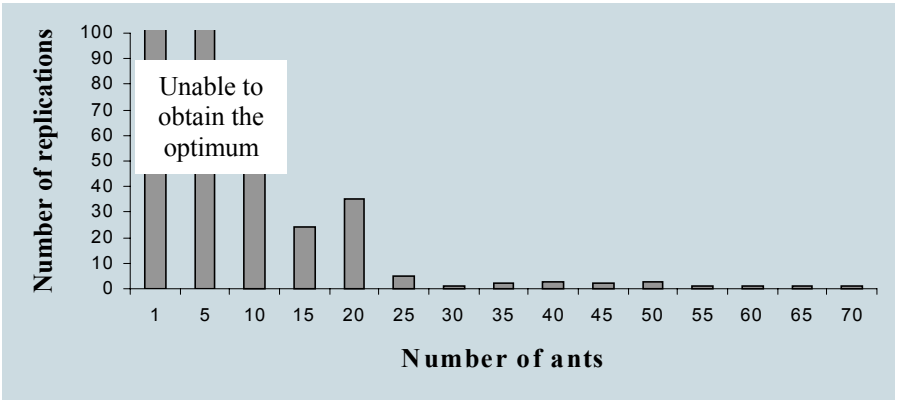


Figure 5.3.b: Number of replications required to obtain the optimum number of stations. Gunther problem with 35 tasks, C=54. The experiment has been carried out for (100 replications • 100 ant tours).

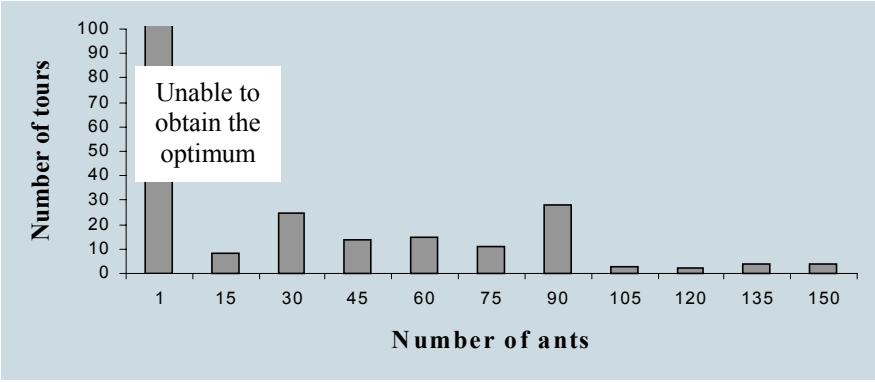


Figure 5.3.c: Number of tours required to obtain the optimum number of stations. Barthold problem with 148 tasks, C=805. The experiment has been carried out for (1 replications • 100 ant tours).

5.3.2 Parameters Setting

The performance of ACS with respect to the parameters β, q_0, ρ_1 and ρ_2 is given in Table 5.3. We take the number of ants *m,* equal to the number of tasks *n*. The results indicate that the optimal solution is found at the first replication (tour for Barthold problem) when the parameters are taken as: β= 1, q_0= 0.2 and $\rho_1=\rho_2= 0.4$. The shaded cell in Table 5.3 indicates the best set of parameters.

Table 5.3: Fine tune-up of the parameters β, q_0, ρ_1 and ρ_2 for ACS.

$\rho_1 = \rho_2 = 0.1$						
q_0 / β	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**
0	$5^{(1)}$ / $9^{(6)}$ / $7^{(45)}$	$5^{(1)}$ / $9^{(26)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(9)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(37)}$ / $7^{(11)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(22)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(12)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(36)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(32)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(24)}$ / $7^{(12)}$	$5^{(1)}$ / $9^{(26)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(3)}$	$5^{(1)}$ / $9^{(46)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(6)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.4$						
0	$5^{(1)}$ / $9^{(6)}$ / $7^{(29)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(49)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(17)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(49)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(28)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(26)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(7)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(5)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(83)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(23)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(28)}$ / $7^{(2)}$	$5^{(1)}$ / 10 / $7^{(4)}$	$5^{(1)}$ / $9^{(21)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.7$						
0	$5^{(1)}$ / $9^{(2)}$ / $7^{(11)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(22)}$	$5^{(1)}$ / $9^{(3)}$ / $7^{(16)}$	$5^{(1)}$ / $9^{(7)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(2)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(9)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(5)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(7)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(7)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(17)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / $7^{(1)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(6)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.9$						
0	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(3)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(89)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(24)}$	$5^{(1)}$ / $9^{(17)}$ / $7^{(25)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(7)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(15)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(10)}$	$5^{(1)}$ / $9^{(3)}$ / $7^{(8)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(8)}$	$5^{(1)}$ / $9^{(13)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(2)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(34)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(24)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(21)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(22)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.99$						
0	$5^{(1)}$ / $9^{(3)}$ / $7^{(27)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(22)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(26)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(5)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(3)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(2)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(15)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(15)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(26)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8

5.4 Experimental Setting for Modified ACS with Random Search

The *Modified ACS with Random Search* is an extension of ACS. Thus, the same parameter set is used in this algorithm and the number of ants is taken equal to the number of tasks.

5.5 Experimental Setting for New ACO Approach, Version 1

5.5.1 Number of Ants

The effect of the number of ants on the efficiency of the new ACO approach *(Version 1)* is given in Figure 5.4. The abscissa represents the total number of ants used in each set of replication and the ordinate represents the number of replications (tours for Barthold problem) required to obtain the optimum.

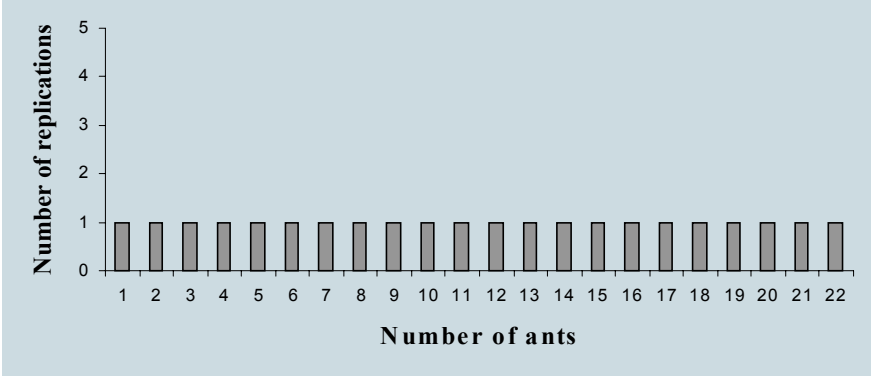


Figure 5.4.a: Number of replications required to obtain the optimum number of stations. Jackson problem with 11 tasks, *C*=10. The experiment has been carried out for (100 replications • 100 ant tours).

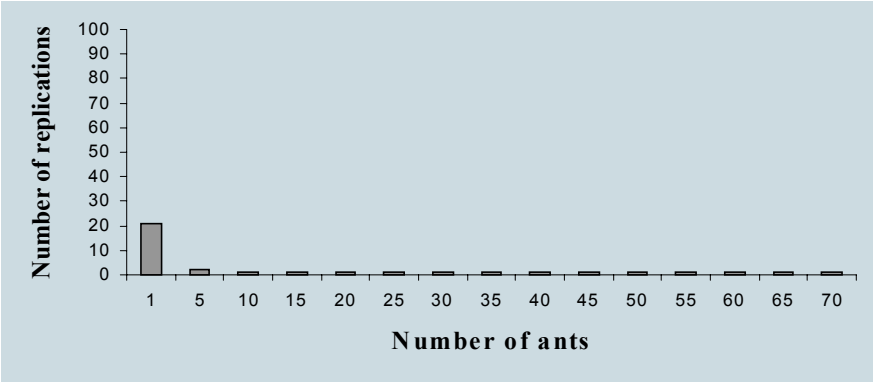


Figure 5.4.b: Number of replications required to obtain the optimum number of stations. Gunther problem with 35 tasks, *C*=54. The experiment has been carried out for (100 replications • 100 ant tours).

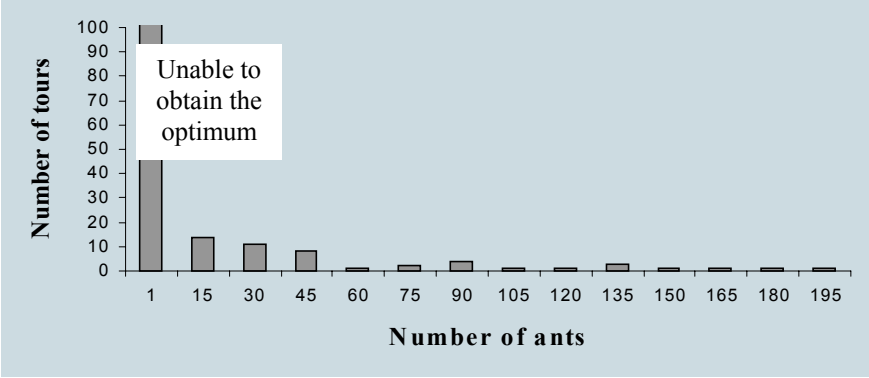


Figure 5.4.c: Number of tours required to obtain the optimum number of stations. Barthold problem with 148 tasks, *C*=805. The experiment has been carried out for (1 replications • 100 ant tours).

For Jackson problem, *Version 1* finds the optimal solution for any number of ants at the first replication. For Gunther problem, the optimum is obtained with minimum number of replications when more than 5 ants are used. For Barthold problem, *Version 1* yields the optimum with less number of tours when more than 45 ants are used. Version 1 finds the optimum at the first replication for 60, 105, 120, 150, 165, 180 and 195 ants. *Version 1* works efficiently when there is more ants. For Barthold problem, although the optimal is obtained at the first replication for 60, 105 and 120 ants, *Version 1* finds the optimum at third replication with 135 ants. Therefore it is more reliable to take the number of ants equal to the number of tasks.

5.5.2 Parameters Setting

The performance of *Version 1* with respect to the parameters α, q_0, ρ_1 and ρ_2 is given in Table 5.4. β is left constant and taken as 1. We take the number of ants *m,* equal to the number of tasks *n*.

The results indicate that the optimal solution is found at the first replication (tour for Barthold problem) for various values of the parameters: α, q_0, ρ_1 and ρ_2. The shaded cells in Table 5.4 indicate the best set of parameters.

The performance of *Version 1* with respect to β, q_0, ρ_1 and ρ_2 is given in Table 5.5. α is left constant and taken as 1. We take the number of ants *m,* equal to the number of tasks *n*.

Referring to the results given in Table 5.5, the optimal solution is found at the first replication (tour for Barthold problem) for various values of the parameters: β, q_0, ρ_1 and ρ_2. The shaded cells in Table 5.5 indicate the best set of parameters.

It is hard to determine a single set of parameters. Therefore, we have carried out another set of experiment on Buxey problem with 29 tasks and cycle time of 36, given (100 replications • 100 ant tours). Keeping β as 1, we first analyze the performance of *Version 1* with respect to α, $q_{0,}$ ρ_1 and ρ_2. Then keeping α as 1 we investigate the behaviour of *Version 1* with respect to β, $q_{0,}$ ρ_1 and ρ_2.

Referring to the results given in Table 5.6, in each cell the first element represents the result of the first experiment and the second element represents the result of the second one. Note that the number of ants is set equal to the number of tasks.

Table 5.4: Fine tune-up of parameters α, q_0, ρ_1 and ρ_2 for New ACO Approach, Version 1.

$\rho_1 = \rho_2 = 0.1$						
α \ q_0	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**
0	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
1	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(3)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
2	$5^{(1)} / 9^{(2)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(2)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
5	$5^{(1)} / 9^{(3)} / 7^{(1)}$	$5^{(1)} / 9^{(3)} / 7^{(3)}$	$5^{(1)} / 9^{(2)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(7)} / 7^{(3)}$	$5^{(1)} / 10 / 8$
$\rho_1 = \rho_2 = 0.4$						
0	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
1	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(4)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
2	$5^{(1)} / 9^{(2)} / 7^{(3)}$	$5^{(1)} / 9^{(1)} / 7^{(3)}$	$5^{(1)} / 9^{(1)} / 7^{(3)}$	$5^{(1)} / 9^{(2)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
5	$5^{(1)} / 9^{(1)} / 7^{(3)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(5)} / 7^{(2)}$	$5^{(1)} / 9^{(5)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
$\rho_1 = \rho_2 = 0.7$						
0	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
1	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
2	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
5	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(5)} / 7^{(9)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(4)} / 7^{(2)}$	$5^{(1)} / 9^{(5)} / 7^{(11)}$	$5^{(1)} / 10 / 8$
$\rho_1 = \rho_2 = 0.9$						
0	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(3)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
1	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
2	$5^{(1)} / 9^{(2)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(2)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(5)} / 7^{(1)}$	$5^{(1)} / 10 / 8$
5	$5^{(1)} / 9^{(35)} / 7^{(8)}$	$5^{(1)} / 9^{(7)} / 7^{(12)}$	$5^{(1)} / 9^{(19)} / 7^{(13)}$	$5^{(1)} / 10 / 7^{(4)}$	$5^{(1)} / 9^{(24)} / 7^{(8)}$	$5^{(1)} / 10 / 8$
$\rho_1 = \rho_2 = 0.99$						
0	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 10 / 8$
1	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(1)} / 7^{(2)}$	$5^{(1)} / 9^{(2)} / 7^{(6)}$	$5^{(1)} / 10 / 8$
2	$5^{(1)} / 9^{(4)} / 7^{(5)}$	$5^{(1)} / 9^{(2)} / 7^{(1)}$	$5^{(1)} / 9^{(1)} / 7^{(6)}$	$5^{(1)} / 9^{(14)} / 7^{(5)}$	$5^{(1)} / 9^{(11)} / 7^{(22)}$	$5^{(1)} / 10 / 8$
5	$5^{(1)} / 10 / 7^{(21)}$	$5^{(1)} / 10 / 8$	$5^{(1)} / 10 / 7^{(1)}$	$5^{(1)} / 9^{(39)} / 8$	$5^{(1)} / 9^{(43)} / 8$	$5^{(1)} / 10 / 8$

Table 5.5: Fine tune-up of the parameters β, q_0, ρ_1 and ρ_2 for New ACO Approach, Version 1.

$\rho_1 = \rho_2 = 0.1$						
q_0 / β	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**
0	$5^{(1)}$ / $9^{(6)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(16)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(17)}$ / $7^{(32)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(20)}$	$5^{(1)}$ / $9^{(16)}$ / $7^{(42)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.4$						
0	$5^{(1)}$ / $9^{(6)}$ / $7^{(6)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(5)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(66)}$	$5^{(1)}$ / $9^{(94)}$ / $7^{(14)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(3)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.7$						
0	$5^{(1)}$ / $9^{(3)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(32)}$ / $7^{(19)}$	$5^{(1)}$ / $9^{(13)}$ / $7^{(31)}$	$5^{(1)}$ / $9^{(34)}$ / $7^{(50)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(5)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(3)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(9)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.9$						
0	$5^{(1)}$ / $9^{(11)}$ / $7^{(12)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(24)}$	$5^{(1)}$ / $9^{(60)}$ / $7^{(10)}$	$5^{(1)}$ / $9^{(6)}$ / $7^{(39)}$	$5^{(1)}$ / $9^{(87)}$ / $7^{(16)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(10)}$ / $7^{(1)}$	$5^{(1)}$ / 10 / 8
$\rho_1 = \rho_2 = 0.99$						
0	$5^{(1)}$ / $9^{(29)}$ / $7^{(81)}$	$5^{(1)}$ / $9^{(84)}$ / $7^{(7)}$	$5^{(1)}$ / $9^{(35)}$ / $7^{(19)}$	$5^{(1)}$ / 10 / $7^{(76)}$	$5^{(1)}$ / 10 / $7^{(66)}$	$5^{(1)}$ / 10 / 8
1	$5^{(1)}$ / $9^{(1)}$ / $7^{(4)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(5)}$	$5^{(1)}$ / 10 / 8
2	$5^{(1)}$ / $9^{(1)}$ / $7^{(2)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(10)}$	$5^{(1)}$ / 10 / 8
5	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(1)}$ / $7^{(1)}$	$5^{(1)}$ / $9^{(2)}$ / $7^{(3)}$	$5^{(1)}$ / $9^{(4)}$ / $7^{(6)}$	$5^{(1)}$ / 10 / 8

Table 5.6: Fine tune-up of parameters α, β, q_0, ρ_1 and ρ_2 for New ACO Approach, Version 1.

$\rho_1 = \rho_2 = 0.1$						
α or β \ q_0	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**
0	$9^{(85)}$ / 10	$9^{(46)}$ / 10	$9^{(55)}$ / 10	10 / 10	10 / 10	10 / 10
1	10 / 10	$9^{(22)}$ / 10	10 / 10	10 / 10	10 / 10	10 / 10
2	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
5	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
$\rho_1 = \rho_2 = 0.4$						
0	$9^{(35)}$ / 10	$9^{(89)}$ / 10	$9^{(80)}$ / 10	10 / 10	10 / 10	10 / 10
1	10 / $9^{(69)}$	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
2	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
5	$9^{(29)}$ / 10	10 / $9^{(77)}$	10 / 10	10 / 10	10 / 10	10 / 10
$\rho_1 = \rho_2 = 0.7$						
0	10 / 10	$9^{(63)}$ / 10	$9^{(81)}$ / 10	$9^{(64)}$ / 10	10 / 10	10 / 10
1	10 / 10	$9^{(54)}$ / 10	10 / 10	10 / 10	10 / 10	10 / 10
2	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
5	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
$\rho_1 = \rho_2 = 0.9$						
0	10 / 10	10 / 10	$9^{(75)}$ / 10	$9^{(37)}$ / 10	10 / 10	10 / 10
1	10 / 10	$9^{(5)}$ / 10	$9^{(76)}$ / 10	10 / 10	10 / 10	10 / 10
2	$9^{(42)}$ / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
5	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10	10 / 10
$\rho_1 = \rho_2 = 0.99$						
0	10 / $9^{(38)}$	$9^{(32)}$ / 10	10 / 10	10 / 10	10 / 10	10 / 10
1	$9^{(98)}$ / 10	$9^{(42)}$ / $9^{(43)}$	10 / $9^{(8)}$	10 / 10	10 / 10	10 / 10
2	10 / 10	10 / $9^{(44)}$	10 / 10	10 / 10	10 / 10	10 / 10
5	10 / 10	$9^{(60)}$ / 10	10 / 10	10 / 10	10 / 10	10 / 10

In both of the experiments, *Version 1* obtains the optimal solution with the minimum number of replications when the parameters are taken as: α=1, β=1, q_0=0.2 and ρ_1=ρ_2=0.99. In the first experiment, *Version 1* finds the optimal solution with the minimum number of replications for α=1, β=1, q_0=0.2 and ρ_1=ρ_2=0.9. The shaded cells indicate the best set of parameters. Details are given at Chapter 6.

5.6 Experimental Setting for New ACO Approach, Version 2

5.6.1 Number of Ants

The effect of the number of ants on the efficiency of the New ACO Approach *(Version 2)* is given in Figure 5.5. The abscissa represents the total number of ants used in each set of replication and the ordinate represents the number of replications (tours for Barthold problem) required to obtain the optimum.

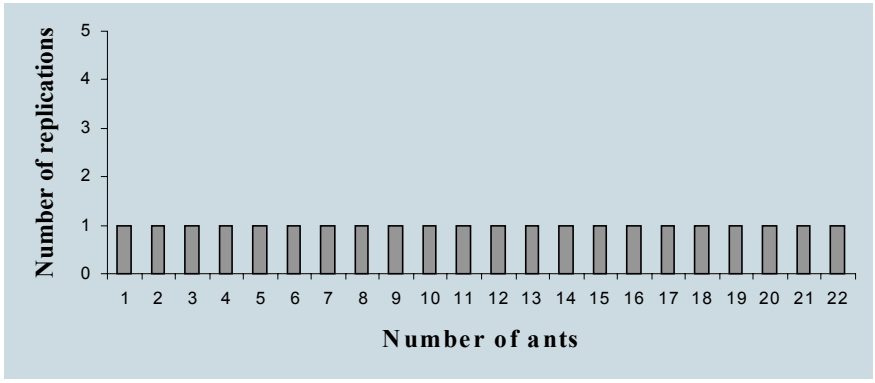


Figure 5.5.a: Number of replications required to obtain the optimum number of stations. Jackson problem with 11 tasks, *C*=10. The experiment has been carried out for (100 replications • 100 ant tours).

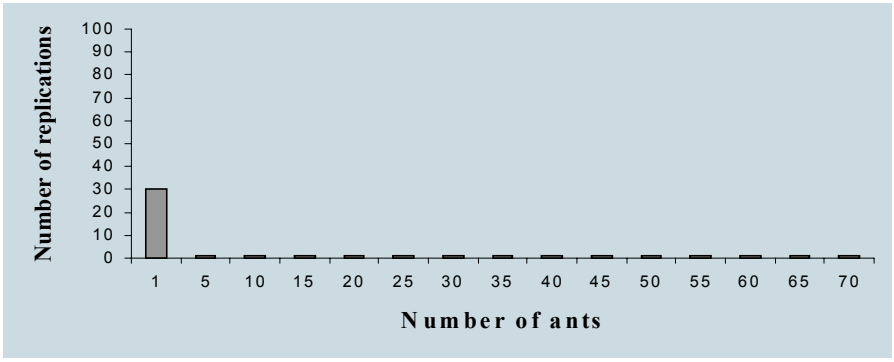


Figure 5.5.b: Number of replications required to reach optimum number of stations. Gunther problem with 35 tasks, *C*=54. The experiment has been carried out for (100 replications • 100 ant tours).

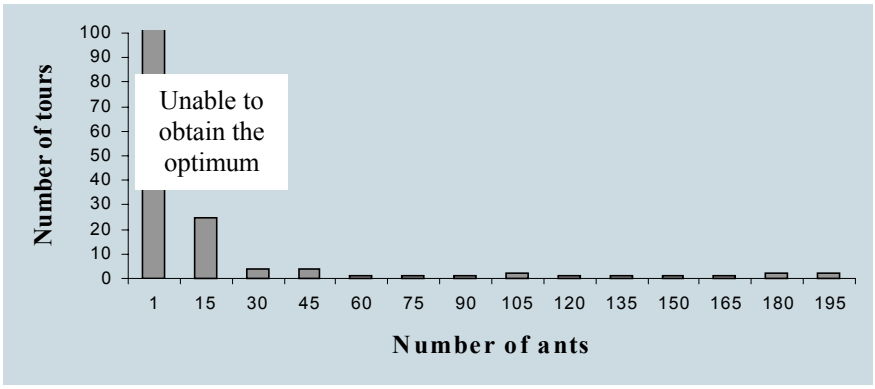


Figure 5.5.c: Number of tours required to reach optimum number of stations. Barthold problem with 148 tasks, *C*=805. The experiment has been carried out for (1 replications • 100 ant tours).

The results indicate that for Jackson problem, *Version 2* finds the optimal station number for any number of ants at the first replication. For Gunther problem, the optimum is obtained with the minimum number of replications when more than 1 ant is used. For Barthold problem, when more than 45 ants are used, *Version 2* finds the optimum with less number of tours. At first tour, the optimum is reached for 60, 75, 90, 120, 135, 150 and 165 ants. After 165 ants, *Version 1* needs more that 1 tour to find the optimum. Therefore it is more reliable to take the number of ants equal to the number of tasks.

5.6.2 Parameters Setting

Version 2 is an extension of *Version 1*. Therefore, the same parameter set is used for *Version 2*. However there are some problem instances that the performance of *Version 2* is better when the parameter q_0 is taken as 0.3 and 0.8. The details are given at Chapter 6.

Chapter 6

Computational Results

In this chapter we present the results of the proposed algorithms on several test problems. Recall that the algorithms under consideration are *Ant System*, *Ant System with Elitist Strategy*, *Ant Colony System*, *Modified Ant Colony System with Random Search*, *New Ant Colony Optimization Approach; Version 1* and *Version 2***.**

Two data sets are used. The first data set (Talbot et al., 1986) considers 64 instances with varying problem sizes ranging from 8 to 111 tasks. The second data set (Scholl, 1993) is relatively new and most complex with 168 instances ranging from 25 to 297 tasks.

We run the proposed algorithms with 190 instances from these data sets and compare the results with those of ULINO (Scholl and Klein, 1999) and the simulated annealing based algorithm of Erel, Sabuncuoglu and Aksu (2001). The results indicate that ULINO finds the optimal solution for most of the instances. In some cases, results are given in the interval. A simple lower bound on the minimal number of stations is equal to $LB = \left\lceil \sum_{i=1}^{n} t_i / C \right\rceil$, where $\lceil x \rceil$ is the smallest integer larger than x (Scholl and Klein, 1999; Erel, Sabuncuoglu and Aksu, 2001). Besides the simple lower bound *LB,* Scholl and Klein (1999) use three additional bound arguments. The upper bound on the minimal number of stations is

proposed to be the required number of stations after all the tasks are assigned to the stations.

Solutions for 15 out of 190 instances are given in intervals with a lower bound and an upper bound. The upper bound and the lower bound of the interval are explained above. SA could not find the optimal solution in 45 out of 187 instances. SA finds optimal solution for only 127 instances. For 15 instances, solutions are given in intervals. ULINO finds the optimal solution for 169 out of 190 instances. For 6 instances, ULINO could not find the optimum, but the result are given in intervals in this case.

In the following (Sections 6.1-6.6), we present the computational results. The results are given in Tables 6.1-6.5. Each table has 3 parts. In the first part, the information about the problem (problem name, number of tasks, cycle time and total task time) is given. In the second part, the results of SA, ULINO, proposed ant algorithm and the optimal solution are reported. For the proposed ant algorithm, the computation time is given in milliseconds. We run the algorithm until (100 replications $\times$ 100 ant tours) have been completed. However, for some problem instances more ant tours are required. Details about these instances are given in Tables 6.4 and 6.5. In the third part of the table, the difference between the results of the ant algorithm, SA and ULINO and the optimal solution are reported. In each table, the *light shaded* cells represent the problem instances that result greater than the optimal and the *dark shaded* cells represents the instances that the result is given in interval.

The computational requirements are not high for any of the proposed algorithms. The algorithms are written in Borland Delphi 6.0. The average computational time for an experiment requires a few seconds on an AMD Athlon XP 2000+, 266 Mhz machine with 256 MB RAM (333MHZ).

6.1 Computational Results for AS

In the proposed AS, we take the number of ants *m,* equal to the number of tasks *n*. The parameters are selected as follows: $\alpha=2$, $\beta=2$, $\rho=0.7$, *initial trail=1, Q=1*. The computational results of Ant System are given in Table 6.1.

Recall that, in 15 instances the number of stations is given in intervals. Referring to Table 6.1, AS could not find the optimal solution in 103 out of 190 instances. In fact, in 97 out of 103 instances AS yields the solution with one station more than the optimal, in 4 out of 103 instances AS finds two stations more than the optimal, and in 2 out of 103 instances AS yields three stations more than the optimal. For the remaining 72 instances AS finds the optimal station number.

In 5 instances AS performs better than SA and finds the optimal solution. However, in 63 instances SA performs better than AS and finds the optimal solution. There is only one instance that none of them can find the optimum. In that instance AS finds 1 station less than SA.

In 6 instances ULINO could not find the optimum and the solutions are given in intervals whereas AS finds the upper bound in 5 of these instances. However, SA finds the optimum in 5 of these instances. In 97 problem instances ULINO performs better than AS and finds the optimum.

We generally observe that AS can solve small and medium size problems (8-32 tasks). However AS displays generally a poor performance for the large size problems (53, 58, 89, 94, 148 and 297 tasks).

Table 6.1: Computational Results of Ant System

	Problem	Task Num	Cycle Time	Total Time	ULINO	SA	AS	Optimal	CPU milisec	Opt-AS	Opt-ULINO	Opt-SA
1	Bowman	8	20	75	4	5	4	***4***	160	0	0	-1
2	Mansoor	11	48	185	4	4	4	***4***	0	0	0	0
3			62		3	3	3	***3***	50	0	0	0
4			94		2	2	2	***2***	0	0	0	0
5	Jackson	11	7	46	7	7	7	***7***	380	0	0	0
6			9		6	6	6	***6***	0	0	0	0
7			10		5	5	5	***5***	0	0	0	0
8			13		4	4	4	***4***	0	0	0	0
9			14		4	4	4	***4***	0	0	0	0
10			21		3	3	3	***3***	0	0	0	0
11	Mitchell	21	14	105	8	8	8	***8***	0	0	0	0
12			15		8	8	8	***8***	0	0	0	0
13			21		5	5	5	***5***	0	0	0	0
14			26		5	-	5	***5***	0	0	0	-
15			35		3	-	3	***3***	0	0	0	-
16			39		3	-	3	***3***	0	0	0	-
17	Roszieg	25	14	125	9	9	9	***9***	1100	0	0	0
18			16		8	8	8	***8***	0	0	0	0
19			18		7	7	7	***7***	0	0	0	0
20			21		6	6	6	***6***	0	0	0	0
21			25		5	5	5	***5***	0	0	0	0
22			32		4	4	4	***4***	50	0	0	0
23	Buxey	29	27	324	13	13	13	***13***	0	0	0	0
24			30		11	11	11	***11***	19010	0	0	0
25			33		10	10	11	***10***	0	-1	0	0
26			36		9	9	10	***9***	0	-1	0	0
27			41		8	8	8	***8***	60	0	0	0
28			47		7	7	7	***7***	0	0	0	0
29			54		6	7	6	***6***	1430	0	0	-1
30	Lutz1	32	1414	14140	11	11	11	***11***	0	0	0	0
31			1572		10	10	10	***10***	0	0	0	0
32			1768		9	9	9	***9***	0	0	0	0
33			2020		8	8	8	***8***	0	0	0	0
34			2357		7	7	7	***7***	50	0	0	0
35			2828		6	6	6	***6***	0	0	0	0
36	Gunther	35	41	483	12	13	13	***12***	50	-1	0	-1
37			44		12	12	12	***12***	0	0	0	0
38			49		10	11	11	***10***	0	-1	0	-1
39			54		9	9	10	***9***	50	-1	0	0
40			61		8	9	8	***8***	9720	0	0	-1
41			69		7	8	8	***7***	0	-1	0	-1
42			81		6	7	6	***6***	0	0	0	-1
43	Hahn	53	2004	14026	8	8	8	***8***	50	0	0	0
44			2338		7	7	7	***7***	50	0	0	0
45			2806		5	6	6	***5***	0	-1	0	-1
46			3507		5	5	5	***5***	0	0	0	0

Table 6.1: (Cont'd)

47			4676		3	4	3	***3***	5930	0	0	-1
48	Warnecke	58	54	1548	30, 31	31	32	***30, 31***	50			
49			56		29	29	30	***29***	8790	-1	0	0
50			58		28	29	29	***28***	0	-1	0	-1
51			60		27	27	28	***27***	0	-1	0	0
52			62		26, 27	27	27	***26, 27***	45970			
53			65		24, 25	25	26	***24, 25***	60			
54			68		23, 24	24	25	***23, 24***	50			
55			71		22, 23	23	24	***22, 23***	0			
56			74		21, 22	22	22	***21, 22***	64200			
57			78		20	21	21	***20***	60	-1	0	-1
58			82		19, 20	20	20	***19, 20***	110			
59			86		18	19	19	***18***	50	-1	0	-1
60			92		17	17	18	***17***	0	-1	0	0
61			97		16	17	17	***16***	0	-1	0	-1
62			104		15	15	16	***15***	0	-1	0	0
63			111		14	14	15	***14***	50	-1	0	0
64	Wee-mag	75	28	1499	63	63	63	***63***	60	0	0	0
65			29		63	63	63	***63***	50	0	0	0
66			30		62	62	62	***62***	170	0	0	0
67			31		62	62	62	***62***	60	0	0	0
68			32		61	61	61	***61***	60	0	0	0
69			33		61	61	61	***61***	60	0	0	0
70			34		61	61	61	***61***	50	0	0	0
71			35		60	60	60	***60***	110	0	0	0
72			36		60	60	60	***60***	50	0	0	0
73			37		60	60	60	***60***	60	0	0	0
74			38		60	60	60	***60***	60	0	0	0
75			39		60	60	60	***60***	60	0	0	0
76			40		60	60	60	***60***	50	0	0	0
77			41		59	59	59	***59***	60	0	0	0
78			42		55	55	55	***55***	50	0	0	0
79			43		50	50	50	***50***	50	0	0	0
80			45		38	38	38	***38***	11090	0	0	0
81			46		34	34	34	***34***	75520	0	0	0
82			47		32, 33	33	33	***32, 33***	60			
83			49		31, 32	32	32	***31, 32***	50			
84			50		31, 32	32	32	***31, 32***	60			
85			52		31	31	31	***31***	110	0	0	0
86			54		31	31	31	***31***	60	0	0	0
87			56		30	30	30	***30***	50	0	0	0
88	Lutz2	89	11	485	45	49	48	***45***	50	-3	0	-4
89			12		41	44	44	***41***	220	-3	0	-3
90			13		38	40	40	***38***	390	-2	0	-2
91			14		35	36	37	***35***	50	-2	0	-1
92			15		33	34	34	***33***	160	-1	0	-1
93			16		31	31	32	***31***	50	-1	0	0
94			17		29	29	30	***29***	110	-1	0	0

Table 6.1: (Cont'd)

95			18		27	28	28	***27***	50	-1	0	-1
96			19		26	26	26	***26***	940	0	0	0
97			20		25	25	25	***25***	60	0	0	0
98			21		24	24	24	***24***	50	0	0	0
99	Lutz3	89	75	1644	22	23	23	***22***	50	-1	0	-1
100			79		21	22	22	***21***	50	-1	0	-1
101			83		20	21	21	***20***	60	-1	0	-1
102			87		19	20	20	***19***	60	-1	0	-1
103			92		18	19	19	***18***	50	-1	0	-1
104			97		17	18	18	***17***	60	-1	0	-1
105			103		16	17	17	***16***	60	-1	0	-1
106			110		15	15	16	***15***	50	-1	0	0
107			118		14	14	15	***14***	60	-1	0	0
108			127		13	14	14	***13***	60	-1	0	-1
109			137		12	13	13	***12***	60	-1	0	-1
110			150		11	12	12	***11***	50	-1	0	-1
111	Mukherje	94	176	4208	24, 25	25	25	***24, 25***	110			
112			183		23	24	24	***23***	60	-1	0	-1
113			192		22	23	23	***22***	50	-1	0	-1
114			201		21	22	22	***21***	50	-1	0	-1
115			211		20	21	21	***20***	110	-1	0	-1
116			222		19	20	20	***19***	110	-1	0	-1
117			234		18	19	19	***18***	110	-1	0	-1
118			248		17	18	18	***17***	60	-1	0	-1
119			263		16	17	17	***16***	110	-1	0	-1
120			281		15	16	16	***15***	110	-1	0	-1
121			301		14	15	15	***14***	110	-1	0	-1
122			324		13	14	14	***13***	110	-1	0	-1
123			351		12	13	13	***12***	110	-1	0	-1
124	Arcus2	111	5755	150399	27	27	27	***27***	110	0	0	0
125			8847		17, 18	18	18	***17, 18***	160			
126			10027		15, 16	16	16	***15, 16***	160			
127			10743		14, 15	15	15	***14, 15***	170			
128			11378		14	14	14	***14***	170	0	0	0
129			17067		9	9	9	***9***	170	0	0	0
130	Barthold	148	403	5634	14	14	15	***14***	440	-1	0	0
131			434		13	13	14	***13***	490	-1	0	0
132			470		12	12	13	***12***	440	-1	0	0
133			513		11	11	11	***11***	61350	0	0	0
134			564		10	10	10	***10***	1590	0	0	0
135			626		9	9	9	***9***	128690	0	0	0
136			705		8	8	8	***8***	440	0	0	0
137			805		7	7	7	***7***	1040	0	0	0
138	Barthol2	148	84	4234	51	51	52	***51***	132650	-1	0	0
139			85		50, 51	50	52	***50***	380	-2		0
140			87		49	49	51	***49***	390	-2	0	0
141			89		48, 49	48	49	***48***	380	-1		0
142			91		47	47	48	***47***	3300	-1	0	0

Table 6.1: (Cont'd)

143			93		46, 47	46	47	***46***	380	-1		0
144			95		45	45	46	***45***	390	-1	0	0
145			97		44, 45	44	45	***44***	3290	-1		0
146			99		43	43	44	***43***	2090	-1	0	0
147			101		42	42	43	***42***	440	-1	0	0
148			104		41	41	42	***41***	380	-1	0	0
149			106		40	40	41	***40***	390	-1	0	0
150			109		39	39	40	***39***	380	-1	0	0
151			112		38	38	39	***38***	440	-1	0	0
152			115		37	37	38	***37***	390	-1	0	0
153			118		36	36	37	***36***	380	-1	0	0
154			121		35	35	36	***35***	440	-1	0	0
155			125		34	34	35	***34***	380	-1	0	0
156			129		33	33	34	***33***	440	-1	0	0
157			133		32	32	33	***32***	390	-1	0	0
158			137		31	31	32	***31***	440	-1	0	0
159			142		30	30	31	***30***	380	-1	0	0
160			146		29	29	30	***29***	440	-1	0	0
161			152		28	28	29	***28***	440	-1	0	0
162			157		27	27	28	***27***	390	-1	0	0
163			163		26	26	27	***26***	430	-1	0	0
164			170		25	25	26	***25***	440	-1	0	0
165	Scholl	297	1394	69655	50, 51	50	51	***50***	720	-1		0
166			1422		49, 50	50	50	***49, 50***	710			
167			1452		48	48	49	***48***	660	-1	0	0
168			1483		47	47	48	***47***	710	-1	0	0
169			1515		46, 47	47	47	***46***	720	-1		-1
170			1548		45	46	46	***45***	710	-1	0	-1
171			1584		44	44	45	***44***	660	-1	0	0
172			1620		43	44	44	***43***	720	-1	0	-1
173			1659		42	43	43	***42***	710	-1	0	-1
174			1699		41	41	42	***41***	710	-1	0	0
175			1742		40	40	41	***40***	720	-1	0	0
176			1787		39	39	40	***39***	710	-1	0	0
177			1834		38	38	39	***38***	720	-1	0	0
178			1883		37	37	38	***37***	710	-1	0	0
179			1935		36	36	37	***36***	710	-1	0	0
180			1991		35	35	36	***35***	720	-1	0	0
181			2049		34	34	35	***34***	710	-1	0	0
182			2111		33	33	34	***33***	720	-1	0	0
183			2177		32	32	33	***32***	710	-1	0	0
184			2247		31	31	32	***31***	710	-1	0	0
185			2322		30	30	31	***30***	720	-1	0	0
186			2402		29	29	30	***29***	710	-1	0	0
187			2488		28	28	29	***28***	720	-1	0	0
188			2580		27	27	28	***27***	710	-1	0	0
189			2680		26	26	27	***26***	710	-1	0	0
190			2787		25	25	26	***25***	720	-1	0	0

6.2 Computational Results for AS_{elite}

For AS_{elite}, we take the number of ants *m,* equal to the number of tasks *n*. The parameters are selected as follows: $\alpha=2$, $\beta=1$, $\rho=0.9$, *initial trail=1, Q=1*. The computational results of AS_{elite} are given in Table 6.2.

Referring to Table 6.2, AS_{elite} could not find the optimal solution in 94 out of 190 problem instances. In fact, in 90 out of 94 instances, AS_{elite} finds the solution with one station more than the optimal and AS_{elite} finds two stations more than the optimal in 4 of 94 instances. For the remaining 81 instances, AS_{elite} finds the optimal station number. Recall that, in 15 instances the number of stations is given in intervals.

In 9 instances AS_{elite} performs better than SA finding the optimal solution. However, in 59 instances SA performs better finding the optimal solution. There are 3 instances that none of them find the optimal. In 2 of these instances AS_{elite} finds 1 station less than SA and in the other instance AS_{elite} finds 2 stations less than SA.

In 6 instances ULINO could not find the optimum where as AS_{elite} finds the upper bound in 5 of these instances. However SA finds optimum in 5 of these instances. In 88 problem instances, ULINO performs better than AS_{elite} and finds the optimum. We also observe that AS_{elite}'s can solve small and medium size problems (8-45 tasks). Also, AS_{elite} performs well for some large size problems (53, some instances of 58 tasks, 111 and 148 tasks). However, its performance is poor for most large size problems (some instances of 58 tasks, 89, 94, 148 and 297 tasks).

In conclusion, AS_{elite} performs slightly better than AS that AS_{elite} finds the optimal solution in 9 instances more than AS .

Table 6.2: Computational Results of AS_{elite}

	Problem	Task Num	Cycle Time	Total Time	ULINO	SA	AS_{elite}	Optimal	CPU milisec	Opt-AS_{elite}	Opt-ULINO	Opt-SA
1	Bowman	8	20	75	4	5	4	***4***	60	0	0	-1
2	Mansoor	11	48	185	4	4	4	***4***	0	0	0	0
3			62		3	3	3	***3***	50	0	0	0
4			94		2	2	2	***2***	0	0	0	0
5	Jackson	11	7	46	7	7	7	***7***	50	0	0	0
6			9		6	6	6	***6***	0	0	0	0
7			10		5	5	5	***5***	0	0	0	0
8			13		4	4	4	***4***	0	0	0	0
9			14		4	4	4	***4***	0	0	0	0
10			21		3	3	3	***3***	0	0	0	0
11	Mitchell	21	14	105	8	8	8	***8***	0	0	0	0
12			15		8	8	8	***8***	0	0	0	0
13			21		5	5	5	***5***	0	0	0	0
14			26		5	-	5	***5***	0	0	0	-
15			35		3	-	3	***3***	0	0	0	-
16			39		3	-	3	***3***	0	0	0	-
17	Roszieg	25	14	125	9	9	9	***9***	3410	0	0	0
18			16		8	8	8	***8***	0	0	0	0
19			18		7	7	7	***7***	220	0	0	0
20			21		6	6	6	***6***	0	0	0	0
21			25		5	5	5	***5***	0	0	0	0
22			32		4	4	4	***4***	0	0	0	0
23	Buxey	29	27	324	13	13	13	***13***	0	0	0	0
24			30		11	11	11	***11***	15320	0	0	0
25			33		10	10	11	***10***	0	-1	0	0
26			36		9	9	10	***9***	0	-1	0	0
27			41		8	8	8	***8***	0	0	0	0
28			47		7	7	7	***7***	0	0	0	0
29			54		6	7	6	***6***	440	0	0	-1
30	Lutz1	32	1414	14140	11	11	11	***11***	0	0	0	0
31			1572		10	10	10	***10***	0	0	0	0
32			1768		9	9	9	***9***	0	0	0	0
33			2020		8	8	8	***8***	0	0	0	0
34			2357		7	7	7	***7***	0	0	0	0
35			2828		6	6	6	***6***	50	0	0	0
36	Gunther	35	41	483	12	13	12	***12***	8790	0	0	-1
37			44		12	12	12	***12***	0	0	0	0
38			49		10	11	10	***10***	40480	0	0	-1
39			54		9	9	9	***9***	3350	0	0	0
40			61		8	9	8	***8***	7470	0	0	-1
41			69		7	8	8	***7***	0	-1	0	-1
42			81		6	7	6	***6***	0	0	0	-1
43	Hahn	53	2004	14026	8	8	8	***8***	60	0	0	0
44			2338		7	7	7	***7***	0	0	0	0
45			2806		5	6	5	***5***	36030	0	0	-1
46			3507		5	5	5	***5***	0	0	0	0

Table 6.2: (Cont'd)

47			4676		3	4	3	***3***	7470	0	0	-1
48	Warnecke	58	54	1548	30, 31	31	31	***30, 31***	229090			
49			56		29	29	30	***29***	4230	-1	0	0
50			58		28	29	28	***28***	218000	0	0	-1
51			60		27	27	28	***27***	110	-1	0	0
52			62		26, 27	27	27	***26, 27***	8460			
53			65		24, 25	25	26	***24, 25***	170			
54			68		23, 24	24	24	***23, 24***	19880			
55			71		22, 23	23	23	***22, 23***	32290			
56			74		21, 22	22	22	***21, 22***	0			
57			78		20	21	21	***20***	160	-1	0	-1
58			82		19, 20	20	20	***19, 20***	110			
59			86		18	19	19	***18***	60	-1	0	-1
60			92		17	17	18	***17***	0	-1	0	0
61			97		16	17	17	***16***	50	-1	0	-1
62			104		15	15	16	***15***	0	-1	0	0
63			111		14	14	15	***14***	60	-1	0	0
64	Wee-mag	75	28	1499	63	63	63	***63***	50	0	0	0
65			29		63	63	63	***63***	60	0	0	0
66			30		62	62	62	***62***	100	0	0	0
67			31		62	62	62	***62***	60	0	0	0
68			32		61	61	61	***61***	50	0	0	0
69			33		61	61	61	***61***	60	0	0	0
70			34		61	61	61	***61***	50	0	0	0
71			35		60	60	60	***60***	60	0	0	0
72			36		60	60	60	***60***	50	0	0	0
73			37		60	60	60	***60***	60	0	0	0
74			38		60	60	60	***60***	110	0	0	0
75			39		60	60	60	***60***	50	0	0	0
76			40		60	60	60	***60***	60	0	0	0
77			41		59	59	59	***59***	50	0	0	0
78			42		55	55	55	***55***	60	0	0	0
79			43		50	50	50	***50***	50	0	0	0
80			45		38	38	38	***38***	60	0	0	0
81			46		34	34	34	***34***	432320	0	0	0
82			47		32, 33	33	33	***32, 33***	50			
83			49		31, 32	32	32	***31, 32***	60			
84			50		31, 32	32	32	***31, 32***	110			
85			52		31	31	31	***31***	50	0	0	0
86			54		31	31	31	***31***	60	0	0	0
87			56		30	30	30	***30***	50	0	0	0
88	Lutz2	89	11	485	45	49	47	***45***	85740	-2	0	-4
89			12		41	44	43	***41***	166260	-2	0	-3
90			13		38	40	39	***38***	81290	-1	0	-2
91			14		35	36	36	***35***	11540	-1	0	-1
92			15		33	34	33	***33***	380	0	0	-1
93			16		31	31	31	***31***	64760	0	0	0
94			17		29	29	29	***29***	119960	0	0	0

Table 6.2: (Cont'd)

95			18		27	28	28	***27***	60	-1	0	-1
96			19		26	26	26	***26***	44920	0	0	0
97			20		25	25	25	***25***	60	0	0	0
98			21		24	24	24	***24***	50	0	0	0
99	Lutz3	89	75	1644	22	23	23	***22***	60	-1	0	-1
100			79		21	22	22	***21***	50	-1	0	-1
101			83		20	21	21	***20***	60	-1	0	-1
102			87		19	20	20	***19***	50	-1	0	-1
103			92		18	19	19	***18***	110	-1	0	-1
104			97		17	18	18	***17***	60	-1	0	-1
105			103		16	17	17	***16***	50	-1	0	-1
106			110		15	15	16	***15***	60	-1	0	0
107			118		14	14	15	***14***	50	-1	0	0
108			127		13	14	14	***13***	60	-1	0	-1
109			137		12	13	13	***12***	50	-1	0	-1
110			150		11	12	12	***11***	110	-1	0	-1
111	Mukherje	94	176	4208	24, 25	25	25	***24, 25***	60			
112			183		23	24	24	***23***	110	-1	0	-1
113			192		22	23	23	***22***	110	-1	0	-1
114			201		21	22	22	***21***	50	-1	0	-1
115			211		20	21	21	***20***	110	-1	0	-1
116			222		19	20	20	***19***	110	-1	0	-1
117			234		18	19	19	***18***	60	-1	0	-1
118			248		17	18	18	***17***	110	-1	0	-1
119			263		16	17	17	***16***	110	-1	0	-1
120			281		15	16	16	***15***	50	-1	0	-1
121			301		14	15	15	***14***	110	-1	0	-1
122			324		13	14	14	***13***	110	-1	0	-1
123			351		12	13	13	***12***	110	-1	0	-1
124	Arcus2	111	5755	150399	27	27	27	***27***	170	0	0	0
125			8847		17, 18	18	18	***17, 18***	110			
126			10027		15, 16	16	16	***15, 16***	160			
127			10743		14, 15	15	15	***14, 15***	110			
128			11378		14	14	14	***14***	170	0	0	0
129			17067		9	9	9	***9***	160	0	0	0
130	Barthold	148	403	5634	14	14	15	***14***	390	-1	0	0
131			434		13	13	14	***13***	60250	-1	0	0
132			470		12	12	12	***12***	56850	0	0	0
133			513		11	11	11	***11***	990	0	0	0
134			564		10	10	10	***10***	440	0	0	0
135			626		9	9	9	***9***	75800	0	0	0
136			705		8	8	8	***8***	440	0	0	0
137			805		7	7	7	***7***	2080	0	0	0
138	Barthol2	148	84	4234	51	51	52	***51***	139900	-1	0	0
139			85		50, 51	50	52	***50***	380	-2		0
140			87		49	49	51	***49***	330	-2	0	0
141			89		48, 49	48	49	***48***	279850	-1		0
142			91		47	47	48	***47***	70520	-1	0	0

Table 6.2: (Cont'd)

143			93		46, 47	46	47	***46***	380	-1		0
144			95		45	45	46	***45***	200320	-1	0	0
145			97		44, 45	44	45	***44***	131650	-1		0
146			99		43	43	44	***43***	3790	-1	0	0
147			101		42	42	43	***42***	6540	-1	0	0
148			104		41	41	42	***41***	380	-1	0	0
149			106		40	40	41	***40***	390	-1	0	0
150			109		39	39	40	***39***	380	-1	0	0
151			112		38	38	39	***38***	390	-1	0	0
152			115		37	37	38	***37***	380	-1	0	0
153			118		36	36	37	***36***	390	-1	0	0
154			121		35	35	36	***35***	380	-1	0	0
155			125		34	34	35	***34***	390	-1	0	0
156			129		33	33	34	***33***	380	-1	0	0
157			133		32	32	33	***32***	380	-1	0	0
158			137		31	31	32	***31***	440	-1	0	0
159			142		30	30	31	***30***	390	-1	0	0
160			146		29	29	30	***29***	380	-1	0	0
161			152		28	28	29	***28***	390	-1	0	0
162			157		27	27	28	***27***	440	-1	0	0
163			163		26	26	27	***26***	380	-1	0	0
164			170		25	25	26	***25***	390	-1	0	0
165	Scholl	297	1394	69655	50, 51	50	51	***50***	710	-1		0
166			1422		49, 50	50	50	***49, 50***	720			
167			1452		48	48	49	***48***	650	-1	0	0
168			1483		47	47	48	***47***	720	-1	0	0
169			1515		46, 47	47	47	***46***	660	-1		-1
170			1548		45	46	46	***45***	710	-1	0	-1
171			1584		44	44	45	***44***	660	-1	0	0
172			1620		43	44	44	***43***	720	-1	0	-1
173			1659		42	43	43	***42***	650	-1	0	-1
174			1699		41	41	42	***41***	720	-1	0	0
175			1742		40	40	41	***40***	660	-1	0	0
176			1787		39	39	40	***39***	710	-1	0	0
177			1834		38	38	39	***38***	720	-1	0	0
178			1883		37	37	38	***37***	650	-1	0	0
179			1935		36	36	37	***36***	720	-1	0	0
180			1991		35	35	36	***35***	710	-1	0	0
181			2049		34	34	35	***34***	660	-1	0	0
182			2111		33	33	34	***33***	720	-1	0	0
183			2177		32	32	33	***32***	710	-1	0	0
184			2247		31	31	32	***31***	660	-1	0	0
185			2322		30	30	31	***30***	710	-1	0	0
186			2402		29	29	30	***29***	720	-1	0	0
187			2488		28	28	29	***28***	710	-1	0	0
188			2580		27	27	28	***27***	660	-1	0	0
189			2680		26	26	27	***26***	710	-1	0	0
190			2787		25	25	26	***25***	720	-1	0	0

6.3 Computational Results for ACS

In ACS, we take the number of ants *m,* equal to the number of tasks *n*. The parameters are selected as follows: $\beta=1$, $\rho_1=\rho_2=0.4$, $q_0=0.2$, $\tau_0=0.0028$, *initial trail=1.* The computational results of ACS are given in Table 6.3.

In 15 instances, the solution is given in intervals. Referring to Table 6.3, ACS yields the optimal solution in 81 problem instances. However, ACS can not find the optimal solution for 94 instances. In fact, in 90 out of 94 instances, ACS finds a solution with one station more than the optimum and two stations more than the optimal for the remaining 4 instances.

In 9 instances, ACS performs better than SA finding the optimal solution. In 58 instances, SA performs better finding the optimal solution. There are 3 instances that none of them find the optimal solution. In fact, in 2 of these instances, ACS finds 1 station less than SA and in the remaining instance ACS finds 2 stations less than SA.

In 6 instances ULINO could not find the optimum whereas ACS finds the upper bound in 5 of these instances. However, SA finds the optimum in 5 of these instances. In 88 problem instances ULINO performs better than ACS.

We also note that ACS performs well up to 53 tasks. It yields the optimal solution for 75 and 111 task problems. Its performance is poor for large size problems (58, 89, 94, 148 and 297 tasks).

In conclusion, the performance of AS_{elite} and ACS are the same and both of them solve 81 problem instances optimally.

Table 6.3: Computational Results of ACS

	Problem	Task Num	Cycle Time	Total Time	ULINO	SA	ACS	Optimal	CPU milisec	Opt-ACS	Opt-ULINO	Opt-SA
1	Bowman	8	20	75	4	5	4	***4***	0	0	0	-1
2	Mansoor	11	48	185	4	4	4	***4***	0	0	0	0
3			62		3	3	3	***3***	0	0	0	0
4			94		2	2	2	***2***	0	0	0	0
5	Jackson	11	7	46	7	7	7	***7***	0	0	0	0
6			9		6	6	6	***6***	0	0	0	0
7			10		5	5	5	***5***	0	0	0	0
8			13		4	4	4	***4***	0	0	0	0
9			14		4	4	4	***4***	0	0	0	0
10			21		3	3	3	***3***	0	0	0	0
11	Mitchell	21	14	105	8	8	8	***8***	0	0	0	0
12			15		8	8	8	***8***	0	0	0	0
13			21		5	5	5	***5***	0	0	0	0
14			26		5	-	5	***5***	0	0	0	-
15			35		3	-	3	***3***	0	0	0	-
16			39		3	-	3	***3***	0	0	0	-
17	Roszieg	25	14	125	9	9	9	***9***	50	0	0	0
18			16		8	8	8	***8***	0	0	0	0
19			18		7	7	7	***7***	0	0	0	0
20			21		6	6	6	***6***	0	0	0	0
21			25		5	5	5	***5***	0	0	0	0
22			32		4	4	4	***4***	0	0	0	0
23	Buxey	29	27	324	13	13	13	***13***	0	0	0	0
24			30		11	11	11	***11***	2030	0	0	0
25			33		10	10	10	***10***	7190	0	0	0
26			36		9	9	10	***9***	50	-1	0	0
27			41		8	8	8	***8***	0	0	0	0
28			47		7	7	7	***7***	0	0	0	0
29			54		6	7	6	***6***	930	0	0	-1
30	Lutz1	32	1414	14140	11	11	11	***11***	60	0	0	0
31			1572		10	10	10	***10***	0	0	0	0
32			1768		9	9	9	***9***	50	0	0	0
33			2020		8	8	8	***8***	0	0	0	0
34			2357		7	7	7	***7***	0	0	0	0
35			2828		6	6	6	***6***	0	0	0	0
36	Gunther	35	41	483	12	13	12	***12***	4280	0	0	-1
37			44		12	12	12	***12***	0	0	0	0
38			49		10	11	10	***10***	1260	0	0	-1
39			54		9	9	9	***9***	870	0	0	0
40			61		8	9	8	***8***	490	0	0	-1
41			69		7	8	7	***7***	19110	0	0	-1
42			81		6	7	6	***6***	0	0	0	-1
43	Hahn	53	2004	14026	8	8	8	***8***	60	0	0	0
44			2338		7	7	7	***7***	0	0	0	0
45			2806		5	6	6	***5***	50	-1	0	-1
46			3507		5	5	5	***5***	0	0	0	0

Table 6.3: (Cont'd)

47			4676		3	4	3	***3***	110	0	0	-1
48	Warnecke	58	54	1548	30, 31	31	31	***30, 31***	3350			
49			56		29	29	30	***29***	110	-1	0	0
50			58		28	29	29	***28***	50	-1	0	-1
51			60		27	27	28	***27***	380	-1	0	0
52			62		26, 27	27	27	***26, 27***	2690			
53			65		24, 25	25	26	***24, 25***	110			
54			68		23, 24	24	24	***23, 24***	15550			
55			71		22, 23	23	23	***22, 23***	7030			
56			74		21, 22	22	22	***21, 22***	15210			
57			78		20	21	21	***20***	60	-1	0	-1
58			82		19, 20	20	20	***19, 20***	0			
59			86		18	19	19	***18***	50	-1	0	-1
60			92		17	17	18	***17***	0	-1	0	0
61			97		16	17	17	***16***	60	-1	0	-1
62			104		15	15	16	***15***	0	-1	0	0
63			111		14	14	15	***14***	50	-1	0	0
64	Wee-mag	75	28	1499	63	63	63	***63***	110	0	0	0
65			29		63	63	63	***63***	50	0	0	0
66			30		62	62	62	***62***	60	0	0	0
67			31		62	62	62	***62***	50	0	0	0
68			32		61	61	61	***61***	110	0	0	0
69			33		61	61	61	***61***	60	0	0	0
70			34		61	61	61	***61***	50	0	0	0
71			35		60	60	60	***60***	110	0	0	0
72			36		60	60	60	***60***	60	0	0	0
73			37		60	60	60	***60***	50	0	0	0
74			38		60	60	60	***60***	60	0	0	0
75			39		60	60	60	***60***	110	0	0	0
76			40		60	60	60	***60***	50	0	0	0
77			41		59	59	59	***59***	60	0	0	0
78			42		55	55	55	***55***	50	0	0	0
79			43		50	50	50	***50***	60	0	0	0
80			45		38	38	38	***38***	9880	0	0	0
81			46		34	34	35	***34***	110	-1	0	0
82			47		32, 33	33	33	***32, 33***	170			
83			49		31, 32	32	32	***31, 32***	110			
84			50		31, 32	32	32	***31, 32***	50			
85			52		31	31	31	***31***	50	0	0	0
86			54		31	31	31	***31***	110	0	0	0
87			56		30	30	30	***30***	60	0	0	0
88	Lutz2	89	11	485	45	49	47	***45***	39380	-2	0	-4
89			12		41	44	43	***41***	19060	-2	0	-3
90			13		38	40	39	***38***	8630	-1	0	-2
91			14		35	36	36	***35***	1210	-1	0	-1
92			15		33	34	33	***33***	28890	0	0	-1
93			16		31	31	31	***31***	9180	0	0	0
94			17		29	29	29	***29***	24380	0	0	0

Table 6.3: (Cont'd)

95			18		27	28	28	***27***	210	-1	0	-1
96			19		26	26	26	***26***	16370	0	0	0
97			20		25	25	25	***25***	610	0	0	0
98			21		24	24	24	***24***	110	0	0	0
99	Lutz3	89	75	1644	22	23	23	***22***	110	-1	0	-1
100			79		21	22	22	***21***	60	-1	0	-1
101			83		20	21	21	***20***	50	-1	0	-1
102			87		19	20	20	***19***	110	-1	0	-1
103			92		18	19	19	***18***	60	-1	0	-1
104			97		17	18	18	***17***	110	-1	0	-1
105			103		16	17	17	***16***	50	-1	0	-1
106			110		15	15	16	***15***	60	-1	0	0
107			118		14	14	15	***14***	110	-1	0	0
108			127		13	14	14	***13***	50	-1	0	-1
109			137		12	13	13	***12***	110	-1	0	-1
110			150		11	12	12	***11***	60	-1	0	-1
111	Mukherje	94	176	4208	24, 25	25	25	***24, 25***	110			
112			183		23	24	24	***23***	50	-1	0	-1
113			192		22	23	23	***22***	110	-1	0	-1
114			201		21	22	22	***21***	110	-1	0	-1
115			211		20	21	21	***20***	110	-1	0	-1
116			222		19	20	20	***19***	110	-1	0	-1
117			234		18	19	19	***18***	110	-1	0	-1
118			248		17	18	18	***17***	110	-1	0	-1
119			263		16	17	17	***16***	110	-1	0	-1
120			281		15	16	16	***15***	110	-1	0	-1
121			301		14	15	15	***14***	110	-1	0	-1
122			324		13	14	14	***13***	110	-1	0	-1
123			351		12	13	13	***12***	110	-1	0	-1
124	Arcus2	111	5755	150399	27	27	27	***27***	440	0	0	0
125			8847		17, 18	18	18	***17, 18***	170			
126			10027		15, 16	16	16	***15, 16***	160			
127			10743		14, 15	15	15	***14, 15***	170			
128			11378		14	14	14	***14***	160	0	0	0
129			17067		9	9	9	***9***	170	0	0	0
130	Barthold	148	403	5634	14	14	15	***14***	500	-1	0	0
131			434		13	13	13	***13***	5220	0	0	0
132			470		12	12	12	***12***	4500	0	0	0
133			513		11	11	11	***11***	3190	0	0	0
134			564		10	10	10	***10***	7910	0	0	0
135			626		9	9	9	***9***	21750	0	0	0
136			705		8	8	8	***8***	1870	0	0	0
137			805		7	7	7	***7***	5100	0	0	0
138	Barthol2	148	84	4234	51	51	52	***51***	2150	-1	0	0
139			85		50, 51	50	52	***50***	430	-2		0
140			87		49	49	51	***49***	440	-2	0	0
141			89		48, 49	48	49	***48***	6920	-1		0
142			91		47	47	48	***47***	1210	-1	0	0

Table 6.3: (Cont'd)

143			93		46, 47	46	47	***46***	500	-1		0
144			95		45	45	46	***45***	440	-1	0	0
145			97		44, 45	44	45	***44***	31580	-1		0
146			99		43	43	44	***43***	10710	-1	0	0
147			101		42	42	43	***42***	23340	-1	0	0
148			104		41	41	42	***41***	500	-1	0	0
149			106		40	40	41	***40***	1970	-1	0	0
150			109		39	39	40	***39***	440	-1	0	0
151			112		38	38	39	***38***	440	-1	0	0
152			115		37	37	38	***37***	500	-1	0	0
153			118		36	36	37	***36***	440	-1	0	0
154			121		35	35	36	***35***	490	-1	0	0
155			125		34	34	35	***34***	440	-1	0	0
156			129		33	33	34	***33***	500	-1	0	0
157			133		32	32	33	***32***	430	-1	0	0
158			137		31	31	32	***31***	500	-1	0	0
159			142		30	30	31	***30***	440	-1	0	0
160			146		29	29	30	***29***	490	-1	0	0
161			152		28	28	29	***28***	440	-1	0	0
162			157		27	27	28	***27***	500	-1	0	0
163			163		26	26	27	***26***	490	-1	0	0
164			170		25	25	26	***25***	440	-1	0	0
165	Scholl	297	1394	69655	50, 51	50	51	***50***	270	-1		0
166			1422		49, 50	50	50	***49, 50***	110			
167			1452		48	48	49	***48***	110	-1	0	0
168			1483		47	47	48	***47***	60	-1	0	0
169			1515		46, 47	47	47	***46***	110	-1		-1
170			1548		45	46	46	***45***	110	-1	0	-1
171			1584		44	44	45	***44***	110	-1	0	0
172			1620		43	44	44	***43***	110	-1	0	-1
173			1659		42	43	43	***42***	50	-1	0	-1
174			1699		41	41	42	***41***	110	-1	0	0
175			1742		40	40	41	***40***	110	-1	0	0
176			1787		39	39	40	***39***	110	-1	0	0
177			1834		38	38	39	***38***	50	-1	0	0
178			1883		37	37	38	***37***	110	-1	0	0
179			1935		36	36	37	***36***	110	-1	0	0
180			1991		35	35	36	***35***	110	-1	0	0
181			2049		34	34	35	***34***	110	-1	0	0
182			2111		33	33	34	***33***	60	-1	0	0
183			2177		32	32	33	***32***	110	-1	0	0
184			2247		31	31	32	***31***	110	-1	0	0
185			2322		30	30	31	***30***	110	-1	0	0
186			2402		29	29	30	***29***	50	-1	0	0
187			2488		28	28	29	***28***	110	-1	0	0
188			2580		27	27	28	***27***	110	-1	0	0
189			2680		26	26	27	***26***	110	-1	0	0
190			2787		25	25	26	***25***	110	-1	0	0

6.4 Computational Results for Modified ACS with Random Search

Modified ACS with random search is a two-phase approach and it requires excessive amount of time to collect data for the *T2* matrix. For that reason, we test this approach with some medium and large size problems that AS, AS_{elite} and ACS could not yield the optimal solution. These problems are Buxey with 29 tasks (*C*=36), Gunther with 35 tasks (C=69), Warnecke with 58 tasks (*C*=60, *C*=78, *C*=82, *C*=86, *C*=92, *C*=97, *C*=104, *C*=111), Lutz2 with 89 tasks (*C*=11, *C*=12, *C*=13, *C*=14, *C*=17).

Modified ACS finds the optimal allocation for only three of these fifteen problem instances, Buxey (*C*=36), Gunther (*C*=69), and Warnecke (*C*=60, *C*=111). However, it is an important development that for the first time we find the optimal allocation for these problems.

Our observations on the structure of the *T2* matrix led us to develop the New ACO Approach (*Version 1* and *Version 2*).

6.5 Computational Results for New ACO Approach, Version 1

In the New ACO Approach, *Version 1* we take the number of ants *m,* equal to the number of tasks *n*. The parameters are selected as follows: $\alpha=1$, $\beta=1$, $q_0=0.2$, $\rho_1=\rho_2=0.9$, $\tau_0=1$, *initial trail=1,* $Q=1$. For some problems ρ_1 and ρ_2 are taken as 0.99; q_0 is taken as 0.3. The computational results of *Version 1* are given in Table 6.4.

Referring to Table 6.4, the performance of *Version 1* is highly satisfying when compared with the previous algorithms. In 108 out of 190 instances, *Version 1* finds the optimum solution. Recall that, in 15 instances optimal solutions are given in intervals. In the remaining 67 instances *Version 1* can not find the optimum.

In 25 instances *Version 1* performs better than SA finding the optimal solution. However, in 47 instances SA performs better than *Version 1* finding the optimal solution. Generally in 16 out of 18 problems the performance of *Version 1* is better. Somehow, only in the two largest problems SA beats *Version 1*.

There are 2 instances that none of them can find the optimal solution. In fact, in 1 of these instances *Version 1* finds 3 stations less than SA and in the other instance *Version 1* finds 2 stations less than SA. Also, there is one instance that *Version 1* finds the optimal solution with 2 stations less than SA.

In 6 instances ULINO could not find the optimum whereas *Version1* finds the upper bound in 5 of these instances. However, SA finds the optimal solution in 5 of these instances. In 61 problem instances ULINO performs better than *Version 1* and finds the optimal solution.

For some problem instances, we observe that *Version 1* can find the optimal solution with the parameters different from the parameters stated in

Section 5.5.2. Some problems need more ant tours to find the optimal solution. Also for some problems when the parameter q_0 is taken as 0.3, *Version 1* obtains better results than when q_0=0.2. Also setting the parameters ρ_1 and ρ_2 equal to 0.99 yields better results instead of taking ρ_1 and ρ_2 equal to 0.9. These observations are also pointed out in Table 6.4.

Table 6.4: Computational Results of the New ACO Approach, *Version 1.*

		Problem	Task Num	Cycle Time	Total Time	ULINO	SA	Ver. 1	Optimal	CPU milisec	Opt-Ver. 1	Opt-ULINO	Opt-SA
	1	Bowman	8	20	75	4	5	4	***4***	0	0	0	-1
	2	Mansoor	11	48	185	4	4	4	***4***	0	0	0	0
	3			62		3	3	3	***3***	0	0	0	0
	4			94		2	2	2	***2***	0	0	0	0
	5	Jackson	11	7	46	7	7	7	***7***	0	0	0	0
	6			9		6	6	6	***6***	0	0	0	0
	7			10		5	5	5	***5***	0	0	0	0
	8			13		4	4	4	***4***	0	0	0	0
	9			14		4	4	4	***4***	0	0	0	0
	10			21		3	3	3	***3***	0	0	0	0
	11	Mitchell	21	14	105	8	8	8	***8***	0	0	0	0
	12			15		8	8	8	***8***	0	0	0	0
	13			21		5	5	5	***5***	0	0	0	0
	14			26		5	-	5	***5***	0	0	0	-
	15			35		3	-	3	***3***	0	0	0	-
	16			39		3	-	3	***3***	0	0	0	-
	17	Roszieg	25	14	125	9	9	9	***9***	0	0	0	0
	18			16		8	8	8	***8***	0	0	0	0
	19			18		7	7	7	***7***	0	0	0	0
	20			21		6	6	6	***6***	0	0	0	0
	21			25		5	5	5	***5***	0	0	0	0
	22			32		4	4	4	***4***	0	0	0	0
	23	Buxey	29	27	324	13	13	13	***13***	0	0	0	0
	24			30		11	11	11	***11***	50	0	0	0
	25			33		10	10	10	***10***	60	0	0	0
	26			36		9	9	9	***9***	9010	0	0	0
	27			41		8	8	8	***8***	0	0	0	0
	28			47		7	7	7	***7***	0	0	0	0
	29			54		6	7	6	***6***	0	0	0	-1
	30	Lutz1	32	1414	14140	11	11	11	***11***	0	0	0	0
	31			1572		10	10	10	***10***	0	0	0	0
	32			1768		9	9	9	***9***	0	0	0	0
	33			2020		8	8	8	***8***	0	0	0	0
	34			2357		7	7	7	***7***	60	0	0	0
	35			2828		6	6	6	***6***	0	0	0	0
	36	Gunther	35	41	483	12	13	12	***12***	60	0	0	-1
	37			44		12	12	12	***12***	0	0	0	0
	38			49		10	11	10	***10***	330	0	0	-1
	39			54		9	9	9	***9***	50	0	0	0
	40			61		8	9	8	***8***	0	0	0	-1
	41			69		7	8	7	***7***	0	0	0	-1
	42			81		6	7	6	***6***	0	0	0	-1
	43	Hahn	53	2004	14026	8	8	8	***8***	0	0	0	0
	44			2338		7	7	7	***7***	0	0	0	0
	45			2806		5	6	5	***5***	30810	0	0	-1
	46			3507		5	5	5	***5***	0	0	0	0

Table 6.4: (Cont'd)

	47			4676		3	4	3	***3***	710	0	0	-1
	48	Warnecke	58	54	1548	30, 31	31	31	***30, 31***	110			
	49			56		29	29	29	***29***	4940	0	0	0
	50			58		28	29	28	***28***	5660	0	0	-1
	51			60		27	27	27	***27***	440	0	0	0
	52			62		26, 27	27	27	***26, 27***	60			
	53			65		24, 25	25	25	***24, 25***	5600			
	54			68		23, 24	24	24	***23, 24***	0			
	55			71		22, 23	23	23	***22, 23***	0			
	56			74		21, 22	22	22	***21, 22***	50			
	57			78		20	21	21	***20***	0	-1	0	-1
	58			82		19, 20	20	20	***19, 20***	0			
	59			86		18	19	19	***18***	0	-1	0	-1
(1)	60			92		17	17	17	***17***	6760	0	0	0
	61			97		16	17	17	***16***	60	-1	0	-1
(1)	62			104		15	15	15	***15***	15110	0	0	0
	63			111		14	14	15	***14***	60	-1	0	0
	64	Wee-mag	75	28	1499	63	63	63	***63***	60	0	0	0
	65			29		63	63	63	***63***	50	0	0	0
	66			30		62	62	62	***62***	0	0	0	0
	67			31		62	62	62	***62***	60	0	0	0
	68			32		61	61	61	***61***	50	0	0	0
	69			33		61	61	61	***61***	0	0	0	0
	70			34		61	61	61	***61***	60	0	0	0
	71			35		60	60	60	***60***	50	0	0	0
	72			36		60	60	60	***60***	0	0	0	0
	73			37		60	60	60	***60***	60	0	0	0
	74			38		60	60	60	***60***	50	0	0	0
	75			39		60	60	60	***60***	60	0	0	0
	76			40		60	60	60	***60***	0	0	0	0
	77			41		59	59	59	***59***	50	0	0	0
	78			42		55	55	55	***55***	60	0	0	0
	79			43		50	50	50	***50***	0	0	0	0
	80			45		38	38	38	***38***	160	0	0	0
	81			46		34	34	34	***34***	220	0	0	0
	82			47		32, 33	33	33	***32, 33***	50			
	83			49		31, 32	32	32	***31, 32***	60			
	84			50		31, 32	32	32	***31, 32***	0			
	85			52		31	31	31	***31***	50	0	0	0
	86			54		31	31	31	***31***	60	0	0	0
	87			56		30	30	30	***30***	50	0	0	0
	88	Lutz2	89	11	485	45	49	46	***45***	6590	-1	0	-4
	89			12		41	44	42	***41***	3020	-1	0	-3
(2)	90			13		38	40	38	***38***	165440	0	0	-2
	91			14		35	36	36	***35***	720	-1	0	-1

(1) For these instances the algorithm is run until 1000 ant tours have been completed. In these experiments q_o is taken as 0.3.
(2) For these instances the algorithm is run until 250 ant tours have been completed.

Table 6.4: (Cont'd)

	92			15		33	34	33	***33***	170	0	0	-1
	93			16		31	31	31	***31***	720	0	0	0
	94			17		29	29	29	***29***	50	0	0	0
(2)	95			18		27	28	27	***27***	166370	0	0	-1
	96			19		26	26	26	***26***	50	0	0	0
	97			20		25	25	25	***25***	60	0	0	0
	98			21		24	24	24	***24***	50	0	0	0
	99	Lutz3	89	75	1644	22	23	23	***22***	60	-1	0	-1
	100			79		21	22	22	***21***	50	-1	0	-1
(3)	101			83		20	21	20	***20***	12250	0	0	-1
(3)	102			87		19	20	19	***19***	63110	0	0	-1
(3)	103			92		18	19	18	***18***	770	0	0	-1
(3)	104			97		17	18	17	***17***	159120	0	0	-1
(3)	105			103		16	17	16	***16***	69590	0	0	-1
(3)	106			110		15	15	15	***15***	20320	0	0	0
(3)	107			118		14	14	14	***14***	1210	0	0	0
(3)	108			127		13	14	13	***13***	3400	0	0	-1
(3)	109			137		12	13	12	***12***	55690	0	0	-1
(3)	110			150		11	12	11	***11***	170	0	0	-1
	111	Mukherje	94	176	4208	24, 25	25	25	***24, 25***	110			
	112			183		23	24	24	***23***	110	-1	0	-1
	113			192		22	23	23	***22***	110	-1	0	-1
	114			201		21	22	22	***21***	60	-1	0	-1
	115			211		20	21	21	***20***	110	-1	0	-1
	116			222		19	20	20	***19***	110	-1	0	-1
	117			234		18	19	19	***18***	50	-1	0	-1
	118			248		17	18	18	***17***	110	-1	0	-1
	119			263		16	17	17	***16***	110	-1	0	-1
(4)	120			281		15	16	15	***15***	497510	0	0	-1
(4)	121			301		14	15	14	***14***	290450	0	0	-1
(4)	122			324		13	14	13	***13***	492580	0	0	-1
(4)	123			351		12	13	12	***12***	35200	0	0	-1
	124	Arcus2	111	5755	150399	27	27	27	***27***	170	0	0	0
	125			8847		17, 18	18	18	***17, 18***	160			
	126			10027		15, 16	16	16	***15, 16***	110			
	127			10743		14, 15	15	15	***14, 15***	160			
	128			11378		14	14	14	***14***	160	0	0	0
	129			17067		9	9	9	***9***	110	0	0	0
	130	Barthold	148	403	5634	14	14	14	***14***	2190	0	0	0
	131			434		13	13	13	***13***	660	0	0	0
	132			470		12	12	12	***12***	1760	0	0	0
	133			513		11	11	11	***11***	270	0	0	0
	134			564		10	10	10	***10***	330	0	0	0
	135			626		9	9	9	***9***	48610	0	0	0

(3) For these instances the algorithm is run until 250 ant tours have been completed. In these experiments q_o is taken as 0.3.
(4) For these instances the algorithm is run until 1000 ant tours have been completed. In these experiments q_o is taken as 0.3; ρ_1 and ρ_2 are taken as 0.99.

Table 6.4: (Cont'd)

	136			705		8	8	8	***8***	330	0	0	0
	137			805		7	7	7	***7***	280	0	0	0
	138	Barthol2	148	84	4234	51	51	52	***51***	280	-1	0	0
	139			85		50, 51	50	51	***50***	14550	-1		0
	140			87		49	49	50	***49***	330	-1	0	0
	141			89		48, 49	48	49	***48***	280	-1		0
	142			91		47	47	48	***47***	270	-1	0	0
	143			93		46, 47	46	47	***46***	280	-1		0
	144			95		45	45	46	***45***	330	-1	0	0
	145			97		44, 45	44	45	***44***	270	-1		0
	146			99		43	43	44	***43***	270	-1	0	0
	147			101		42	42	43	***42***	330	-1	0	0
	148			104		41	41	42	***41***	280	-1	0	0
	149			106		40	40	41	***40***	270	-1	0	0
	150			109		39	39	40	***39***	330	-1	0	0
	151			112		38	38	39	***38***	280	-1	0	0
	152			115		37	37	38	***37***	270	-1	0	0
	153			118		36	36	37	***36***	330	-1	0	0
	154			121		35	35	36	***35***	280	-1	0	0
	155			125		34	34	35	***34***	270	-1	0	0
	156			129		33	33	34	***33***	330	-1	0	0
	157			133		32	32	33	***32***	270	-1	0	0
	158			137		31	31	32	***31***	280	-1	0	0
	159			142		30	30	31	***30***	330	-1	0	0
	160			146		29	29	30	***29***	270	-1	0	0
	161			152		28	28	28	***28***	118420	0	0	0
	162			157		27	27	28	***27***	280	-1	0	0
	163			163		26	26	27	***26***	330	-1	0	0
	164			170		25	25	25	***25***	163900	0	0	0
	165	Scholl	297	1394	69655	50, 51	50	51	***50***	720	-1		0
	166			1422		49, 50	50	50	***49, 50***	710			
	167			1452		48	48	49	***48***	720	-1	0	0
	168			1483		47	47	48	***47***	710	-1	0	0
	169			1515		46, 47	47	47	***46***	710	-1		-1
	170			1548		45	46	46	***45***	720	-1	0	-1
	171			1584		44	44	45	***44***	710	-1	0	0
	172			1620		43	44	44	***43***	720	-1	0	-1
	173			1659		42	43	43	***42***	710	-1	0	-1
	174			1699		41	41	42	***41***	710	-1	0	0
	175			1742		40	40	41	***40***	720	-1	0	0
	176			1787		39	39	40	***39***	710	-1	0	0
	177			1834		38	38	39	***38***	720	-1	0	0
	178			1883		37	37	38	***37***	710	-1	0	0
	179			1935		36	36	37	***36***	720	-1	0	0
	180			1991		35	35	36	***35***	710	-1	0	0
	181			2049		34	34	35	***34***	710	-1	0	0
	182			2111		33	33	34	***33***	660	-1	0	0
	183			2177		32	32	33	***32***	720	-1	0	0

Table 6.4: (Cont'd)

	184			2247		31	31	32	***31***	710	-1	0	0
	185			2322		30	30	31	***30***	710	-1	0	0
	186			2402		29	29	30	***29***	720	-1	0	0
	187			2488		28	28	29	***28***	710	-1	0	0
	188			2580		27	27	28	***27***	720	-1	0	0
	189			2680		26	26	27	***26***	710	-1	0	0
	190			2787		25	25	26	***25***	710	-1	0	0

6.6 Computational Results for New ACO Approach, Version 2

In the New ACO Approach, *Version 2* we take the number of ants *m,* equal to one fourth of the number of tasks (*n/4)*. *Version 2* is an extension of *Version 1*. Therefore, the same parameter set is used for *Version 2*. However, *Version 2* can not find optimal solution in most of the problem instances with the following parameters: $\alpha=1$, $\beta=1$, $q_0=0.2$, $\rho_1=\rho_2=0.9$, $\tau_0=1$, *initial trail=1*, $Q=1$. Therefore, for *Version 2* the new parameters are selected as follows: $\alpha=1$, $\beta=1$, $q_0=0.8$, $\rho_1=\rho_2=0.99$, $\tau_0=1$, *initial trail=1*, $Q_1=1$ and $Q_2=100$. We run the algorithm, *Version 2* until 250 ant tours have been completed. The computational results of *Version 2* are given in Table 6.5.

Referring to Table 6.5, in most of the problem instances the performance of *Version 2* is much better than the previous algorithms. In 144 out of 190 instances *Version 2* finds the optimal solution. Only in 31 instances *Version 2* performs poor.

Version 2 performs better than SA in 39 instances finding the optimal solution. In fact, there are three instances that SA finds 4 stations more, 3 stations more and 2 stations more in each. In 25 instances SA performs better than *Version 2* finding the optimal solution.

In 6 instances ULINO could not find the optimum whereas *Version 2* finds the upper bound in 3 of these 6 instances and the optimum in 3 out of these 6 instances. In 28 instances ULINO performs better than *Version 2*.

We also note that, one advantage of *Version 2* is that it can find the optimal solution with a fewer number of ants. Recall that, in 190 problem instances number of ants *m,* is taken to be equal to one fourth of the number of task (*n*/4), whereas we take the number of ants equal to the number of tasks in the

previous algorithms. This fact indicates that *Version 2* is powerful and the trail update mechanism can be effective when the number of ants is equal to only *n/4*.

Table 6.5: Computational Results of the New ACO Approach, *Version 2.*

		Problem	Task Num	Cycle Time	Total Time	ULINO	SA	Ver. 2	Optimal	CPU milisec	Opt-Ver. 2	Opt-ULINO	Opt-SA
	1	Bowman	8	20	75	4	5	4	***4***	110	0	0	-1
	2	Mansoor	11	48	185	4	4	4	***4***	0	0	0	0
	3			62		3	3	3	***3***	0	0	0	0
	4			94		2	2	2	***2***	0	0	0	0
	5	Jackson	11	7	46	7	7	7	***7***	110	0	0	0
	6			9		6	6	6	***6***	0	0	0	0
	7			10		5	5	5	***5***	0	0	0	0
	8			13		4	4	4	***4***	0	0	0	0
	9			14		4	4	4	***4***	0	0	0	0
	10			21		3	3	3	***3***	0	0	0	0
	11	Mitchell	21	14	105	8	8	8	***8***	0	0	0	0
	12			15		8	8	8	***8***	0	0	0	0
	13			21		5	5	5	***5***	60	0	0	0
	14			26		5	-	5	***5***	0	0	0	-
	15			35		3	-	3	***3***	0	0	0	-
	16			39		3	-	3	***3***	0	0	0	-
	17	Roszieg	25	14	125	9	9	9	***9***	60	0	0	0
	18			16		8	8	8	***8***	0	0	0	0
	19			18		7	7	7	***7***	0	0	0	0
	20			21		6	6	6	***6***	0	0	0	0
	21			25		5	5	5	***5***	0	0	0	0
	22			32		4	4	4	***4***	0	0	0	0
	23	Buxey	29	27	324	13	13	13	***13***	0	0	0	0
	24			30		11	11	11	***11***	380	0	0	0
	25			33		10	10	10	***10***	0	0	0	0
	26			36		9	9	9	***9***	990	0	0	0
(1)	27			41		8	8	8	***8***	0	0	0	0
	28			47		7	7	7	***7***	0	0	0	0
	29			54		6	7	6	***6***	440	0	0	-1
	30	Lutz1	32	1414	14140	11	11	11	***11***	0	0	0	0
	31			1572		10	10	10	***10***	0	0	0	0
	32			1768		9	9	9	***9***	0	0	0	0
	33			2020		8	8	8	***8***	0	0	0	0
	34			2357		7	7	7	***7***	0	0	0	0
	35			2828		6	6	6	***6***	0	0	0	0
	36	Gunther	35	41	483	12	13	12	***12***	440	0	0	-1
	37			44		12	12	12	***12***	0	0	0	0
	38			49		10	11	10	***10***	5990	0	0	-1
	39			54		9	9	9	***9***	0	0	0	0
	40			61		8	9	8	***8***	0	0	0	-1
	41			69		7	8	7	***7***	0	0	0	-1
	42			81		6	7	6	***6***	0	0	0	-1
	43	Hahn	53	2004	14026	8	8	8	***8***	0	0	0	0
	44			2338		7	7	7	***7***	0	0	0	0

(1) In these problems q_0 is taken as 0.3.

Table 6.5: (Cont'd)

(1)	45			2806		5	6	5	***5***	12090	0	0	-1
	46			3507		5	5	5	***5***	0	0	0	0
	47			4676		3	4	3	***3***	2310	0	0	-1
	48	Warnecke	58	54	1548	30, 31	31	31	***30, 31***	220			
	49			56		29	29	29	***29***	18350	0	0	0
	50			58		28	29	28	***28***	5220	0	0	-1
	51			60		27	27	27	***27***	7630	0	0	0
	52			62		26, 27	27	27	***26, 27***	0			
	53			65		24, 25	25	25	***24, 25***	6210			
	54			68		23, 24	24	24	***23, 24***	0			
	55			71		22, 23	23	23	***22, 23***	0			
	56			74		21, 22	22	22	***21, 22***	0			
(1)	57			78		20	21	20	***20***	80520	0	0	-1
	58			82		19, 20	20	20	***19, 20***	0			
	59			86		18	19	19	***18***	0	-1	0	-1
	60			92		17	17	17	***17***	2420	0	0	0
	61			97		16	17	16	***16***	1430	0	0	-1
	62			104		15	15	15	***15***	440	0	0	0
	63			111		14	14	14	***14***	3570	0	0	0
	64	Wee-mag	75	28	1499	63	63	63	***63***	50	0	0	0
	65			29		63	63	63	***63***	0	0	0	0
	66			30		62	62	62	***62***	110	0	0	0
	67			31		62	62	62	***62***	60	0	0	0
	68			32		61	61	61	***61***	0	0	0	0
	69			33		61	61	61	***61***	0	0	0	0
	70			34		61	61	61	***61***	0	0	0	0
	71			35		60	60	60	***60***	0	0	0	0
	72			36		60	60	60	***60***	50	0	0	0
	73			37		60	60	60	***60***	0	0	0	0
	74			38		60	60	60	***60***	0	0	0	0
	75			39		60	60	60	***60***	0	0	0	0
	76			40		60	60	60	***60***	0	0	0	0
	77			41		59	59	59	***59***	60	0	0	0
	78			42		55	55	55	***55***	0	0	0	0
	79			43		50	50	50	***50***	0	0	0	0
	80			45		38	38	38	***38***	0	0	0	0
	81			46		34	34	34	***34***	0	0	0	0
	82			47		32, 33	33	32	***32, 33***	82770			
	83			49		31, 32	32	32	***31, 32***	0			
	84			50		31, 32	32	32	***31, 32***	50			
	85			52		31	31	31	***31***	60	0	0	0
	86			54		31	31	31	***31***	0	0	0	0
	87			56		30	30	30	***30***	0	0	0	0
	88	Lutz2	89	11	485	45	49	45	***45***	47010	0	0	-4
	89			12		41	44	41	***41***	94970	0	0	-3
	90			13		38	40	38	***38***	52120	0	0	-2
	91			14		35	36	35	***35***	7140	0	0	-1
	92			15		33	34	33	***33***	0	0	0	-1

Table 6.5: (Cont'd)

	93			16		31	31	31	***31***	170	0	0	0
	94			17		29	29	29	***29***	0	0	0	0
	95			18		27	28	27	***27***	880	0	0	-1
	96			19		26	26	26	***26***	50	0	0	0
	97			20		25	25	25	***25***	0	0	0	0
	98			21		24	24	24	***24***	0	0	0	0
	99	Lutz3	89	75	1644	22	23	22	***22***	96620	0	0	-1
	100			79		21	22	21	***21***	32680	0	0	-1
	101			83		20	21	20	***20***	0	0	0	-1
	102			87		19	20	19	***19***	930	0	0	-1
	103			92		18	19	18	***18***	50	0	0	-1
	104			97		17	18	17	***17***	57240	0	0	-1
	105			103		16	17	16	***16***	3240	0	0	-1
	106			110		15	15	15	***15***	2030	0	0	0
	107			118		14	14	14	***14***	770	0	0	0
	108			127		13	14	13	***13***	160	0	0	-1
	109			137		12	13	12	***12***	8080	0	0	-1
	110			150		11	12	11	***11***	50	0	0	-1
	111	Mukherje	94	176	4208	24, 25	25	24	***24, 25***	24390			
	112			183		23	24	24	***23***	0	-1	0	-1
	113			192		22	23	22	***22***	2200	0	0	-1
	114			201		21	22	21	***21***	8730	0	0	-1
	115			211		20	21	20	***20***	930	0	0	-1
	116			222		19	20	19	***19***	19610	0	0	-1
	117			234		18	19	18	***18***	32960	0	0	-1
	118			248		17	18	17	***17***	20760	0	0	-1
	119			263		16	17	16	***16***	32680	0	0	-1
	120			281		15	16	15	***15***	3020	0	0	-1
	121			301		14	15	14	***14***	1040	0	0	-1
	122			324		13	14	13	***13***	8190	0	0	-1
	123			351		12	13	12	***12***	270	0	0	-1
	124	Arcus2	111	5755	150399	27	27	27	***27***	60	0	0	0
	125			8847		17, 18	18	18	***17, 18***	0			
	126			10027		15, 16	16	16	***15, 16***	50			
	127			10743		14, 15	15	15	***14, 15***	60			
	128			11378		14	14	14	***14***	0	0	0	0
	129			17067		9	9	9	***9***	50	0	0	0
	130	Barthold	148	403	5634	14	14	14	***14***	550	0	0	0
	131			434		13	13	13	***13***	710	0	0	0
	132			470		12	12	12	***12***	60	0	0	0
	133			513		11	11	11	***11***	50	0	0	0
	134			564		10	10	10	***10***	110	0	0	0
	135			626		9	9	9	***9***	2150	0	0	0
	136			705		8	8	8	***8***	50	0	0	0
	137			805		7	7	7	***7***	110	0	0	0
	138	Barthol2	148	84	4234	51	51	51	***51***	23130	0	0	0
	139			85		50, 51	50	51	***50***	2030	-1		0
	140			87		49	49	50	***49***	60	-1	0	0

Table 6.5: (Cont'd)

	141			89		48, 49	48	48	***48***	26750	0		0
	142			91		47	47	47	***47***	127320	0	0	0
	143			93		46, 47	46	46	***46***	10720	0		0
	144			95		45	45	45	***45***	15660	0	0	0
	145			97		44, 45	44	44	***44***	19170	0		0
	146			99		43	43	43	***43***	13840	0	0	0
	147			101		42	42	43	***42***	50	-1	0	0
	148			104		41	41	41	***41***	4060	0	0	0
	149			106		40	40	40	***40***	43280	0	0	0
	150			109		39	39	39	***39***	9780	0	0	0
	151			112		38	38	38	***38***	15810	0	0	0
	152			115		37	37	37	***37***	5110	0	0	0
	153			118		36	36	36	***36***	68940	0	0	0
	154			121		35	35	36	***35***	110	-1	0	0
	155			125		34	34	34	***34***	6750	0	0	0
	156			129		33	33	33	***33***	17740	0	0	0
	157			133		32	32	32	***32***	4500	0	0	0
	158			137		31	31	31	***31***	21860	0	0	0
	159			142		30	30	30	***30***	7200	0	0	0
	160			146		29	29	29	***29***	99250	0	0	0
	161			152		28	28	28	***28***	6980	0	0	0
	162			157		27	27	27	***27***	30040	0	0	0
	163			163		26	26	26	***26***	4060	0	0	0
	164			170		25	25	25	***25***	6860	0	0	0
	165	Scholl	297	1394	69655	50, 51	50	51	***50***	770	-1		0
	166			1422		49, 50	50	50	***49, 50***	770			
	167			1452		48	48	49	***48***	710	-1	0	0
	168			1483		47	47	48	***47***	720	-1	0	0
	169			1515		46, 47	47	47	***46***	770	-1		-1
	170			1548		45	46	46	***45***	710	-1	0	-1
	171			1584		44	44	45	***44***	710	-1	0	0
	172			1620		43	44	44	***43***	770	-1	0	-1
	173			1659		42	43	43	***42***	720	-1	0	-1
	174			1699		41	41	42	***41***	710	-1	0	0
	175			1742		40	40	41	***40***	770	-1	0	0
	176			1787		39	39	40	***39***	710	-1	0	0
	177			1834		38	38	39	***38***	720	-1	0	0
	178			1883		37	37	38	***37***	770	-1	0	0
	179			1935		36	36	37	***36***	710	-1	0	0
	180			1991		35	35	36	***35***	770	-1	0	0
	181			2049		34	34	35	***34***	710	-1	0	0
	182			2111		33	33	34	***33***	720	-1	0	0
	183			2177		32	32	33	***32***	770	-1	0	0
	184			2247		31	31	32	***31***	710	-1	0	0
	185			2322		30	30	31	***30***	770	-1	0	0
	186			2402		29	29	30	***29***	710	-1	0	0
	187			2488		28	28	29	***28***	720	-1	0	0
	188			2580		27	27	28	***27***	770	-1	0	0
	189			2680		26	26	27	***26***	710	-1	0	0
	190			2787		25	25	26	***25***	770	-1	0	0

Chapter 7

Conclusion

In this thesis, we study single model U-type assembly line balancing problem (UALBP). We consider a U-shaped line with constant operation times, no waiting times, and no walking times. Our objective is to minimize the number of stations, given the cycle time, *C*. This is achieved by finding a proper allocation of tasks to the stations.

We propose a new heuristic (Ant Colony Optimization (ACO) meta-heuristic) and its variants for the single model U-type assembly line balancing problem (UALBP). Although there are two ant algorithm developed for the single assembly line balancing problem (Bautista and Pereira, 2002; McMullen and Tarasewich, 2003) to the best of our knowledge, this study is the first application of the ACO meta-heuristic to U-shaped production lines.

Even though several heuristics have been developed for SALBP (Erel and Sarin, 1998) for the single model UALBP, we could find only three heuristics in the literature. These are: RPWT based heuristic (Miltenburg and Wijngaard, 1994), branch and bound based heuristic (Scholl and Klein, 1999) and simulated annealing based heuristic (Erel, Sabuncuoglu and Aksu, 2001).

Proposed methods for UALBP are considered in three groups: (i) directly methods, (ii) modified methods, and (iii) methods in which ACO approach is augmented with some metaheuristic. The first group includes algorithms such as AS, AS_{elite}, AS_{rank}, and ACS that is directly applied to UALBP. No modification is done in the structure of the algorithms. The second group includes new approaches that the structure of the algorithms is modified. Also, performance of the algorithms in this group is much better than the first group. The last group includes two specialized approaches that ACO is augmented with simulated annealing (SA) and beam search (BS).

The performance of the proposed algorithms is tested by using benchmark problems available in the literature. We use proposed algorithms (*Ant System*, *Ant System with Elitist Strategy*, *Ant Colony System*, *New Ant Colony Optimization Approach; Version 1* and *Version 2*) to solve 190 instances from these data sets and compare the results with those from ULINO (Scholl and Klein, 1999) and the simulated annealing based algorithm proposed by Erel, Sabuncuoglu and Aksu (2001).

The computational requirements are not high for any of the proposed algoritms. The algorithms are coded in Borland Delphi 6.0. The average computational time for an experiment requires a few seconds on an AMD Athlon XP 2000+, 266 Mhz machine with 256 MB RAM (333MHZ).

AS is the first algorithm used to solve UALBP. Later we apply AS_{elite}, ACS and the performance of these algorithms is better than AS. In these methods, some deficiencies of AS are improved. However, none of the algorithms give sufficient performance for UALBP. Structure of these algorithms, especially AS_{elite}, AS_{rank}, is not suitable for UALBP.

Actually, this is related with the topology of the cost function. Hertz and Widmer (2003) state that the topology of the cost function should not be too flat

for the heuristics to get the optimal solution. The cost function can be considered as an altitude with mountains, valleys and plateaus. If the cost function is too flat, it is difficult for the search algoritm to escape from the large plateaus to fall into the valleys. To tackle this problem Hertz and Widmer (2003) suggest adding a component to the cost function to discriminate the solutions with the same original cost function value.

The second group methods are modified from the first group to tackle this problem. Their *task selection* and *pheromone trail update mechanisms* are totally improved. In general, their performance is better than that of the first group.

The third group includes algorithms in which ACO is augmented with a metaheuristic. One of them is a modified version of ACS augmented with SA and the other one is a modified version of ACS augmented with beam search (BS). ACS with SA performs poor in terms of computational time. Even for the second smallest problem Jackson with 11 elements, computation time ranges between 2.53 hours and 164.63 hours. The structure of the beam search is very suitable for ACS however the performance of the algorithm was poor in terms of computational time. It requires excessive amount of time to complete a single tour.

We can list the following directions for further research:

(i) The proposed algorithms can be applied to different U-line configurations such as, stochastic assembly line balancing problems, mixed model assembly systems, etc.

It is possible to extend the proposed algorithms for more complex U-lines such as: *multi-lines in a single U, double-dependent U-lines, embedded U-lines,* and *multi-U-line facility.*

It is more realistic to consider a mixed-model U-line (a mixed-model production line is a line where different products are produced) instead of a single

model U-line. Because in todays business environment, many firms produce different kinds of products in order to satisfy the customer's needs.

(ii) It would be interesting to augment the new Ant Colony Optimization (ACO) approach (*Version 1* and *Version 2*) with a metaheuristic. However this augmentation must be done in an efficient way to prove the fast search of the search space.

(iii) A meta-heuristic can be used to find the best set of the parameters. When there are many parameters it will be tedious to fine tune these parameters by using an experimental design. In such cases, it may be more intelligent to use a meta-heuristic to find the best values for these parameters. Botee and Bonabeau (1998) use a genetic algorithm to select some of the parameters of the ACO algorithm. For our proposed algorithms, it is possible to use a genetic algorithm or an another meta-heuristic too.

(iv) The new Ant Colony Optimization (ACO) approach (*Version 1* and *Version 2*) can be effectively used to solve Type II problems (the minimization of cycle time, given the number of stations). In that case, we would have to re-balance an existing line with a particular number of stations.

Bibliography

[1] Bautista, J., Pereira, J., "Ant Algorithms for Assembly Line Balancing", In *Proceedings of Ant Algorithms, Third International Workshop, ANTS 2002,* Dorigo, M., Caro, D.G., Sampels, M., (Eds), Brussels, Belgium, 65-75, 2002.

[2] Baybars, I., "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem", *Management Science*, 32, 909-932, 1996.

[3] Baysakoglu, A., Gindy, N.Z. Nabil, "A Simulated Annealing Algorithm for Dynamic Layout", *Computers & Operations Research,* 28, 1403-1426, 2001.

[4] Besten, M., Stutzle, T., Dorigo, M., "Ant Colony Optimization for the Total Weighted Tardiness Problem". In *Parallel Problem Solving from Nature: 6th International Conference*, Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P., (Eds), Berlin, Springer Verlag. Vol. 1917 of LNCS, 611-620, 2000. (Also available as Tech.Rep.IRIDIA/99-16, Université Libre de Bruxelles, Belgium.)

[5] Besten, M.L., Stutzle, T., Dorigo, M., "An Ant Colony Optimization Application to the Single Machine Total Weighted Tardiness Problem". In *Abstract Proceedings of ANTS'2000: From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, Dorigo, M., Middendorf, M., Stützle, T., (Eds), Brussels, Belgium, 39-42, 2000.

[6] Blum, C., Sampels, M., "Ant Colony Optimization for FOP Shop Scheduling: A Case Study on Different Pheromone Representations". In *Proceedings of*

the 2002 Congress on Evolutionary Computation (CEC '02), IEEE Computer Society Press, 2, 1558-1563, 2002.

[7] Botee, H.M., Bonabeau, E., "Evolving Ant Colony Optimization", *Advance Complex Systems,* 1, 149-159, 1998.

[8] Bullnheimer, B., Hartl, R.F., Strauss, C., "A New Rank-Based Version of the Ant System: A Computational Study". *Central European Journal for Operations Research and Economics,* 7-1, 25-38, 1999.

[9] Bullnheimer, B., Kotsis, G., Strauss, C., "Parallelization Strategies for the Ant System". In Leone, R.D., Murli, A., Pardalos, P., Toraldo, G., (Eds.), *High Performance Algorithms and Software in Nonlinear Optimization,* Vol 24 of *Applied Optimization,* 87-100, Kluwer Academic Publishers, Dordrecht, NL, 1998.

[10] Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., Trubian, M., "Heuristics from Nature for Hard Combinatorial Optimization Problems", *International Transactions in Operational Research,* 3, 1, 1-21, 1996.

[11] Colorni, A., Dorigo, M., Maniezzo, V., "An Investigation of Some Properties of an Ant Algorithm", *Proceedings of the Parallel Problem Solving From Nature Conference (PPSN 92)*, Brussels, Belgium, Manner, R., Manderick, B., (Eds.), Elsevier Publishing, 509-520, 1992.

[12] Colorni, A., Dorigo, M., Maniezzo, V., "Distributed Optimization by Ant Colonies", *Proceedings of ECAL91-European Conference on Artificial Life,* Paris, France*,* Elsevier Publishing, 134-142, 1991.

[13] Deneubourg, J.L., Aron, S., Goss, S., Paspeels, J.M., "The self-organizing exploratory pattern of the argentine ant", *Journal of Insect Behavior,* 3, 159-168, 1990.

[14] Dorigo, M., Caro, D.C., and Gambardella, L.M., "Ant Algorithms for Discrete Optimization", *Artificial Life,* 5(2), 137-172, 1999.

[15] Dorigo, M., Gambardella, L.M., "A Study of Some Properties of Ant-Q". In *Proceedings of PPSN IV- Fourth International Conference on Parallel Problem Solving From Nature,* Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.S. (Eds.), Springer-Verlag, Berlin 656-665, 1996.

[16] Dorigo, M., Gambardella, L.M., "Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem", *IEEE Transactions on Evolutionary Computation,* 1-1, 53-66, 1997a.

[17] Dorigo, M., Gambella, L.M., "Ant Colonies for the Traveling Salesman Problem", *BioSystems,* 43, 73-81, 1997b.

[18] Dorigo, M., Maniezzo, V., Colorni, A., "Ant System: An Autocatalytic Optimizing Process", Technical Report 91-016 Revised, Dipartmentio di Eletronica e Informazione, Politecnico di Mileno, Italy, 1991.

[19] Dorigo, M., Maniezzo, V., Colorni, A., "Positive Feedback as a Search Strategy", Technical Report 91-016, Dipartmentio di Eletronica, Politecnico di Mileno, Italy, 1991.

[20] Dorigo, M., Maniezzo, V., Colorni, A., "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, and Cybernetics – Part B,* 26-1, 1-13, 1996.

[21] Dorigo, M., Stutzle, T., "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances", Technical Report, IRIDIA/2000-32, IRIDIA, University Libre de Bruxelles, Belgium, 2000.

[22] Erel, E. Sarin, S.C., "A Survey of the Assembly Line Balancing Procedures", *Production Planning and Control,* 9, 414-434, 1998.

[23] Erel, E., Sabuncuoglu, I., Aksu, B.A., "Balancing of U-Type Assembly Systems Using Simulated Annealing", *International Journal of Production Research*, 39-13, 3003-3015, 2001.

[24] Fenet S., Hassas S., "A.N.T: A Distributed Problem-Solving Framework Based on Mobile Agents". In *Advances in Mobile Agents Systems Research*, Vol. 1., *Theory and Practice*, G.E. Lasker, J. Dospisil, E. Kendall, Kendall, (Eds.), MAA'2000, *International Institute for Advanced Studies in Systems Research and Cybernetics*, 39-44, 2000.

[25] Gagné C., Gravel M., Price W.L., "A Look-ahead Addition to the Ant Colony Optimization Algorithm and its Application to an Industrial Scheduling Problem". In *Proceedings of the – 4th Metaheuristics International Conference (MIC'2001)*, Porto, Portugal, 79-84, 2001.

[26] Gambardela, L.M., Dorigo, M., "Solving Symmetric and Asymmetric TSPs by Ant Colonies". In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96),* IEEE Press, Piscataway, NJ, 622-627, 1996.

[27] Gambardella, L.M., Dorigo, M., "Ant-Q: A Reinforcement Learning Approach to the Travelling Salesman Problem". In *Proceedings of the Twelfth International Conference on Machine Learning (ML-95),* Prieditis, A., Russell, S., (Eds.), Morgan Kaufmann Publishers, Palo Alto, CA, 252-260, 1995.

[28] Glover, F., Greenberg, H.J., "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence", *European Journal of Operational Research*, 39, 119-130, 1989.

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

[30] Grassé, P.P., "La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs", *Insectes Sociaux,* 6, 41-81, 1959.

[31] Hertz, A., Widmer, M., "Guidelines for the Use of Meta-Heuristics in Combinatorial Optimization", *European Journal of Operational Research,* 151, 247-252, 2003.

[32] Hoffmann, T.R., "Assembly Line Balancing - A Set of Challenging Problems", *International Journal of Production Research*, 28, 1807-1815, 1990.

[33] Hoffmann, T.R., EUREKA: "A Hybrid System for Assembly Line Balancing", *Management Science*, 38, 39-42, 1992.

[34] Kim, Y.K., Kim, S.J., Kim, J.Y., "Balancing and Sequencing Mixed- Model U-lines with a Co-evolutionary Algorithm", *Production Planning & Control,* 11-8, 754-764, 2000.

[35] Krishnaiyer, K., Cheraghi, S.H., "Ant Algorithms: Review and Future Applications". In *IERC´02, Industrial Engineering Research Conference*, Orlando, Florida, USA, 2002.

[36] Maniezzo, V., Carbonaro, A., "Ant Colony Optimization: An Overview". In Ribeiro, C., (Eds.) *Essays and Surveys in Metaheuristics*, Kluwer, 21-44, 2001.

[37] McMullen, P.R., Tarasewich, P. "Using Ant Techniques to Solve the Assembly Line Balancing Problem", *IIE Transactions,* 35, 605-617, 2003.

[38] Middendorf, M., Reischle, F., Schmeck, H., "Multi Colony Ant Algorithms", *Journal of Heuristics,* Kluwer Akademic Publishers, 305-320, 2002.

[39] Miltenburg, G.J., Sparling, D., "Optimal Solution Algorithms for the U-line Balancing Problem", Working Paper, McMaster University, Hamilton, Ontario, Canada, 1995.

[40] Miltenburg, G.J., Wijngaard, J., "The U-line Line Balancing Problem", *Management Science,* 40-10, 1378-1388, 1994.

[41] Miltenburg, J., "Balancing U-lines in a Multiple U-line Facility", *European Journal of Operational Research*, 109, 1-23, 1998.

[42] Miltenburg, J., "One-piece Flow Manufacturing on U-shaped Production Lines: A Tutorial", *IIE Transactions,* 33, 303-321, 2001b.

[43] Miltenburg, J., "The Effect of Breakdowns on U-shaped Production Lines", *International Journal of Production Research,* 38, 353-364, 2000.

[44] Miltenburg, J., "U-shaped Production Lines: A Review of Theory and Practice", *International Journal of Production Economics*, 70, 201-214, 2001a.

[45] Monden, Y., "Toyota Production System" (Norcross, GA: Industrial Engineering and Management Press, Institute of Industrial Engineers), 1993.

[46] Montgomery, J., Randall, M., "Alternative Pheromone Applications for Ant Colony Optimization", Technical Report TR02-07, School of Information Technology, Bond University, Australia, 2002.

[47] Nakade, K., Ohno, K., "An Optimal Worker Allocation Problem for a U-shaped Production Line", *International Journal of Production Economics*, 60-1, 353-358, 1999.

[48] Nakade, K., Ohno, K., "Separate and Carousel Type Allocations of Workers in a U-shaped Production Line", *European Journal of Operational Research,* 145, 403-424, 2003.

[49] Nakade, K., Ohno, K., Shanthikumar, J.G., "Bounds and Approximations for Cycle Times of a U-shaped Production Line", *Operations Research Letters,* 21, 191-200, 1997.

[50] Nakade, K., Ohno, K., "Stochastic Analysis of a U-shaped Production Line with Multiple Workers", *Computers and Industrial Engineering,* 33, 809-812, 1997.

[51] Reeves, C.R. (ed.), "Modern Heuristic Techniques for Combinatorial Problems", Blackwell Scientific Press, Oxford, 1993.

[52] Sabuncuoglu, I., Bayiz, M., "Job Shop Scheduling with Beam Search", *European Journal of Operational Research,* 118, 390-412, 1999.

[53] Salveson, M.E., "The Assembly Line Balancing Problem", *Journal of Industrial Engineering*, 6, 18-25, 1955.

[54] Scholl, A. "Data of Assembly Line Balancing Problems". Schriften zur Quantitativen Betriebswirtschaftslehre 16/93, TU Darmstadt, 1993.

[55] Scholl, A., Klein, R., "ULINO: Optimally Balancing U-shaped JIT Assembly Lines", *International Journal of Production Research*, 37-4, 721-736, 1999.

[56] Sparling, D., "Balancing Just-In-Time Production Units: The N U-line Balancing Problem", *Information Systems and Operational Research*, 36-4, 215-237, 1998.

[57] Sparling, D., Miltenburg, J., The Mixed-Model U-line Balancing Problem, *International Journal of Production Research*, 36-2, 485-501, 1998.

[58] Stutze, T., Dorigo, M., "ACO Algorithms for the Travelling Salesman Problem". In Miettinen, K., Makela, M., Neittaanmaki, P., and Periaux, J., (Eds.), *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications.* John Wiley & Sons., 163-183, 1999.

[59] Stutzle, T., "An Ant Approach to the Flow Shop Problem". In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EU-FIT'98),* Verlag Mainz, Aachen, 3, 1560-1564, 1998.

[60] T'kindt, V., Monmarché, N., Tercinet, F., Laügt, D., "An Ant Colony Optimization Algorithm to Solve a 2-Machine Bicriteria Flowshop Scheduling Problem", *European Journal of Operational Research,* 142, 250-257, 2002.

[61] Talbot, F.B., J.H. Patterson, Gehrlein, W.V., "A comparative evaluation of heuristic line balancing techniques", *Management Science* 32, 430 – 454, 1986.

[62] Urban, T.L., "Optimal Balancing of U-shaped Assembly Lines", *Management Science,* 44, 738-741, 1998.

[63] Zanakis, S.H., Evans, J.R., Vazacopoulos, A.A., "Heuristic Methods and Applications: A categorized Survey", *European Journal of Operations Research,* 43, 88-110, 1989.

A.1 APPENDIX A

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.1.1: Trail matrix gathered for tour number 10

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.65	0.01	0.40	0.01	0.45	0.01	0.08	0.01	0.06	0.01	0.01	0.01	0.01	0.01
2	0.50	0.01	0.55	0.01	0.22	0.02	0.26	0.01	0.11	0.01	0.04	0.01	0.01	0.01
3	0.36	0.01	0.50	0.01	0.23	0.02	0.26	0.02	0.26	0.01	0.05	0.01	0.02	0.02
4	0.09	0.01	0.27	0.01	0.22	0.03	0.17	0.03	0.31	0.05	0.21	0.08	0.12	0.03
5	0.41	0.01	0.45	0.01	0.40	0.01	0.30	0.01	0.07	0.01	0.01	0.01	0.01	0.01
6	0.26	0.01	0.36	0.01	0.27	0.01	0.26	0.01	0.26	0.01	0.16	0.01	0.11	0.01
7	0.26	0.01	0.36	0.01	0.27	0.01	0.22	0.01	0.31	0.01	0.16	0.01	0.12	0.02
8	0.22	0.01	0.31	0.01	0.27	0.01	0.22	0.00	0.31	0.09	0.17	0.03	0.11	0.04
9	0.01	0.01	0.05	0.01	0.21	0.04	0.22	0.08	0.35	0.08	0.22	0.13	0.11	0.09
10	0.23	0.01	0.26	0.01	0.31	0.01	0.31	0.01	0.26	0.01	0.18	0.01	0.11	0.01
11	0.01	0.01	0.09	0.09	0.10	0.09	0.11	0.07	0.14	0.13	0.11	0.26	0.17	0.26
12	0.18	0.01	0.31	0.01	0.21	0.01	0.41	0.01	0.26	0.01	0.15	0.01	0.13	0.01
13	0.01	0.01	0.00	0.21	0.05	0.08	0.03	0.22	0.17	0.26	0.01	0.21	0.06	0.18
14	0.01	0.01	0.23	0.01	0.36	0.01	0.21	0.01	0.26	0.01	0.26	0.02	0.21	0.09
15	0.01	0.01	0.07	0.01	0.21	0.01	0.14	0.01	0.31	0.03	0.41	0.06	0.26	0.14
16	0.01	0.01	0.01	0.01	0.09	0.01	0.14	0.04	0.12	0.08	0.18	0.21	0.31	0.26
17	1.45	0.01	0.19	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
18	0.01	0.01	0.17	0.01	0.36	0.01	0.27	0.01	0.31	0.01	0.21	0.05	0.31	0.08
19	0.01	0.01	0.03	0.01	0.17	0.01	0.21	0.01	0.36	0.02	0.31	0.04	0.21	0.21
20	0.01	0.01	0.01	0.01	0.02	0.01	0.12	0.07	0.12	0.06	0.26	0.22	0.36	0.26
21	0.01	0.01	0.01	0.01	0.01	0.11	0.01	0.12	0.01	0.31	0.07	0.31	0.35	0.23
22	0.01	0.01	0.01	0.01	0.01	0.16	0.01	0.12	0.01	0.40	0.05	0.36	0.26	0.26
23	0.01	0.01	0.01	0.01	0.01	0.26	0.01	0.31	0.01	0.31	0.01	0.23	0.06	0.31
24	0.01	0.01	0.01	0.13	0.01	0.30	0.01	0.31	0.01	0.31	0.01	0.27	0.01	0.31
25	0.01	0.01	0.01	0.21	0.01	0.36	0.01	0.36	0.01	0.18	0.04	0.40	0.11	0.08
26	0.01	0.01	0.01	0.21	0.01	0.45	0.01	0.27	0.01	0.31	0.00	0.31	0.09	0.08
27	0.01	0.01	0.01	0.55	0.01	0.27	0.01	0.31	0.01	0.26	0.01	0.17	0.01	0.15
28	0.01	0.55	0.01	0.60	0.01	0.26	0.01	0.18	0.01	0.09	0.01	0.04	0.01	0.06
29	0.01	1.30	0.01	0.31	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
30	0.01	0.40	0.01	0.17	0.01	0.26	0.01	0.26	0.01	0.19	0.02	0.26	0.07	0.02
31	0.01	0.40	0.01	0.21	0.01	0.31	0.01	0.22	0.01	0.22	0.01	0.17	0.01	0.12
32	0.01	0.40	0.01	0.36	0.01	0.31	0.01	0.31	0.01	0.14	0.01	0.12	0.01	0.11
33	0.01	0.40	0.01	0.36	0.01	0.31	0.01	0.31	0.01	0.14	0.01	0.16	0.01	0.01
34	0.01	1.30	0.01	0.31	0.01	0.04	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
35	0.01	1.35	0.01	0.27	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

A.1 APPENDIX A (Cont'd)

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.1.2: Trail matrix gathered for tour number 100

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.75	0.00	0.45	0.00	0.31	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.05	0.00
2	0.55	0.00	0.50	0.00	0.36	0.00	0.22	0.00	0.06	0.00	0.06	0.00	0.00	0.05
3	0.40	0.00	0.45	0.00	0.22	0.00	0.45	0.00	0.17	0.00	0.06	0.05	0.03	0.00
4	0.00	0.00	0.17	0.00	0.36	0.00	0.26	0.00	0.26	0.00	0.31	0.07	0.23	0.09
5	0.45	0.00	0.50	0.00	0.40	0.00	0.27	0.00	0.08	0.00	0.00	0.00	0.06	0.00
6	0.26	0.00	0.45	0.00	0.31	0.00	0.27	0.00	0.22	0.00	0.12	0.00	0.05	0.00
7	0.26	0.00	0.41	0.00	0.36	0.00	0.24	0.00	0.23	0.00	0.16	0.00	0.02	0.05
8	0.26	0.00	0.31	0.00	0.26	0.00	0.36	0.00	0.20	0.16	0.15	0.04	0.07	0.11
9	0.00	0.00	0.26	0.00	0.17	0.00	0.22	0.13	0.26	0.07	0.13	0.16	0.19	0.22
10	0.31	0.00	0.26	0.00	0.22	0.00	0.40	0.00	0.23	0.00	0.18	0.00	0.07	0.00
11	0.00	0.00	0.00	0.00	0.09	0.14	0.13	0.19	0.08	0.26	0.13	0.18	0.10	0.31
12	0.22	0.00	0.27	0.00	0.31	0.00	0.22	0.00	0.26	0.00	0.26	0.00	0.21	0.00
13	0.00	0.00	0.00	0.00	0.12	0.18	0.01	0.31	0.16	0.26	0.08	0.17	0.07	0.23
14	0.00	0.00	0.20	0.00	0.41	0.00	0.31	0.00	0.27	0.00	0.22	0.06	0.16	0.06
15	0.00	0.00	0.00	0.00	0.18	0.00	0.27	0.00	0.36	0.00	0.41	0.06	0.14	0.16
16	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.17	0.14	0.31	0.26	0.31	0.22
17	1.30	0.00	0.35	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.23	0.00	0.26	0.00	0.26	0.00	0.31	0.00	0.26	0.09	0.31	0.02
19	0.00	0.00	0.00	0.00	0.18	0.00	0.21	0.00	0.31	0.04	0.31	0.09	0.40	0.12
20	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.13	0.13	0.31	0.22	0.31	0.22
21	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.17	0.08	0.27	0.11	0.27	0.35	0.36
22	0.00	0.00	0.00	0.00	0.00	0.26	0.00	0.18	0.00	0.31	0.14	0.23	0.21	0.36
23	0.00	0.00	0.00	0.00	0.00	0.27	0.00	0.22	0.00	0.36	0.00	0.36	0.18	0.22
24	0.00	0.00	0.00	0.45	0.00	0.14	0.00	0.36	0.00	0.31	0.00	0.22	0.12	0.08
25	0.00	0.00	0.00	0.08	0.00	0.31	0.00	0.45	0.08	0.50	0.06	0.22	0.12	0.05
26	0.00	0.00	0.00	0.08	0.00	0.50	0.00	0.41	0.04	0.36	0.06	0.22	0.12	0.07
27	0.00	0.00	0.00	0.50	0.00	0.36	0.00	0.41	0.00	0.26	0.00	0.08	0.08	0.12
28	0.00	0.45	0.00	0.65	0.00	0.41	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00
29	0.00	1.25	0.00	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.50	0.00	0.10	0.00	0.21	0.00	0.24	0.00	0.40	0.04	0.22	0.12	0.01
31	0.00	0.50	0.00	0.17	0.00	0.36	0.00	0.27	0.00	0.26	0.02	0.13	0.06	0.12
32	0.00	0.50	0.00	0.35	0.00	0.36	0.00	0.22	0.00	0.22	0.02	0.07	0.02	0.12
33	0.00	0.50	0.00	0.41	0.00	0.31	0.00	0.26	0.00	0.18	0.00	0.13	0.00	0.00
34	0.00	1.30	0.00	0.31	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
35	0.00	1.40	0.00	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

A.1 APPENDIX A (Cont'd)

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.1.3: Trail matrix gathered for tour number 500

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.60	0.00	0.45	0.00	0.50	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.02	0.00
2	0.46	0.00	0.50	0.00	0.45	0.00	0.31	0.00	0.08	0.02	0.02	0.00	0.00	0.00
3	0.27	0.00	0.55	0.00	0.31	0.00	0.31	0.00	0.26	0.02	0.09	0.00	0.01	0.00
4	0.00	0.00	0.36	0.00	0.13	0.00	0.26	0.00	0.26	0.07	0.27	0.06	0.22	0.12
5	0.32	0.00	0.50	0.00	0.50	0.00	0.31	0.00	0.13	0.00	0.00	0.00	0.02	0.00
6	0.20	0.00	0.31	0.00	0.40	0.00	0.31	0.00	0.22	0.00	0.21	0.00	0.11	0.00
7	0.20	0.00	0.31	0.00	0.40	0.00	0.31	0.00	0.23	0.00	0.21	0.00	0.10	0.06
8	0.20	0.00	0.31	0.00	0.22	0.00	0.31	0.03	0.31	0.12	0.21	0.04	0.16	0.00
9	0.00	0.00	0.15	0.00	0.18	0.00	0.31	0.08	0.14	0.13	0.19	0.12	0.17	0.21
10	0.26	0.00	0.31	0.00	0.31	0.00	0.31	0.00	0.27	0.00	0.26	0.00	0.10	0.00
11	0.00	0.00	0.00	0.00	0.14	0.17	0.03	0.22	0.10	0.22	0.00	0.22	0.26	0.14
12	0.22	0.00	0.31	0.00	0.17	0.00	0.36	0.00	0.41	0.00	0.31	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00	0.13	0.19	0.09	0.26	0.18	0.26	0.17	0.23	0.01	0.13
14	0.00	0.00	0.22	0.00	0.26	0.00	0.27	0.00	0.27	0.00	0.45	0.06	0.23	0.07
15	0.00	0.00	0.00	0.00	0.18	0.00	0.26	0.00	0.18	0.00	0.41	0.06	0.26	0.18
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.12	0.26	0.27	0.31	0.35
17	1.50	0.00	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.23	0.00	0.26	0.00	0.22	0.00	0.41	0.00	0.40	0.00	0.15	0.13
19	0.00	0.00	0.00	0.00	0.22	0.00	0.31	0.00	0.22	0.00	0.22	0.08	0.31	0.18
20	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.21	0.22	0.22	0.22	0.31	0.22
21	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.22	0.11	0.50	0.22	0.17	0.18	0.27
22	0.00	0.00	0.00	0.00	0.00	0.26	0.00	0.26	0.00	0.36	0.21	0.26	0.23	0.17
23	0.00	0.00	0.00	0.00	0.00	0.31	0.00	0.36	0.00	0.31	0.11	0.23	0.14	0.22
24	0.00	0.00	0.00	0.45	0.00	0.22	0.00	0.31	0.00	0.17	0.00	0.22	0.17	0.21
25	0.00	0.00	0.00	0.21	0.00	0.31	0.00	0.40	0.11	0.32	0.17	0.14	0.04	0.07
26	0.00	0.00	0.00	0.21	0.00	0.50	0.00	0.36	0.06	0.23	0.17	0.12	0.11	0.12
27	0.00	0.00	0.00	0.65	0.00	0.36	0.00	0.26	0.00	0.18	0.00	0.08	0.13	0.13
28	0.00	0.55	0.00	0.60	0.00	0.26	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.00
29	0.00	1.25	0.00	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.55	0.00	0.13	0.00	0.16	0.00	0.36	0.05	0.36	0.08	0.17	0.06	0.03
31	0.00	0.55	0.00	0.17	0.00	0.40	0.00	0.27	0.00	0.22	0.09	0.12	0.02	0.05
32	0.00	0.55	0.00	0.36	0.00	0.31	0.00	0.26	0.00	0.22	0.09	0.06	0.00	0.05
33	0.00	0.55	0.00	0.36	0.00	0.36	0.00	0.22	0.00	0.22	0.00	0.00	0.00	0.10
34	0.00	1.60	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
35	0.00	1.35	0.00	0.36	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

A.1 APPENDIX A (Cont'd)

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.1.4: Trail matrix gathered for tour number 1000

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.65	0.00	0.55	0.00	0.31	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.06	0.00
2	0.45	0.00	0.70	0.00	0.25	0.00	0.18	0.00	0.16	0.00	0.03	0.00	0.06	0.00
3	0.22	0.00	0.45	0.00	0.41	0.00	0.31	0.00	0.15	0.00	0.21	0.00	0.03	0.06
4	0.00	0.00	0.21	0.00	0.26	0.00	0.19	0.00	0.31	0.00	0.13	0.03	0.26	0.22
5	0.45	0.00	0.60	0.00	0.28	0.00	0.31	0.00	0.09	0.00	0.00	0.00	0.02	0.06
6	0.21	0.00	0.50	0.00	0.26	0.00	0.41	0.00	0.18	0.00	0.14	0.06	0.07	0.00
7	0.21	0.00	0.50	0.00	0.22	0.00	0.41	0.00	0.22	0.00	0.14	0.06	0.09	0.05
8	0.17	0.00	0.40	0.00	0.31	0.00	0.22	0.05	0.31	0.07	0.16	0.06	0.13	0.06
9	0.00	0.00	0.12	0.00	0.21	0.00	0.31	0.09	0.18	0.08	0.26	0.09	0.18	0.14
10	0.26	0.00	0.40	0.00	0.26	0.00	0.36	0.00	0.31	0.00	0.15	0.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.08	0.13	0.21	0.23	0.09	0.26	0.16	0.17	0.09	0.22
12	0.22	0.00	0.26	0.00	0.35	0.00	0.27	0.00	0.31	0.00	0.18	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00	0.00	0.26	0.11	0.18	0.06	0.26	0.00	0.26	0.15	0.36
14	0.00	0.00	0.31	0.00	0.36	0.00	0.22	0.00	0.45	0.00	0.31	0.06	0.07	0.10
15	0.00	0.00	0.00	0.00	0.26	0.00	0.22	0.00	0.26	0.00	0.40	0.08	0.41	0.08
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.31	0.22	0.26	0.31
17	1.55	0.00	0.18	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.22	0.00	0.50	0.00	0.17	0.00	0.36	0.06	0.31	0.05	0.04	0.17
19	0.00	0.00	0.00	0.00	0.31	0.00	0.22	0.00	0.31	0.06	0.36	0.09	0.22	0.17
20	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.00	0.21	0.17	0.31	0.26	0.31	0.22
21	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.26	0.04	0.31	0.35	0.45	0.22	0.15
22	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.26	0.00	0.40	0.17	0.36	0.26	0.23
23	0.00	0.00	0.00	0.00	0.00	0.31	0.00	0.31	0.00	0.36	0.06	0.31	0.26	0.13
24	0.00	0.00	0.00	0.18	0.00	0.36	0.00	0.31	0.00	0.36	0.00	0.22	0.21	0.09
25	0.00	0.00	0.00	0.36	0.00	0.20	0.00	0.22	0.04	0.28	0.21	0.35	0.06	0.12
26	0.00	0.00	0.00	0.36	0.00	0.36	0.00	0.20	0.00	0.27	0.13	0.31	0.07	0.06
27	0.00	0.00	0.00	0.50	0.00	0.32	0.00	0.36	0.00	0.22	0.00	0.08	0.11	0.12
28	0.00	0.55	0.00	0.50	0.00	0.31	0.00	0.21	0.00	0.00	0.00	0.00	0.00	0.00
29	0.00	1.30	0.00	0.41	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.50	0.00	0.16	0.00	0.31	0.00	0.19	0.00	0.22	0.10	0.26	0.01	0.07
31	0.00	0.50	0.00	0.22	0.00	0.36	0.00	0.22	0.00	0.14	0.07	0.26	0.02	0.08
32	0.00	0.50	0.00	0.41	0.00	0.27	0.00	0.26	0.00	0.17	0.07	0.12	0.00	0.06
33	0.00	0.50	0.00	0.41	0.00	0.27	0.00	0.31	0.00	0.21	0.00	0.00	0.00	0.08
34	0.00	1.45	0.00	0.26	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
35	0.00	1.40	0.00	0.31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

A.2 APPENDIX B

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.2.1: T2 matrix gathered for tour number 5000

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	4106	0	2711	0	2263	0	825	0	3	0	3	0	45	0
2	3007	0	2978	0	2370	1	1267	0	239	3	33	5	50	3
3	1572	0	2828	0	2202	2	1948	2	957	5	320	10	61	34
4	7	0	1256	0	1359	3	1488	3	1410	47	1335	81	1216	369
5	2592	0	2780	0	2443	0	1746	0	315	0	26	0	3	36
6	1238	0	2350	0	1868	0	2054	0	1593	0	672	37	49	5
7	1206	0	2239	0	1846	0	1932	0	1731	0	803	37	63	32
8	1128	0	1761	0	1723	8	1801	85	1731	154	978	115	268	96
9	0	0	917	0	1000	13	1044	288	1196	516	1113	608	943	848
10	1400	0	1764	0	1497	0	1847	0	1657	0	1097	0	194	5
11	0	0	5	53	617	226	455	1161	375	1331	556	1163	530	1431
12	1461	0	1748	0	1475	0	1799	0	1713	0	1037	0	342	0
13	0	0	2	838	219	1003	508	1115	358	1270	448	1089	376	1104
14	0	0	1356	0	1761	0	1609	0	2177	2	1881	90	494	376
15	0	0	6	0	1197	0	1272	0	1696	4	2078	114	1985	666
16	0	0	0	0	16	0	635	2	981	26	1436	1269	1676	1572
17	7963	0	1838	0	109	0	36	0	5	0	3	0	2	0
18	0	0	1396	0	1773	0	1578	1	2177	59	1832	144	550	286
19	0	0	15	0	1204	0	1313	1	1744	82	1956	221	1725	776
20	0	0	0	0	19	0	616	15	1043	706	1393	1152	1711	1319
21	0	0	0	0	0	582	9	776	290	1588	806	1876	1287	1863
22	0	0	0	0	0	947	2	1021	11	1716	630	1947	954	1779
23	0	0	0	0	0	1474	0	1484	2	1930	8	1887	657	1435
24	0	0	0	1356	0	1312	0	1666	0	1851	1	1644	128	1289
25	0	0	0	892	0	1665	7	2199	245	2263	357	1230	259	456
26	0	0	0	892	0	2373	1	2453	133	1897	263	1017	164	479
27	0	0	0	3101	0	1960	0	2132	0	1275	1	661	104	626
28	0	2884	0	3873	0	1835	0	978	0	32	0	7	0	315
29	0	8001	0	1916	0	34	0	4	0	1	0	0	0	0
30	0	2966	0	541	0	966	2	1662	60	1523	155	1035	280	350
31	0	2966	0	718	0	1774	1	1565	0	1177	95	851	108	461
32	0	2966	0	1672	0	1973	1	1318	0	982	71	627	59	271
33	0	2966	0	1940	0	1913	0	1332	0	1079	0	508	0	106
34	0	8170	0	1681	0	80	0	19	0	6	0	0	0	0
35	0	7982	0	1921	0	40	0	12	0	1	0	0	0	0

A.2 APPENDIX B (Cont'd)

Gunther problem, cycle time = 69; α=1, β=1, ρ_1=0.4, ρ_2=0.4, q_0=0.2, τ_0=0.0028.

(Optimal station number is 7. The trail values are considered only for 7 stations)

Table A.2.2: T2 matrix gathered for tour number 10000

Station	1		2		3		4		5		6		7	
Location / Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	139811	0	94356	0	71442	0	25039	0	4	0	7	0	744	0
2	102231	0	101452	0	81750	2	36735	59	7084	317	1077	71	642	246
3	55477	0	94467	0	76505	3	62303	110	29996	366	8933	147	2226	640
4	11	0	43575	0	46457	8	53880	447	48564	971	47217	5372	33832	11714
5	89111	0	94391	0	81735	0	53637	0	10496	0	1050	4	447	635
6	43749	0	78736	0	62966	0	68191	0	50634	0	22954	618	1292	144
7	42710	0	75098	0	61644	0	65516	0	55200	0	26655	622	1367	888
8	39462	0	59128	0	58848	3	59518	848	57827	2973	34189	3528	9225	2307
9	0	0	30978	0	33928	15	43448	3269	42505	13156	42631	26041	28529	24231
10	48134	0	60723	0	49557	0	60316	0	55088	0	36674	0	1119	380
11	0	0	9	89	18524	13560	23289	40242	11112	41403	20533	44195	17282	43144
12	47917	0	60963	0	49944	0	60531	0	55371	0	37351	0	1265	19
13	0	0	11	408	14840	39726	22658	41835	7482	44443	14708	45244	15720	37864
14	0	0	47428	0	61394	0	51026	1	72532	1	64154	3308	10660	12933
15	0	0	2	0	40844	0	46285	1	56681	12	71747	4816	59098	25407
16	0	0	0	0	5	0	5702	20	39708	235	53020	48339	60570	51357
17	264875	0	62022	0	3575	0	869	0	268	0	96	0	5	0
18	0	0	47099	0	61692	0	51130	0	73345	1178	63813	2308	11468	10992
19	0	0	2	0	40814	0	45913	0	56826	2032	72246	3356	61244	21846
20	0	0	0	0	10	0	1713	3	41199	25037	52891	1462	61486	59408
21	0	0	0	0	0	15012	17	30264	7103	54017	36375	63152	36919	63845
22	0	0	0	0	0	29239	0	36452	49	56077	26395	68685	29064	58038
23	0	0	0	0	0	49605	0	49589	0	64258	4469	65614	23152	42337
24	0	0	0	59691	0	32524	0	55950	0	61622	200	52577	13370	31418
25	0	0	0	37361	0	47945	5	72964	6269	76065	19943	40457	9377	11964
26	0	0	0	37361	0	72901	0	79872	869	64771	14079	32114	10781	9266
27	0	0	0	97522	0	64805	0	79937	0	37653	17	16229	11110	19684
28	0	93719	0	127142	0	62533	0	36778	0	14	0	65	0	0
29	0	266352	0	63928	0	1134	0	290	0	9	0	3	0	0
30	0	98186	0	19407	0	30424	12	57280	348	51702	12421	30696	7947	10533
31	0	98186	0	24543	0	57997	0	54405	1	38054	6025	24485	4787	12537
32	0	98186	0	54115	0	65582	0	50221	1	24939	5850	15569	1405	15474
33	0	98186	0	62813	0	62860	0	48962	0	35766	0	734	0	20079
34	0	270575	0	57761	0	2564	0	650	0	165	0	1	0	0
35	0	262345	0	67179	0	1571	0	588	0	27	0	2	0	4

A.3 APPENDIX C

Buxey problem, cycle time = 36; α=1, β=1, ρ_1=0.99, ρ_2=0.99, q_0=0.3, τ_0=1.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.3.1: Trail matrix gathered for tour number 100

Station / Task	1	2	3	4	5	6	7	8	9
1	182	167	54	153	60	104	53	77	30
2	164	190	7	17	2	13	25	67	82
3	2	161	175	164	73	145	117	102	79
4	0	107	172	166	123	149	142	132	119
5	0	1	7	153	168	155	144	162	144
6	104	172	177	87	88	91	39	108	115
7	174	161	173	127	113	60	36	20	71
8	0	0	0	0	126	166	161	174	162
9	153	170	175	150	109	102	42	37	74
10	17	119	144	165	164	145	148	109	117
11	0	0	0	0	2	174	179	148	140
12	0	56	153	168	151	154	7	160	116
13	0	0	0	29	139	124	167	162	167
14	0	19	115	160	158	153	157	153	137
15	0	0	2	3	6	154	167	161	178
16	0	0	0	0	129	165	169	168	160
17	0	0	0	0	176	173	129	148	143
18	0	0	0	2	177	166	162	154	118
19	0	0	0	0	23	133	158	167	177
20	0	0	0	179	180	47	45	140	113
21	0	0	74	177	172	147	145	101	74
22	0	0	111	180	167	131	132	94	86
23	0	0	184	177	115	1	73	22	7
24	182	174	168	9	29	0	4	1	34
25	0	0	18	116	44	92	123	150	171
26	0	177	159	159	158	80	64	59	73
27	0	10	61	124	122	84	138	152	172
28	172	180	168	127	1	3	2	7	15
29	192	162	40	6	0	0	0	0	1

A.3 APPENDIX C (Cont'd)

Buxey problem, cycle time = 36; α=1, β=1, ρ_1=0.99, ρ_2=0.99, q_0=0.3, τ_0=1.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.3.2: Trail matrix gathered for tour number 250

Station / Task	1	2	3	4	5	6	7	8	9
1	184	166	142	145	78	52	70	34	17
2	166	190	5	47	45	0	20	114	0
3	2	166	175	166	153	0	64	121	55
4	0	109	173	166	157	148	98	132	106
5	0	0	4	153	172	168	144	141	139
6	147	164	174	147	99	94	105	122	116
7	174	161	173	142	128	62	45	74	47
8	0	0	0	0	115	172	168	166	164
9	146	171	175	156	129	101	65	82	70
10	35	127	142	157	167	155	142	140	136
11	0	0	0	0	0	162	181	170	146
12	1	118	154	165	159	148	146	150	126
13	0	0	0	24	130	158	164	164	170
14	0	19	125	145	157	158	158	154	154
15	0	0	0	54	52	149	163	166	177
16	0	0	0	0	112	161	161	172	169
17	0	0	0	25	163	175	166	150	147
18	0	0	0	0	172	171	155	162	144
19	0	0	0	1	73	140	149	165	177
20	0	0	0	172	181	145	118	122	104
21	0	0	90	178	172	155	136	123	75
22	0	0	123	183	167	144	126	115	90
23	0	11	180	181	146	5	32	27	53
24	178	174	174	72	2	0	0	0	22
25	2	50	106	118	103	117	126	143	169
26	7	179	169	157	144	97	81	86	65
27	0	76	119	121	121	128	126	144	168
28	173	178	171	122	1	5	6	5	14
29	192	164	58	1	0	0	0	0	0

A.3 APPENDIX C (Cont'd)

Buxey problem, cycle time = 36; α=1, β=1, ρ_1=0.99, ρ_2=0.99, q_0=0.3, τ_0=1.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.3.3: Trail matrix gathered for tour number 1000

Station / Task	1	2	3	4	5	6	7	8	9
1	183	170	146	139	86	51	51	54	23
2	167	189	0	112	47	28	44	55	36
3	0	162	176	169	153	81	93	66	70
4	0	109	172	169	159	149	116	113	97
5	0	0	81	146	171	174	144	139	127
6	150	167	173	145	126	119	101	99	80
7	173	163	170	144	136	82	72	76	31
8	0	0	0	52	104	174	171	165	157
9	143	169	173	159	138	108	87	83	62
10	28	121	140	156	168	154	146	142	135
11	0	0	0	0	0	162	180	173	143
12	0	127	147	161	155	157	148	152	132
13	0	0	11	70	118	157	164	165	171
14	0	14	109	144	154	156	163	155	155
15	0	0	0	26	94	131	159	168	180
16	0	0	0	4	109	163	160	171	171
17	0	0	0	39	171	170	163	156	147
18	0	0	0	53	166	171	160	165	148
19	0	0	0	0	93	138	151	163	177
20	0	0	0	173	181	143	117	117	101
21	0	0	109	176	176	153	141	120	76
22	0	0	131	179	173	144	126	122	89
23	0	2	182	180	144	0	0	69	33
24	180	173	171	78	0	0	0	3	26
25	0	92	119	114	0	122	124	141	167
26	2	179	168	156	142	104	96	70	59
27	0	0	128	128	127	114	123	144	172
28	171	181	170	122	0	10	2	16	22
29	192	165	61	9	0	0	0	0	0

A.4 APPENDIX D

Buxey problem, cycle time= 36; α=1, β=1, ρ_1=0.9, ρ_2=0.9, q_0=0.3, τ_0=1, Q_2=30.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.4.1: Trail matrix gathered for tour number 100

Station / Task	1	2	3	4	5	6	7	8	9
1	904	0	0	0	0	0	0	0	0
2	904	0	2	0	0	0	0	0	0
3	0	74	257	25	175	0	0	53	0
4	0	159	352	65	167	75	69	22	2
5	0	0	0	12	80	300	82	35	57
6	639	389	164	0	10	0	118	0	10
7	441	0	141	1	59	17	68	11	2
8	0	0	0	0	2	5	262	9	10
9	441	2	177	422	20	42	68	2	37
10	0	396	207	50	461	4	145	29	32
11	0	0	0	0	0	0	75	56	10
12	0	0	85	13	294	50	266	1	8
13	0	0	0	0	10	38	89	17	11
14	0	1	37	361	169	66	72	6	9
15	0	0	0	0	0	98	71	60	9
16	0	0	0	0	1	241	10	8	15
17	0	0	0	0	342	29	32	9	16
18	0	0	0	0	168	267	5	10	16
19	0	0	0	0	23	49	208	7	13
20	0	0	0	0	542	0	102	3	8
21	0	0	0	8	221	60	141	156	18
22	0	0	0	474	259	105	28	0	21
23	0	0	0	667	10	0	9	0	14
24	0	601	423	4	0	0	0	0	1
25	0	0	0	59	47	0	0	6	16
26	0	589	381	245	19	10	17	0	0
27	0	0	116	14	11	90	335	22	9
28	0	0	125	345	22	0	0	0	0
29	0	784	154	0	0	0	0	0	0

A.4 APPENDIX D (Cont'd)

Buxey problem, cycle time= 36; α=1, β=1, ρ_1=0.9, ρ_2=0.9, q_0=0.3, τ_0=1, Q_2=30.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.4.2: Trail matrix gathered for tour number 250

Station / Task	1	2	3	4	5	6	7	8	9
1	904	0	0	0	0	0	0	0	0
2	904	0	0	0	0	0	0	0	0
3	0	0	842	147	0	0	0	89	0
4	0	0	4	416	86	152	41	5	3
5	0	0	0	374	115	237	17	29	8
6	627	291	0	104	171	0	24	1	1
7	442	0	4	233	267	0	12	0	0
8	0	0	0	1	414	4	198	7	11
9	442	0	1	230	222	106	5	9	13
10	0	358	2	39	394	64	148	29	47
11	0	0	0	0	0	0	9	28	10
12	0	174	1	148	498	85	156	6	9
13	0	0	0	383	0	134	153	14	34
14	0	21	1	161	461	73	230	9	8
15	0	0	0	0	178	0	1	29	11
16	0	0	0	0	6	93	117	10	11
17	0	0	0	0	0	245	12	16	9
18	0	0	0	0	0	206	30	10	23
19	0	0	0	0	106	115	197	27	75
20	0	0	0	0	510	0	157	8	37
21	0	0	0	0	410	97	207	73	162
22	0	0	0	0	461	55	183	11	16
23	0	0	0	60	0	43	0	0	92
24	0	716	307	16	0	0	0	0	2
25	0	87	677	0	0	0	2	0	5
26	0	708	15	164	202	29	3	1	0
27	0	0	0	2	254	86	169	21	41
28	0	0	842	18	0	0	1	11	0
29	0	907	1	0	0	0	0	12	0

A.4 APPENDIX D (Cont'd)

Buxey problem, cycle time= 36; α=1, β=1, ρ_1=0.9, ρ_2=0.9, q_0=0.3, τ_0=1, Q_2=30.

(Optimal station number is 9. The trail values are considered only for 9 stations)

Table A.4.3: Trail matrix gathered for tour number 1000

Station / Task	1	2	3	4	5	6	7	8	9
1	936	0	0	0	0	0	0	0	0
2	936	0	0	0	0	0	0	0	0
3	0	0	634	212	0	0	0	111	0
4	0	0	101	187	78	84	9	16	3
5	0	0	0	105	125	328	53	1	9
6	598	473	0	0	134	0	0	11	1
7	515	0	9	56	405	0	85	0	0
8	0	0	0	0	329	0	182	19	35
9	515	0	1	50	389	126	65	1	38
10	0	473	3	63	284	70	197	166	55
11	0	0	0	0	0	0	0	30	10
12	0	0	0	81	420	94	139	28	38
13	0	0	0	64	5	65	24	61	17
14	0	0	1	175	205	87	162	9	49
15	0	0	0	0	0	7	24	8	42
16	0	0	0	0	7	231	35	14	16
17	0	0	0	0	0	115	36	11	12
18	0	0	0	0	0	241	16	35	9
19	0	0	0	0	3	40	52	32	23
20	0	0	0	0	494	0	69	5	11
21	0	0	0	0	442	92	18	34	6
22	0	0	0	0	463	107	16	27	5
23	0	0	0	50	0	9	8	0	7
24	0	643	294	0	0	0	1	0	3
25	0	0	459	0	0	0	0	1	8
26	0	643	141	159	6	6	5	17	1
27	0	0	0	15	349	93	141	45	23
28	0	0	564	7	0	0	3	0	1
29	0	936	0	0	0	0	0	0	0