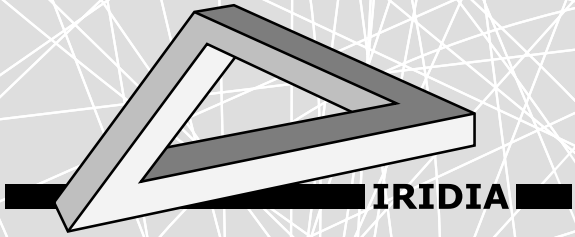

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Ant Colony Optimization under Uncertainty

Prasanna BALAPRAKASH

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2005-028

November 2005

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2005-028

Université Libre de Bruxelles
Faculté des Sciences Appliquées
IRIDIA - *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*

Ant Colony Optimization under Uncertainty

Sampling and Racing Techniques for the Probabilistic Traveling Salesman Problems

Prasanna BALAPRAKASH

Directeur de Mémoire:

Prof. Marco DORIGO

Co-promoteur de Mémoire:

Dr. Mauro BIRATTARI

Mémoire présentée en vue de l'obtention du Diplôme d'Etudes Approfondies en Sciences Appliquées

Année Académique 2004/2005

Abstract

The goal of the thesis is to study the behavior of the ant colony optimization metaheuristic under uncertain conditions. We choose the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) as a test-bed amongst stochastic optimization problems, in much the same way as the TRAVELING SALESMAN PROBLEM (TSP) has been considered a standard amongst deterministic optimization problems. Given a set of cities and the cost of travel between each pair of them, the goal of the TSP is to find the cheapest way of visiting all of the cities and returning to the starting point. This is a computationally hard problem to solve and called as $\mathcal{NP}$-*hard* problem. Ant colony optimization metaheuristic is one of the successful techniques to find the nearly optimal solution for this class of problems. The PTSP is a variant of TSP in which each city only needs to be visited with certain probability. To solve this problem, one first decides upon the order in which the cities are to be visited: the '*a priori*' tour. Subsequently, it is revealed which cities need to be visited, and those which need to be skipped, generating in this way an '*a posteriori tour*'. The objective is to choose an *a priori* tour which minimizes the expected length of the *a posteriori* tour. This is again a $\mathcal{NP}$-*hard* problem.

In this thesis, we present a new approach to compute a good solution for PTSP based on sampling and racing techniques. The accuracy of our approach is experimentally evaluated and compared against the standard ant colony optimization metaheuristic.

எனது தமிழ், தாய், தந்தை மற்றும் தமக்கைகளுக்கு . . . .

Dedicated to my beloved parents and sisters.....

Statement

This research work has been supported by COMP^{2}SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme under contract number MEST-CT-2004-505079.

The information provided is the sole responsibility of the author and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

This work is an original research and has been done in collaboration with the researchers of the COMP^{2}SYS project. A part of this work has already been published in Sixth Metaheuristics International Conference 2005. (Birattari et al. [19]).

The source code of pACS algorithm is developed by Bianchi et al. [13]. It is made available by the authors for the comparative analysis and we would like to express our sincere thanks to them.

Figure 3.1 presented in Chapter 3 is taken from Birattari et al. [18] and Birattari [17]. It is used in this thesis with kind permission of the author.

Acknowledgments

I wish to express my gratitude to my supervisor, Prof. Marco Dorigo, for the opportunity to work in an environment where I had all the necessary and even more. His confidence in me, his patience and his precision have been essential factors to the success of this research. In this context my gratitude goes also to Prof. Hugues Bersini, the director of IRIDIA.

Special thanks to my co-supervisor Dr. Mauro Birattari, for his constant encouragement with brainstorming sessions, his great help in clarifying and synthesizing the results of this work, and the fruitful collaboration we had throughout the project. Without his priceless advice, I would not have been able to reach this point.

I'm in debt with people in IRIDIA who always created the atmosphere that was both scientifically productive and socially relaxed: Carlotta Piscopo, Elio Tuci, Christos Ampatzis, Alexandre Campo, Anders Christensen, Roderich Groß, Thomas Halva Labella, Max Manfrin, Colin Molter, Shervin Nouyan, Rehan O'Grady, Christophe Philemotte, Utku Salihoglu, Pierre Sener, Krzysztof Socha, Vito Trianni, Paola Pellegrini, Bruno Marchal, Fransisco Santos and Tom Lenaerts.

I wish to express, my sincerest gratitude to my father Balaprakash, mother Sakunthala who gave me everything in life and always supported my studies. I am very much indebted to my beloved sisters Siva Sankari and Deepa Aishwarya for their love, affection and care. I would also like to thank my friends Karthik, Lakshmi and Jagadesh for their support and help during these past few years.

Last but not least, I wish to thank Arthi who always prays for my success though she is not with me.

This work was supported by the COMP^{2}SYS, a Marie Curie Early Stage Research Training Site funded by the the European Commission.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Science and technology have changed every aspect of life and society and provided significant benefits to the day to day life. They have also created new issues for the society due to their application and the level of sophistication they have brought. In many cases these scientific and technological advances hold the advantage of positive applications for the benefit of humankind. As long as there have been people, there has been technology. Indeed, the techniques of creating and shaping tools from stones and wood to hunt animals are taken as the chief evidence of the beginning of human culture. On the whole, technology has been a powerful force in the development of civilization and has a very close link with science. Technology, like language, rituals, commerce, and arts, is an intrinsic part of a cultural system and it both shapes and reflects the system's values. In today's world, technology is a complex social network that includes not only research, design, and crafts but also finance, manufacturing, management, labor, marketing, and maintenance.

In the broadest sense, technology extends our abilities to change the world: to cut, shape, or put together materials; to move things from one place to another; to reach farther with our hands, voices, and senses. We use technology to change the world to suit us better. The changes may relate to survival needs such as food, shelter, or defense, or they may relate to human aspirations such as knowledge, art, or control. But the results of changing the world are often complex, uncertain and not completely predictable. They can include unexpected benefits, unexpected costs, and unexpected risks any of which may fall on different social groups at different times.

> *By 'uncertain' knowledge, let me explain, I do not mean merely to distinguish what is known for certain from what is only probable. The game of roulette is not subject, in this sense, to uncertainty...The sense in which I am using the term is that in which the prospect of a European war is uncertain, or the price of copper and the rate of interest twenty years hence...About these matters there is no scientific basis on which to form any calculable probability whatever. We simply do not know. - (J.M. Keynes, 1883-1946, a revolutionary economist)*

Anticipating, modelling and understanding the effects of uncertainty is therefore important for exploiting the benefits and capabilities of the technology. In the ever changing world, the only certainty is uncertainty. Reasoning based on probability and statistics gives modern societies the ability to cope with uncertainty. It has astonishing power to improve decision-making accuracy and to test new ideas.

The last two decades have seen numerous advancement in humans ability to solve large-scale problems with computers. Though a number of factors led to this impressive progress, the most important one is the advancement in the hardware and software technology. The computing power has been growing exponentially in the last decade in terms of processor speed and memory.

This increase in the power of hardware has subsequently facilitated the development of increasingly sophisticated software for large scale problems. Combinatorial optimization problems are a prominent class of such large scale problems. They are conceptually easy to model and challenging to solve in practice. Due to the importance of combinatorial optimization problems for the scientific as well as the industrial world, the number of researchers in this field is growing day by day. Further, the concept of cluster and parallel computing has allowed researchers to gain more in the optimization efforts.

While much research work has been made in finding exact solutions to some combinatorial optimization problems, using techniques such as dynamic programming, cutting planes, and branch and cut methods, many hard combinatorial problems are yet to be solved exactly and require good heuristic methods. In practice we are often solving models that are approximate representation of reality: reaching "optimal solutions" in many cases doesn't have any meaning. The goal of heuristic methods for combinatorial optimization is to quickly produce good-quality solutions, without necessarily providing any guarantee of their optimality. Metaheuristics are high level procedures that coordinate simple heuristics, such as local search, to find solutions that are of better quality than those found by the simple heuristics without any such procedure. The term *metaheuristics* refers to the class of algorithms that can find good enough solutions in a reasonably short computational time and limited resources. Modern metaheuristics include simulated annealing [65, 23], genetic algorithms [57], tabu search [47], GRASP (greedy randomized adaptive search procedure) [40, 41], ant colony optimization [31, 32], variable neighborhood search [54], and their hybrids. In many practical problems they have proved to be effective and efficient approaches, being portable and adaptable to accommodate variations in problem structure and in the objectives considered for the evaluation of solutions. For all these reasons, metaheuristics have probably been one of the most promising research topics in optimization for the last two decades.

The need for efficient algorithms became evident as people attempted to solve large instances of complex problems on computers. This spawned extensive research in several fields of computer science; the problems in these fields experienced world-wide popularity not only because they have an enormous number of applications in computer, communications, and industrial engineering, but also because they have supplied an ideal proving ground for new algorithmic techniques. Nature-inspired computing is the set of computing techniques that are inspired by natural process. The remarkable growth of computing power over the last decades have made the computer a fantastic tool to cope with complexity. The emergence of nature inspired computing is one of the most amazing achievements of the researches. Ant colony optimization [31, 32] inspired by foraging behavior of ants, genetic algorithms [57] inspired by biology, simulated annealing [65, 23] inspired by physics are some of the well known computational techniques inspired by nature.

This thesis addresses a specific family of combinatorial optimization problems which includes probabilistic elements in the problem definition to describe the uncertainty associated with the problem itself. For the applications such as strategic planning for collection and distribution services, communication and transportation systems, job scheduling, organizational structures etc., the probabilistic nature of the models are very attractive as a abstraction of real world systems. Another peculiarity of the thesis is the use of nature inspired computation, ant colony optimization metaheuristic to attack the combinatorial optimization problems under uncertainty.

Goals of the Thesis

The TRAVELING SALESMAN PROBLEM (TSP) is a basic version of a combinatorial optimization problems. Ant colony optimization metaheuristics, a nature inspired computational technique, is one of the successfully applied techniques to find a good enough solution for the TSP in a reasonably short computational time. The PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) is a variant of the TSP under uncertain conditions. The first goal of the thesis is to analyze

the possibility of tackling the PTSP by treating it as TSP with ant colony optimization. More precisely, we experimentally evaluate the influence of the probabilistic nature of the problems on quality of the solutions found by ant colony optimization algorithms. The motivation behind this goal is to study portability, adaptability and robustness of the optimal solution of the deterministic case for the probabilistic scenario. The second goal is to develop an ant colony optimization algorithm based on probabilistic and statistical techniques to find a good solution to the PTSP. The motivation behind the design of new algorithm is to compute a better quality solution to the problems in which the influence of randomness is of major importance.

Original Contributions

The original contributions are:

- In previous research works (Birattari [17], Birattari et al. [18]), the F-Race algorithm is successfully employed to tune the metaheuristic algorithms. In this thesis, we show that the F-Race algorithm can be profitably combined with ACO for developing an algorithm that finds good quality solutions to the combinatorial optimization problems under uncertainty. ACO/F-Race algorithm for optimization under uncertainty (Birattari et al. [19]) has been presented at the Sixth Metaheuristics International Conference 2005.
- pACS [13], the state-of-the-art ACO algorithm designed to solve the PTSP, is based on *mathematical approximations* whereas the proposed ACO/F-Race approach is based on *empirical estimation*. This thesis describes an experimental evaluation method to compare the empirical estimation against mathematical approximation techniques for the PTSP.
- Local search methods are used in general to improve the solution found by ACO algorithms. The state-of-the-art local search methods for the PTSP are based on mathematical approximation [8, 12]. This thesis describes a local search for the PTSP which is based on empirical estimation. This thesis also proposes a comparative analysis of these two approaches.

Structure of the Thesis

The rest of the thesis is organized as follows. Chapter 2 is divided into two sections and will present the background required to understand the entire thesis. The first section defines the problem, its formal notation and informal explanation to understand the complexity and the nature of the problem which is going to be tackled by this thesis. The second section of this chapter will explain the state-of-the-art methodologies to solve the problem under consideration. Ant colony optimization is described in detail, whereas other methodologies will be briefly explained for the sake of completeness. Chapter 3 discusses the proposed approach, the original element of the thesis. This chapter is also divided into two sections where the first section presents the algorithm and the second section explains the local search for the proposed approach. Chapter 4 presents some computational experiments and results. The experimental framework and the criterions for analyzing the results of the proposed approach are described in the first section of this chapter. The second section presents the results with interpretation. Chapter 5 concludes the thesis with a brief summary followed by some suggestions to improve the proposed approach and some directions for the future research work.

Chapter 2

Background and State of the Art

This chapter is composed of two parts: Section 2.1 provides the background knowledge about combinatorial optimization problems. In specific, we describe the probabilistic traveling salesman problem, its formulation, complexity and literature review in subsection 2.1.1. Section 2.2 describes the state-of-the-art solution techniques for the probabilistic traveling salesman problem. We present them in two subsections. Subsection 2.2.1 describes metaheuristics. In particular, we focus on the ant colony optimization metaheuristic and on its stochastic variants with higher importance. Subsection 2.2.2 gives a bird's eye view of the popular exact solution techniques.

2.1 Problem Definition

Combinatorial optimization (CO) problems describe the optimal allocation of limited resources to meet desired objectives when the values of some or all of the resources are restricted by constraints. Constraints on basic resources, such as labor, cost, energy, or distance, restrict the possible alternatives that are considered feasible. The versatility of the CO model stems from the fact that in many practical problems, activities and resources, such as machines, airplanes and people, are indivisible. Also, many problems have only a finite number of alternative choices and consequently they can appropriately be formulated as combinatorial optimization problems. Combinatorial optimization models are used in planning where some or all of the decisions can take on only a finite number of alternative possibilities. In most such problems, there are many possible alternatives to consider and one overall goal determines which of these alternatives is best. For example, an airline company needs to determine crew schedules which minimize the total operating cost; as another example, consider the design of flexible production plant in a manufacturing industry to handle the dynamic needs of the market. Therefore, in day-to-day's changing and competitive industrial world, the difference between using a quickly derived solution and using sophisticated mathematical models to find an optimal solution can determine the success and failure of the enterprizes.

CO is a process of finding one or more optimal solutions in a well defined problem solution space. Such problems occur in almost all fields of management (e.g. commerce, production, scheduling, inventory control and layout), as well as in many engineering disciplines (e.g. optimal design of waterways or bridges, VLSI-circuitry design and testing, the layout of circuits to minimize the area dedicated to wires, design and analysis of data networks, logistics of electrical power generation and transport, the scheduling of lines in flexible manufacturing facilities).

Computational Complexity

Computational complexity (or complexity theory) is a central subfield of the theoretical foundations of computer science. It is concerned with the study of the intrinsic complexity of computational tasks. This study tends to aim at generality; it focuses on computational time and considers

the effect of limiting it on the class of problems that can be solved. It also tends to asymptotic: studying the complexity as the size of data grows. Another related subfield deals with the design and analysis of algorithms for specific (classes of) computational problems that arise in a variety of areas of mathematics, science and engineering. In general, computational complexity studies:

- the efficiency of algorithms
- the inherent "difficulty" of problems of practical and/or theoretical importance

A decision problem is a problem that takes an input and requires either YES or NO as output. If there is an algorithm which is able to produce the correct answer for any input of length n in at most n^k steps, where k is some constant independent of the input, then the problem can be solved in *polynomial time* and are grouped as $\mathcal{P}$ class.

Now consider an algorithm A(w,C) which takes two arguments: input w of length n to the decision problem, and another input C which is an information required to verify a positive answer, such that A produces a YES/NO answer in at most n^k steps. Then we say that the problem can be solved in *non-deterministic polynomial time* and are called as $\mathcal{NP}$ class. For example, we might ask whether 69799 is a multiple of any integers between 1 and 250. The answer is YES, though it would take a fair amount of computational time. On the other hand, if someone claims that the answer is YES because 223 is a divisor of 69799, then we can quickly check that with a single division. Verifying that a number is a divisor is much easier than finding the divisor in the first place.

$\mathcal{NP}$-*complete* problems are the most difficult problems in $\mathcal{NP}$, in the sense that they are the ones most likely not to be in $\mathcal{P}$. A decision problem is $\mathcal{NP}$-*complete* if it is in $\mathcal{NP}$ and every other problem in $\mathcal{NP}$ is reducible to it. The reduction here refers to the transformation of one problem into another problem in a polynomial time. One example of an $\mathcal{NP}$-*complete* problem is the *subset sum* problem which can be stated in the following way: given a finite set of integers, determine whether any non-empty subset of them sums to zero. A supposed answer is very easy to verify for correctness, but no one knows a significantly faster way to solve the problem than to try every single possible subset, which is computationally expensive.

The term $\mathcal{NP}$-*hard* refers to any problem that is at least as hard as any problem in $\mathcal{NP}$. Thus, the $\mathcal{NP}$-*complete* problems are precisely the intersection of the class of $\mathcal{NP}$-*hard* problems with the class $\mathcal{NP}$. Any problem that involves the identification of an optimal solution from a well defined large solution space is known as an optimization problem. In computational complexity, optimization problems whose decision versions are $\mathcal{NP}$-*complete* are $\mathcal{NP}$-*hard*, since solving the optimization version is at least as hard as solving the decision version. A substantial treatment of this topic was presented in [45].

Formal Definition

According to Papadimitriou and Steiglitz [80], a CO problem $\mathcal{P} = (\mathcal{S}, f)$ is an optimization problem in which a finite set of solutions and the search space $\mathcal{S}$ are given along with an objective function $f : \mathcal{S} \rightarrow \Re^+$ that assigns a positive cost to each solution s in the search space $\mathcal{S}$. The goal is to find a solution of minimal cost.[1] The practical importance of CO problems attracted many researchers over time and led to the development of different algorithms to tackle them. It is interesting to note that all these algorithms falls into one of the two classes:

- *Complete* algorithms : Complete algorithms are guaranteed to find for every finite size instance of a CO problem a minimum cost value [80, 77].

[1]Note that minimizing over an objective function f is the same as maximizing over $-f$. Therefore, every CO problem can be described as a minimization problem.

- *Approximate* algorithms: However, for CO problems that are $\mathcal{NP}$-*hard*, no polynomial time algorithm exists, assuming that $\mathcal{P} \neq \mathcal{NP}$ [45]. The amount of time that an exact algorithm takes is likely to be so long that it might be too long for all practical purposes. The approximate algorithms find good enough solution in a reasonable amount of time. The key idea behind this technique is to sacrifice the quality of the solution for a tractable running time.

2.1.1 Probabilistic Traveling Salesman Problem

We adopt the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) as a test-bed amongst stochastic optimization problems, in much the same way as the TRAVELING SALESMAN PROBLEM (TSP) has been considered a standard amongst deterministic optimization problems. PTSP is the most fundamental stochastic routing problem that is available in the literature [60]. Given a set of cities and the cost of travel between each pair of them, the goal of the TSP is to find the cheapest way of visiting all of the cities and returning to the starting point. However, it is certainly not obvious how to use this data to plan the cheapest route. PTSP, as the name reveals, is a variant of TSP to optimization in the face of unknown data. Whereas all of the cities in the TSP must be visited once and only once, in the PTSP each city only needs to be visited with some probability. For example, consider a PTSP through a set of n cities. On any given scenario of the problem only k out of n cities have to be visited. This can be denoted by the subset S where $\mid S \mid = k$. (i.e) on any given day, the salesman may have to visit only a subset S that contains k cities. Therefore, there exist 2^n possible scenarios for the problem.[2]

The most obvious approach in dealing with such cases is to attempt to solve optimally or re-optimize every potential scenarios of the PTSP. Though re-optimization technique is trivial, it aims to solve exponentially many scenarios of a $\mathcal{NP}$-*hard* problem. Moreover, in many applications it is necessary to find a solution to each new scenario quickly but without extra computation and extra resources. A well known approach to tackle this situation is *a priori optimization* or *skipping strategy* introduced by Jaillet [61]. In PTSP, a tour which visits all the cities is called as an *a priori* tour. The $\mid S \mid = k$ cities of the PTSP will be visited in the same order as they appear in the *a priori* tour by skipping the cities that are not a part of S: constructing in this way the *a posteriori* tour. Therefore, the problem of finding an *a priori* tour which minimizes the expected value of the *a posteriori* tour length is defined as PTSP [62]. The expected value is computed over all possible scenarios S_1, S_2 ... S_{2^n}. Figure 2.1 shows one of such possible *a priori tour* of the PTSP. The *a posteriori* tour follows the *a priori* tour by skipping the cites that are not a part of current scenario is shown in Figure 2.2.

From the *a priori optimization* perspective, the formal notation of PTSP can be derived as:

- $G = (N, A)$, complete weighted graph
- N, the set of nodes in G representing cities
- p_i, the probability associated with the node $i \in N$
- A, the set of arcs connecting the nodes in N
- d_{ij}, the weight associated with arcs, that is, the distance between cities $i, j \in N$[3]
- S_1, S_2 ... S_{2^n}, the set of all possible scenarios.

[2]For example consider 2 cities n_1 n_2 PTSP. The set of possible scenarios is {(),(n_1),(n_2),(n_1,n_2) } that has 2^2 cardinality.

[3]For symmetric PTSP d_{ij}=d_{ij} but in asymmetric PTSP, at least one arc (i, j) depend on the direction (i.e) $d_{ij} \neq d_{ij}$

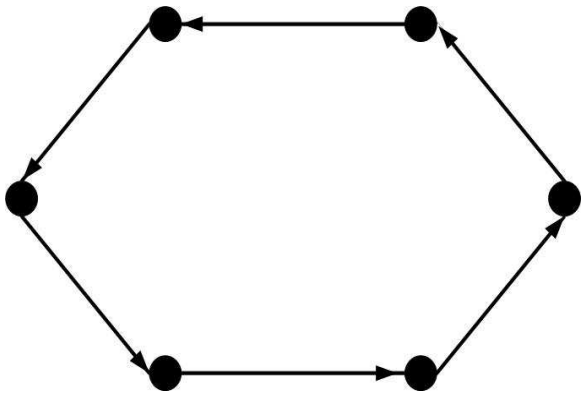

Figure 2.1: First step in the *a priori* optimization: The *a priori* tour of a PTSP instance (which contains six cites) that visits all the cities once and only once.

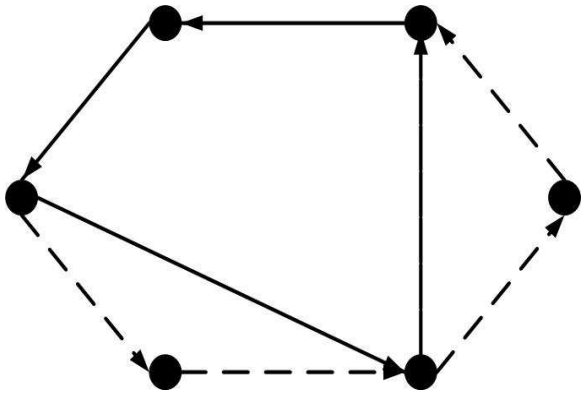

Figure 2.2: Second step in the *a priori* optimization: The *a posteriori* tour (thick line) in which cities of the PTSP will be visited in the same order as they appear in the *a priori* tour (dotted line) by skipping the cities that are not a part of the random scenario S

The goal is to find the *a priori* tour of the graph that minimizes the expected value of the *a posteriori* tour length. The most recent and comprehensive literature review about the PTSP and its solution techniques is presented by Campbell [21] where she describes the aggregation approach to solve PTSP. The rest of this section summarizes the history of PTSP and its solution techniques from [21].

In 1985, Patrick Jaillet first made an exhaustive study of the PTSP in his dissertation [60]. His work attested interesting properties of optimal PTSP tours. Later in [61], he provided a formulation for the expected value of the *a posteriori* tour and established the relationship between optimal PTSP and TSP solutions. Most of the Jaillets results discussed PTSP with homogeneous probabilities.

Berman and Simchi-Levi [6] studied instances of the PTSP with heterogeneous probabilities. They established the lower bound for the PTSP instances. They also explained the potentialities of using a branch-and-bound algorithm to find an optimal *a priori* tour but without any computational results. Therefore, it is hard to observe quantitative relationship between the solution and the instance. Furthermore, the nature of the proposed approach paralyzes its application and generality to large problem instances.

For what concerns the thesis, it is important to observe the work of Rossi and Gavioli [86]. They proposed an approximation algorithm based on nearest neighbor techniques. The expected costs of the resulting solutions are compared with those found using the basic TSP approximation algorithms. The results of computational experiments conclude that it is necessary to employ techniques specifically developed for the PTSP if the number of cities is greater than 50 and the probability of each city requiring a visit is less than 60%.

Bertsimas focussed on a series of other probabilistic combinatorial optimization problems, such as the probabilistic minimum spanning tree and vehicle routing problems in his dissertation [8] and related papers [7, 9, 10]. Inspired by Jaillets work, he generalized the computational techniques of the approximation algorithms for the deterministic problems to the probabilistic context, including the spacefilling curve approach for the TSP to the PTSP.

Bertsimas and Howell [9] investigated the potentialities of applying TSP approximation algorithm for solving the PTSP. They proposed an algorithm based on spacefilling curve approach [4] to construct an initial solution and local search to improve the computed solution. The *2-opt* and *1-shift* techniques developed for the TSP in [68] are extended to compute the change in the objective function in an expected value sense. Notwithstanding some approximation errors [15] in the generalization, improvement based on expected value becomes more significant as the size of the problem becomes large. They also found that expected value based local improvement is particularly important when the probability values associated with the cities are *significantly* less than 1 as proposed by Rossi and Gavoli [86]

The sampling approach for solving PTSP was proposed by Bertsimas et al. [11]. They compared the objective values obtained from the *a priori* tour and *sampling* techniques. While the former is computed from the TSP approximate algorithm and improved with local search, the later is derived by averaging the results obtained from a sampling of realizations. The experimental results show that there is no significant difference in the solution quality but the computational time of the *a priori* approach is lesser than sampling techniques.

When instances of larger sizes must be optimized, or a large sequence of instances must be optimized, exact algorithms are no longer practical. The next best choice, then, is to use approximate methods that find near-optimal tours quickly. These are known as heuristic algorithms. Metaheuristic algorithms [46] guide the heuristic algorithms to find high quality near optimal solutions

in a reasonable computational time. Evolutionary algorithm [38], stochastic annealing approach [53] , and ant colony metaheuristics [13] [52] provide a robust and effective optimization techniques to solve PTSP.

The PTSP has a wide range of applications because it is close to some important real world problems such as routing problems like dynamic vehicle routing with stochastic demands [14], communication network and protocol design [3].

Cost Function Estimation

Optimization problems require finding the minimum or maximum (depending on the problem) of a mathematical expression known as the objective function. The objective function involves a number of variables. The goal of optimization is to find values for these constituent variables that minimize or maximize the objective function. In the context of cost minimization and revenue maximization, it is known as cost function and value function respectively. Robbins and Munro [85] gave the first formal idea of optimizing the stochastic problems by stochastic approximation. The formal description of the cost function of the CO problems under uncertainty can be represented as follows:

- s, a solution in the set of feasible solution $\mathcal{S}$
- ω which describes the influence of the uncertainty associated with the solution s
- $f(s, \omega)$, the cost function which depends on s and ω
- E, the mathematical expectation
- The goal is, Minimize $F(s) = \mathrm{E}[f(s, \omega)]$ subject to $s \in \mathcal{S}$

In the context of PTSP, a feasible solution s represents an *a priori* tour visiting once and only once all the cities. The set $\mathcal{S}$ contains all the feasible *a priori* tours. The variable ω determines the cities to be included in the tour. The function $f(s, \omega)$ is the tour length of the *a posteriori* tour. The goal of the optimal solution in PTSP is to choose an *a priori* tour from the set $\mathcal{S}$ which minimizes the expected value of the *a posteriori* tour length $f(s, \omega)$.

One attempt to solve the PTSP using an exact method was taken by Laporte et al. [66] who introduced the use of integer stochastic programming. This study was severely limited in the size of problem attempted and the stochastic programming algorithm failed to solve the PTSP on certain occasions. Thus the quality of the results is not very high. On the other hand, the empirical estimation approach is employed to solve PTSP. The main advantage of the estimation approach over the one based on *mathematical approximation* is generality: Indeed, a sample estimate of the expected cost of a given solution can be simply obtained by averaging a number of realizations of the cost itself. Further, computing a profitable approximation is a problem-specific issue and requires a deep understanding of the underlying probabilistic model. In the empirical estimation approach to stochastic combinatorial optimization, the expectation $F(s)$ of the cost $f(s, \omega)$ for a given solution s is estimated on the basis of a sample $f(s, \omega_1)$, $f(s, \omega_2)$,..., $f(s, \omega_N)$ obtained from N independently extracted realizations of the random variable ω:

$$\varepsilon(F(s)) = \frac{1}{N} \sum_{v=1}^{N} f(s, \omega_v) \tag{2.1}$$

Clearly, $\varepsilon(F(s))$ is an unbiased estimator of $F(s)$.

2.2 State-of-the-Art Approaches

Due to the economical importance and scientific challenge of the PTSP, several algorithms were devised for their solution. The choice of the exact or approximate algorithms is decided by the practical problems and their time constraints. In the following sections we describe both of them briefly but a substantial treatment is given to ant colony optimization metaheuristics which serves as a background for the approach described in Chapter 3.

2.2.1 Metaheuristic Approaches

The field of metaheuristics for the application to combinatorial optimization problems is an interesting and rapidly growing field of research. This is due to the importance of combinatorial optimization problems for the scientific as well as the industrial world. The term *metaheuristic* is derived from two Greek words. *Heuristic*, which derives from the verb *heurísco* ($\epsilon\upsilon\rho\acute{\iota}\sigma\kappa\omega$) which means "to search", and *meta* ($\mu\epsilon\tau\alpha$), stands for "beyond, on a higher level". The term metaheuristic, first proposed by Glover [46], has been used in the literature with different meanings. Only in the last years some researcher have proposed a general definition [79, 95]. We can cite for example the one given by Stützle [90]:

> *Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid the disadvantages of iterative improvement and, in particular, multiple descent by allowing the local search to escape from local optima. This is achieved by either allowing worsening moves or generating new starting solutions for the local search in a more "intelligent" way than just providing random initial solutions. Many of the methods can be interpreted as introducing a bias such that high quality solutions are produced quickly. This bias can be of various forms and can be cast as descent bias (based on the objective function), memory bias (based on previously made decisions) or experience bias (based on prior performance). Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent, biased form.*

A simple and precise definition is given by *Metaheuristics Network*[4] which states that:

> *A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem.*

Blum and Roli [20] presented a series of characteristic properties of metaheuristics. We summarize them as follows:

- Metaheuristics are strategies that "guide" the search process. Their goal is to efficiently explore the search space in order to find (near-)optimal solutions.
- Metaheuristics may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics can be described on an abstract level (i.e., not tied to a specific problem).
- Metaheuristics often use the experience gained in previous searches (memory) to guide new searches.

[4] Metaheuristics Network was a European Union research project whose main scientific goal is to improve the understanding of metaheuristics work through theoretical and experimental research. www.metaheuristics.org

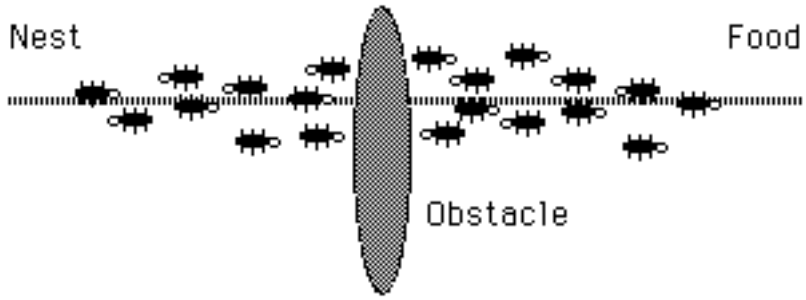

Figure 2.3: The main means used by ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone rather than a poorer one. An obstacle in the ant's path which leads to two paths, a longer and a shorter between nest and food.

- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy. Those strategies must be chosen in such a way to balance dynamically the exploitation of previously gained experience, called **intensification**, and the exploration of the search space, called **diversification**. This balancement is necessary, on one side, to quickly identify region in the search space where good solutions are, on the other side, to not loose to much time in searching inside regions that have already been explored or that seem not to have good solutions.

Ant Colony Optimization

Ant colony optimization(ACO) is one of the latest metaheuristic developed to tackle CO problems. ACO technique was introduced by Dorigo [31]. Henceforth, the number of applications and researchers using this methodology are increasing day by day. In this section we present the description of ACO framework given by Dorigo and Caro [32].

The computational model of ACO was inspired by the foraging behavior of ants. This behavior as described by Deneubourg et al. [30] enables ants to find shortest paths between food sources and their nest. As soon as an ant finds a source of food, the former evaluates the source quantity and quality and carries some food sample to the nest. While returning back, the ant deposits a chemical substance called *pheromone* on the ground. This pheromone serves to attract other ants to follow the same path. Clearly, the ants taking shorter path will return back to the nest sooner than the ants taking the longer path. Henceforth the concentration of the pheromone on the shorter path increases faster than the longer paths. The higher the pheromone intensity, the higher will be the probability that the following ants take the respective (shorter) path. Although leaving the pheromone trails by an ant on the path while returning to the nest seems to be a primitive behavior, a colony of ants engaging in this primitive behavior will emerge as a source of collective intelligence [36]. This intelligent behavior of ant colonies is the inspiration for artificial ant colonies which are developed to tackle CO problems.

For illustration, let us consider Figure 2.3 that sketches an obstacle in the ant's path. Obviously, it forms two paths, a longer and a shorter between nest and food. Figure 2.4 shows the ants will take randomly both longer and shorter paths. Clearly, the pheromone intensity increases more in the shorter path. As a consequence, after some time all the ants will take the shorter path which is shown in Figure 2.5.

ACO algorithm incorporates artificial ants that follow the artificial pheromone trails represented by a parameterized probabilistic model termed as the *pheromone model*. The pheromone model

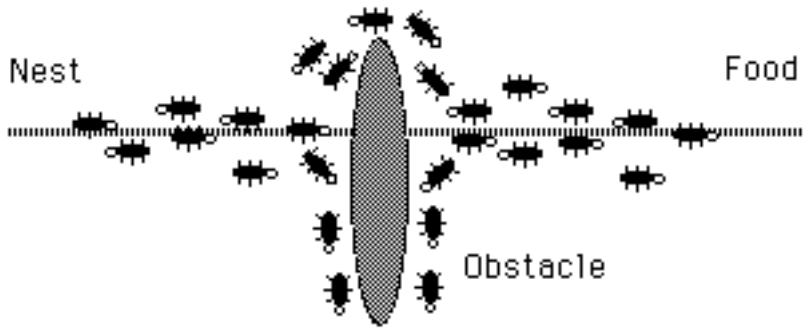

Figure 2.4: Ants which are just in front of the obstacle cannot continue to follow the pheromone trail and therefore they have to choose between turning right or left. Therefore, they randomly choose both longer and shorter paths. The pheromone intensity starts to increase more in the shorter path

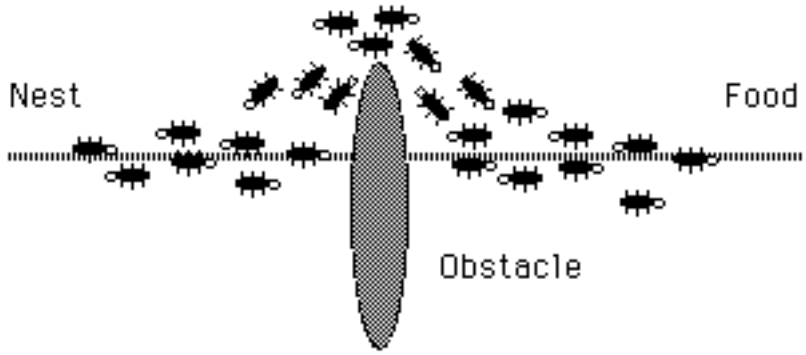

Figure 2.5: Due to the positive feedback process, very soon all the ants will choose the shorter path which contains higher pheromone intensity than the longer one.

consists of a set of model parameters whose values are called the *pheromone values*. The interesting element of the ACO algorithm is the probabilistic construction of solutions using the pheromone values. The motivations behind this solution construction techniques are:

- To generate a solution using the pheromone model from a large solution space which contains solutions of different quality.
- To narrow the search towards the high quality solutions in the solution space by updating the pheromone values with the solutions that were constructed in earlier iterations.

More in general, ACO is a constructive technique that aims to improve the solution after each iteration. The framework of the ACO metaheuristic for CO presented by Dorigo and Stützle [34] is shown in Algorithm 1.

Algorithm 1 Ant Colony Optimization Framework

```
input: an instance x of a CO problem
while termination conditions not met do
  ScheduleActivities
    SolutionConstructionWithAnts()
    PheromoneUpdate()
    DaemonActions()
  end ScheduleActivities
  s_best ← best solution in the population of solutions
end while
output: s_best, "candidate" to optimal solution for x
```

Clearly, the three key algorithmic components are grouped under the **ScheduleActivities** construct. This framework also provides more flexibility to the designer with respect to scheduling and synchronization. More precisely, there is no restriction for parallel and independent or synchronization methodology for the execution.

`SolutionConstructionWithAnts`(): ACO, as a constructive heuristic, assembles solutions as sequences of solution components taken from a finite set $\mathcal{C} = \{c_1, \cdots, c_n\}$. For the TSP these solution components are referred as cities. A starting point in the solution construction step is an empty partial solution $s^p =<>$. Now, we will describe the solution construction phase with the first ACO algorithm called Ant System [35],[31]. Initially, m artificial ants are placed on randomly chosen cities. At each construction step, each ant constructs the solution by using a probabilistic action choice rule called *random proportional rule*, to decide which city to visit next. More precisely, the probability with which ant k, currently at city i, chooses to go to city j is given by Equation:

$$\mathbf{p_{ij}^k} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \text{if } j \in N_i^k \tag{2.2}$$

where, η_{ij} is known as the the *heuristic information* which is inversely proportional to the distance d_{ij} of current city i and the next city j and τ_{ij} refers to *pheromone trails* of the biological metaphor that judge the the desirability of visiting city j from city i. Dorigo and Stützle [34] proposed to set $\eta_{ij} = 1/d_{ij}$. The solution construction mechanism, at each construction step, add a feasible solution component from the set $N_i^k \subseteq \mathcal{C} \setminus s^p$ to the current partial solution s^p. Intuitively, N_i^k is the feasible neighborhood of ant k when being at city i, that is, the set of cities that ant k has not visited yet. It is interesting to note the dependency of the solution construction with respect to positive parameters α and β. The search is intensified towards the already found solutions when the α value is higher, whereas the search is diversified in the solution space to explore the new solution with the lower counterpart. On the other hand, when the β value is higher, the solution

component from $N(i^k)$ which has the lower heuristic value (nearest neighbor) will be selected as the next solution component and viceversa.

`PheromoneUpdate`(): There are several types of pheromone updates employed in ACO algorithms. However the basic ingredient is the same. Similar to the biological metaphor, pheromone update incorporates two elements:

- *pheromone evaporation*, which uniformly decreases all the pheromone values. Henceforth this process implements a useful form of *forgetting*. As a consequence, the search process is diversified in the solution space favoring the exploration of new areas.
- *pheromone deposit*: one or more solutions from the current and/or from earlier iterations are used to increase the values of pheromone trail parameters on solution components that are part of these solutions.

The Ant System [35],[31] uses the following pheromone update rule called *AS-update*.

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\sum_{\{s\in\mathcal{G}_{iter}|c_i\in s\}} F(s),\ \forall(i,j)\in\mathcal{C} \tag{2.3}$$

where $\mathcal{G}_{iter}$ is the set of solutions that were generated in the current iteration. Furthermore, $\rho \in (0,1]$ is a parameter called evaporation rate, and $F : \mathcal{G} \rightarrow \Re^+$ is a function such that $f(s) < f(s') \Rightarrow F(s) \geq F(s'), \forall s \neq s' \in \mathcal{G}$. $F(\cdot)$ is commonly called the *quality function*. In the first work [35],[31] the update rule in the Equation 2.3 was defined without multiplying the pheromone by ρ in the second additive factor. Only afterwards (for example in [33]) this was often done. However, as ρ is a constant, it does not paralyze the qualitative behavior of the algorithm.

`DaemonActions`(): Daemon actions refers to the execution of actions which has to take place centrally. For example, the constructed solution is improved by applying local search. As a another example, the daemon may decide to deposit extra pheromone on the solution components that belong to the best solution found so far.

Several types of pheromone update procedures are available in practice which aim at the intensification or the diversification of the search process and they differ in the way they update the pheromone values. Explaining each flavor is beyond the scope of the thesis. However, it is worthwhile to mention some of the ACO variants: Ant Colony System (ACS)[33], $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [91], Elitist Ant System [35][33]. In the following sections we present the ACO algorithms which are proposed to solve the PTSP.

Explicit Objective Function based Ant Colony System

Bianchi et al. [13] first proposed the potentialities of ACO algorithms for the PTSP. This approach exploits an explicit formula for the calculation of expectation of the objective function. Let the set V denotes cities in the *a priori* tour and S be the subset of cities whose *a posteriori* tour length is $L_\lambda(S)$. We can denote $p(S)$ as the probability that the subset of cities S will require a visit. In this context, the objective function or the expected length of the *a posteriori* tour $E[L_\lambda]$ is averaged over all possible *a posteriori* tour lengths. Jaillet [60] proposed a closed form expression to compute the objective function:

$$E[L_\lambda] = \sum_{S\subseteq V} p(S)L_\lambda(S) \tag{2.4}$$

An arc (i,j) is actually used only when nodes i and j need to be visited by skipping the nodes $i+1, i+2, ..., j-1$. This event occurs with the probability given by Equation 2.5

$$p(i,j) = p_i p_j \prod_{k=i+1}^{j-1}(1-p_k) \tag{2.5}$$

Therefore,

$$p(S) = \prod_{i \epsilon S} p_i \prod_{i \epsilon V-S} (1 - p_i) \tag{2.6}$$

When the *a priori* tour is adapted by skipping a set of cities which do not require a visit, the objective function can be derived as shown in Equation 2.7 [60] in which d_{ij} denotes the distance between two cities i and j. In other words, the number of skipping nodes represented by $|i+1, i+2, ..., j-1|$ ranges from $0, 1, 2....n-2$.

$$E[L_\lambda] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1 - p_k) \prod_{l=1}^{j-1} (1 - p_l) \tag{2.7}$$

For a homogeneous PTSP where all cities have same probability, the Equation 2.7 becomes simple as shown below [13]:

$$E[L_\lambda] = p^2 \sum_{r=0}^{n-2} (1-p)^r L_\lambda^{(r)} \tag{2.8}$$

where, $L_\lambda^{(r)} = \sum_{j=1}^{n} d(j, (j+1+r) \ mod \ n)$ is the sum of the distances between each city and its $(r+1)^{th}$ following city in the *a priori* tour.

The ACS algorithm was introduced as an improved version of AS to harness a better performance. Bianchi et al. [13] developed a modified version of ant colony system (ACS) [33] called *probabilistic ant colony system* (pACS) to tackle homogeneous PTSP that takes the PTSP objective function in Equation 2.8 while selecting the best solution at each iteration. More precisely, for each ant's *a priori* tour, $E[L_\lambda]$ is computed using the Equation 2.8. The *a priori* tour with minimum $E[L_\lambda]$ and the corresponding ant is selected as the *iteration-best* solution and the *iteration-best* ant respectively for the current iteration. The interesting elements which makes the pACS differ from AS are as follows:

- A variant of the Equation 2.2 called *pseudo-random-proportional* rule is used in pACS to perform the construction steps.

$$j = \begin{cases} argmax_{l \varepsilon N_i^k} \{\tau_{il}[\eta]^\beta\} & \text{if } q \leq q_0, \\ J & \text{Otherwise} \end{cases} \tag{2.9}$$

  where q is uniformly distributed in the interval [0,1], q_0 is a parameter such that $[0 \leq q_0 \leq 1]$ and J is a random variable whose value is returned by the Equation 2.2. Therefore, with probability q_0 the ant chooses the best city according to the pheromone trail and the distance between cities, whereas with probability $1 - q_0$ exploration of search space is achieved.

- The search process during the solution construction is diversified by immediately decreasing the pheromone value τ_{ij} when the ants move from city i to city j. This is given by Equation 2.10

$$\tau_{ij} \leftarrow (1 - \xi) \cdot \tau_{ij} + \xi \cdot \tau_{ij}, \tag{2.10}$$

  where ξ, $0 < \xi < 1$, and τ_{ij} are parameters. Dorigo and Stützle [34] recommended to set $\tau_0 = 1/C^{nn}$, where n is the number of cities and C^{nn} is the length of the *nearest neighbor* tour.

- The components of the *best-so-far* solution are only allowed to update the pheromone. The same holds true concerning the pheromone evaporation. Therefore the Equation 2.3 becomes,

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_{ij},\ \forall(i,j) \in s_{best} \tag{2.11}$$

  where s_{best} represents the *iteration-best* tour whose objective function has minimum expected length.

Now we summarize the results and the significance of this approach. In the experimental section, Bianchi et al. [13] presented the comparative analysis of the pACS against ACS when applied to homogenous PTSP with respect to solution quality. The results showed that the smaller the value of probabilities associated with each city in the PTSP, the higher the solution quality of pACS compared to ACS. For the probability values closer to 1, ACS performs better. The reason for these observations is explained with respect to time complexity of the algorithms. The time complexity of one iteration in both the algorithms is $O(n^2)$ but constant of proportionality is higher in pACS. Furthermore, in each iteration the evaluation of best solution takes $O(n^2)$ in pACS because of expensive computation in objective function whereas in ACS it is $O(n)$. When p is near to 1, a good solution to the TSP is also a good solution to the PTSP. As a consequence, ACS, which can perform more iterations than pACS for a same amount of computational time, has a better solution quality. They also concluded that pACS cannot be easily generalized to heterogenous PTSP as the equation for calculating the objective function becomes complex.

Simulation based Ant Colony Optimization

The stochastic combinatorial optimization problem needs not to be essentially different from that of a deterministic problem when the objective function can be represented as an explicit mathematical expression (for example the Equation 2.8) or at least be easily computed numerically [52]. In such case, it is possible to formulate the stochastic problems as the representation of the expected objective function and to solve them with deterministic exact or approximate optimization algorithms. In most cases, it is only possible to determine the estimates of the expected objective function by means of sampling or simulation. Moreover this is the key element in the area of simulation optimization, which has been a topic of interesting and challenging research for several decades. A substantial treatment of this topic and the key features have been described by Fu [44]. In pACS [13], an explicit formula for the expectation of the objective function value is known for the PTSP, so the chosen solution technique cannot be generalized to problems where sampling is necessary to obtain estimates of this expectation. Gutjahr [52] proposed a general purpose, simulation based ACO algorithm called S-ACO for solving stochastic combinatorial optimization problems. The theoretical convergence of S-ACO to the global optimum solution has been described in [51]. Without the loss of generality S-ACO can be employed to solve PTSP. The pseudo code of the S-ACO algorithm is given in Algorithm 2.

The algorithm starts with the initialization of the pheromone values such that the pheromone values for all arcs in PTSP is set to unity. The important and interesting elements which make the difference between AS and S-ACO are the sampling/simulation and pheromone update procedures. Once the ants finished constructing the *a priori* tour with *random proportional rule*, a random scenario ω is generated according to the probability p_i associated with each city in the PTSP. Intuitively, the algorithm generates a random number x for each city in the interval [0,1] to decide a city to be included in the ω. In other words, if $p_i \leq x$ then the city is included otherwise it is excluded. Afterwards, the *a posteriori* tour length for each *a priori* tour is computed using the random scenario ω. The ant whose *a posteriori* tour length is minimum will be considered as the *iteration best* ant and the corresponding *a priori* tour is considered as the *iteration-best* solution s. Gutjahr [52] also experimented with several random scenarios to select the *iteration best* ant but this methodology increased the runtime without improving the solution quality.

Algorithm 2 Simulation based Ant Colony Optimization

input: an instance C of a PTSP problem
set $\tau_{ij} \leftarrow 1$ for all (i,j)
for round $r = 1, 2, 3.....$ **do**
  for ant $\sigma = 1, 2, 3.....m$ **do**
    place the ant in a random city i
    choose the next city by *random proportional rule* until all the cities were included in the tour
  end for
  Based on one random scenario ω select the best tour s from m ants
  if $(r = 1)$ **then**
    set $s_{best} \leftarrow s$
  else
    Based on N_m random scenarios ω_v, compute a sample estimate

$$\varepsilon(F(s) - F(s_{best})) = \frac{1}{N_m} \sum_{v=1}^{N_m} f(s, \omega_v) - f(s_{best}, \omega_v)$$

    if $\varepsilon(F(s) - F(s_{best})) < 0$ **then**
      $s_{best} \leftarrow s$;
    end if
  end if
  evaporation: set $\tau_{ij} \leftarrow 1 - \rho \forall (i, j)$;
  best-so-far pheromone update: set $\tau_{ij} \leftarrow \tau_{ij} + c_1, \forall (i, j) \in s_{best}$;
  iteration best pheromone update: set $\tau_{ij} \leftarrow \tau_{ij} + c_2, \forall (i, j) \in s$;
end for

For the first iteration, the *iteration-best* solution is also considered as the *best-so-far* solution. But for the subsequent iterations, it is not possible anymore to decide deterministically whether a *iteration-best* solution s is better than the solution currently considered as the *best so far* solution s_{best}. After the *iteration-best* solution s has been determined, it is compared with the solution considered currently as the *best-so-far* solution s_{best} using sampling/simulation technique. Intuitively, the algorithm generates N_m iteration specific scenarios[5] and for each of them it computes the *a posteriori* tour length for both s and s_{best}. Only afterwards, the solution with least average is considered as the s_{best}. The larger the N_m, the higher the precision of the decision will be. The number of random scenarios grows linearly with the number of iterations r and Gutjahr [52] proposed to set $N_m = 50 + (0.0001.n^2).r$ where n is the number of cities in the PTSP.

In contrast to AS algorithm, only the solutions s and s_{best} are allowed to update the pheromone. The parameters $c_1 > 0$ and $c_2 > 0$ in the algorithm determine the amount of pheromone increment on the *best-so-far* tour and *iteration-best* tour respectively. Experiments in [52] showed that c_2 should be chosen small compared to c_1, but a small positive c_2 produced better results than setting $c_2 = 0$.

Stochastic Simulated Annealing

Simulated annealing (SA) is one of the oldest metaheuristics and one of the first algorithms that had an explicit strategy to escape from local minima. The history of this algorithm dates back to 1953 with statistical mechanics (Metropolis algorithm [72]). The original idea of SA was inspired by physics, in specific, the annealing process of metal and glass, which assume a low energy configuration when cooled with an appropriate cooling schedule. SA was first presented as

[5]For the next iteration, new scenarios will be drawn.

a search algorithm for CO problems in Kirkpatrick et al. [65] and Cerný [23]. In order to avoid getting trapped in local minima, the fundamental idea is to allow moves to solutions with objective function values that are worse than the objective function value of the current solution. Such a kind of move is often called an *uphill-move*. For the deterministic CO, at each iteration a solution $s' \in \mathcal{N}(s)$ is randomly chosen. If s' is better than s (i.e., has a lower objective function value), then s' is accepted as new current solution. For the stochastic CO problems, the evaluation of the objective function is done by sampling [53]. In other words, accepting a best solution needs the evaluation of N_m random scenarios. Otherwise, if the move from s to s' is an uphill-move, s' is accepted with a probability which is a function of a temperature parameter T_k and $f(s') - f(s)$. Usually this probability is computed by following the Boltzmann distribution:

$$p(s'|T_k, s) = e^{-\frac{f(s')-f(s)}{T_k}}. \tag{2.12}$$

The search process described by SA is a *markov process* [37], as it follows a trajectory in the state space in which the next state is chosen depending only on the current state. Therefore in principle, SA is memory-less. However, the use of memory can be beneficial for SA approaches (as proposed in Chardaire et al. [24]). The algorithmic framework of stochastic SA is described in Algorithm 3.

Algorithm 3 Stochastic Simulated Annealing

input: an instance C of a PTSP problem
$s \leftarrow$ `GenerateInitialSolution`()
$k \leftarrow 0$
$T_k \leftarrow$ `SetInitialTemperature`()
while termination conditions not met **do**
  $s' \leftarrow$ `PickNeighborAtRandom`($\mathcal{N}(x)$)
  Based on N_m random scenarios ω_v, compute a sample estimate

$$\varepsilon(F(s) - F(s')) = \frac{1}{N_m} \sum_{v=1}^{N_m} f(s, \omega_v) - f(s', \omega_v)$$

  if $\varepsilon(F(s) - F(s')) < 0$ **then**
    $s \leftarrow s'$;
  else
    accept s' as new solution with probability $p(s'|T_k, s)$
  end if
  `AdaptTemperature`(T_k)
end while
$s_{best} \leftarrow s$
output: s_{best}, "candidate" to optimal solution for C

`GenerateInitialSolution`(): The algorithm starts with an initial solution which is generated in a random or a heuristic way.

`SetInitialTemperature`(): The initial value of the temperature is set in such a way that it favors uphill-move at the start of the algorithm.

`AdaptTemperature`(T_k): The temperature T_k is changed at each iteration according to a *cooling scheme*. The cooling scheme determines the value of T_k at each iteration k. The choice of an appropriate cooling scheme plays an important role in the performance of the algorithm. The probability of accepting the worsening solutions should be high during the initial iterations to favor the uphill-moves. Afterwards, this probability should be gradually decreased during the search.

Aarts et al. [1] verified the theoretical results with non-homogeneous *markov chains* which states that under certain conditions on the cooling schedule, the algorithm converges in probability to

a global minimum for $k \to \infty$. A particular cooling scheme that fulfills the hypothesis for the convergence to guarantee the optimal solution is the one that follows a logarithmic law. Therefore, T_k is determined as $T_k \leftarrow \frac{r}{\log k+c}$ (where c is a constant). Typically, the cooling schemes which satisfies the logarithmic law will be too slow for the practical purposes. Therefore, a faster cooling scheme is adopted in practice. One of the most popular ones follows a geometric law: $T_k \leftarrow \alpha \times T_{k-1}$, where $\alpha \in (0, 1)$, which corresponds to an exponentially decay of the temperature.

The cooling scheme can be used for balancing between diversification and intensification. For example, at the beginning of the search, T_k might be constant or linearly decreasing in order to sample the search space; then, T_k might follow a rule such as the geometric one in order to make the algorithm converge to a local minimum at the end of the search. The cooling scheme and the initial temperature should be adapted to the particular problem instance considered, since the cost of escaping form local minima depends on the structure of the search landscape. A simple way of empirically determining the starting temperature T_0 is to initially sample the search space with a random walk to roughly evaluate the average and the variance of objective function values. Based on the samples, the starting temperature can be fixed to favor the uphill-moves. Reference of successful applications of SA can be found in Fleischer [39], Ingber [59], Aarts et al. [1].

Genetic Algorithms for Noisy Function

Genetic algorithms was first introduced by Holland [57]. They falls under the computational algorithms called evolutionary computation (EC). EC algorithms can be characterized as computational models of evolutionary process that take inspiration from the natural selection. At each iteration, a number of operators is applied to the individuals of the current population to generate the individuals of the population of the next generation (iteration). EC algorithms use operators called *recombination* or *crossover* to recombine two or more individuals to produce new ones. They also use *mutation* or *modification* operators which cause a self-adaptation of individuals. The driving force in evolutionary algorithms is the *selection* of individuals based on their *fitness* (which can be based on the objective function or some other kind of quality measure). Individuals with a higher fitness have a higher probability to be chosen as members of the population of the next iteration (or as parents for the generation of new individuals). This corresponds to the principle of *survival of the fittest* in natural evolution. It is the capability of the nature to adapt to a changing environment, which gave the inspiration for EC algorithms. Other classes of EC includes **evolutionary programming** (EP) as introduced by Fogel [43], Fogel et al. [42], **evolutionary strategies** (ES) proposed by Rechenberg [83]. Hertz and Kobler [56] give a good overview of the different components of EC algorithms and of the possibilities to define them. In the context of stochastic CO problems, Fitzpatrick and Grefenstette [38] proposed the genetic algorithms based on sampling techniques. Without loss of generality, we can use this approach to PTSP. Algorithm 4 shows the basic structure of stochastic genetic algorithms.

In this algorithm, P denotes the population of individuals. A population of offsprings is generated by the application of *recombination* and *mutation* operators and the individuals for the next population are *selected* from the union of the old population and the offspring population. The main features of the Genetic Algorithm are:

Description of the individuals: GA works with populations of individuals. These individuals are not necessarily solutions to the considered instance. They may be partial solutions, or sets of solutions. Most commonly used individuals in TSP is the representation of solutions as bit-strings or as permutations of number of cities n. Furthermore, individuals are called *genotypes*, whereas the solutions that are encoded by individuals are called *phenotypes*. Radcliffe [82] proposed various representation schemes for different types of problems.

Evolution process: The algorithm has to choose which individuals will enter the population of the next iteration. This is done by a selection scheme. In PTSP, the performance of each candidate solution is computed by estimating the performance via sampling techniques.

Algorithm 4 Genetic Algorithms for Noisy Environments

input: an instance C of a PTSP problem
$s \leftarrow$ `GenerateInitialPopulation`()
`Evaluate`(s)
while termination conditions not met **do**
  $s' \leftarrow$ `Recombine`(s)
  $s'' \leftarrow$ `Mutate`(s')
  Based on N_m random scenarios ω_v, compute a sample estimate

$$\varepsilon(F(s') - F(s'')) = \frac{1}{N_m} \sum_{v=1}^{N_m} f(s', \omega_v) - f(s'', \omega_v)$$

  if $\varepsilon(F(s') - F(s'')) < 0$ **then**
    `Evaluate`(s'')
    $P \leftarrow$ `Select`(s'', s')
  end if
  $s_{best} \leftarrow$ best solution in s
end while
output: s_{best}, "candidate" to optimal solution for C

Neighborhood function: This function defines the rule for recombination. A neighborhood function $\mathcal{N}_{EC} : I \rightarrow 2^I$ assigns to each individual $i \in I$ a set of individuals $\mathcal{N}_{EC}(i) \subseteq I$ whose members are permitted to act as recombination partners for i to create offspring.

Information sources: The most common form of information sources to create offsprings (i.e., new individuals) is a two-parent crossover. Mühlenbein and Voigt [76] and Syswerda [93] proposed various methodologies to generate the next population.

Intensification and Diversification: EC algorithms that apply a local search to each individual of a population to intensify the search process are often called as Memetic Algorithms [75]. On the other hand, the search process is diversified by using a mutation operator. DeJong [29], Cavicchio [22], Goldberg and Richardson [48] and Mahfoud [69] present a substantial explanations for intensification and diversification nature of genetic algorithms.

2.2.2 Mathematical Approaches

In this section we present a brief description of the most widely used mathematical solution techniques for the PTSP. In the following section, we limit ourselves to the very basic concepts characterizing each technique without entering into the explanations. Therefore, the goal of this sub-section is to give a overview of the state-of-the-art *exact* solution techniques.

Sample Average Approximation

The sample average approximation (SAA) [27] is an approach for solving stochastic optimization problems by using *monte carlo* simulation. In this technique the expected objective function of the stochastic problem is approximated by a sample average estimate derived from a random sample. The resulting sample average approximating problem is then solved by deterministic optimization techniques such as *branch and cut* [94]. The process is repeated with different samples to obtain candidate solutions along with statistical estimates of the variations in the optimal solutions.

Variable Sample Random Search Methods

Verweij et al. [94] proposed the application of a certain class of Monte Carlo methods to solve stochastic CO problems. In particular, they studied variable-sample techniques, in which the

objective function is replaced, at each iteration, by a sample average approximation. They also provided general results on the schedule of sample sizes, under which variable-sample methods yield consistent estimators as well as bounds on the estimation error. The work illustrate these ideas by studying a modification of the well-known pure random search method and adapting the variable-sample scheme to show the convergence of the algorithm. In pure random search algorithms, the randomness is not used blindly but in an intelligent, biased form.

Stochastic Branch and Bound Method

The main idea of the stochastic branch and bound method is to iteratively execute three operations:

- partitioning a large solution S space into smaller subsets S_1, S_2, S_3,..., S_n
- stochastic estimation of the objective function within the subsets
- removal of some subsets

If the branch-and-bound works well, most of the S_i will be eliminated. At the next step, partition is done to the remaining S_i and the algorithm is repeated. A stochastic version of the branch and bound method is proposed for solving stochastic global optimization problems by Norkin et al. [78]. The method, instead of deterministic bounds, uses stochastic upper and lower estimates of the optimal value of subproblems, to guide the partitioning process. The authors also prove the convergence of this method. Methods for constructing random bounds for stochastic global optimization problems are also discussed in this work.

Stochastic Ruler Method

Yan and Mukai [97] described the stochastic ruler (SR) method that is related to, but different from the SA method. While the objective value at a new solution candidate is compared with that of the current solution candidate in SA, the objective value at a new solution candidate is compared against a probabilistic ruler in the SR method, where the ruler range covers the range of the observed objective function values. The convergence is shown to be global. Alrefaei and Andradottir [2] proposed another method based on a modification of the SR method. The new algorithm uses a finite number of random scenarios for each iteration whereas the SR method uses an increasing sequence of scenarios per iteration.

Nested Partition Method

The nested partition (NP) algorithm is a new optimization algorithm that is based on the concept of adaptive sampling. The NP algorithm steers the sampling effort toward a decreasing search space until the search space is indivisible. The algorithm partitions the solution feasible region into subregions where it samples each subregion and chooses the best promising subregion to partition further. Regions not belonging to the most promising subregion are aggregated into a surrounding subregion which is also sampled. The algorithm moves back to a larger subregion if the surrounding subregion is found to be more promising. It is shown in [87] that this algorithm forms a *markov chain* and would recognize the global optimum solution that forms an absorbing state and never leaves this state. Making a correct move, between the different transient states until being absorbed in the global optimum solution state, depends on lots of factors such as the partitioning scheme and the quality of the samples [88].

Chapter 3

Proposed Approach

This chapter is exclusively dedicated to the description of the ACO/F-Race algorithm. Section 3.1 starts with a technical overview of the general racing approach. Subsection 3.1.1 describes the ACO/F-Race solution technique to solve the PTSP. The introductory part of this chapter and the description of ACO and its variant S-ACO from the previous chapter will provide sufficient background to understand the proposed approach. Therefore, this subsection doesn't reprise the ACO concepts but concentrates on the description of adopting F-Race in the ACO framework to tackle PTSP. Furthermore, for the sake of clarity and illustration, we focus on the homogeneous version of PTSP. Without loss of generality, all the algorithms proposed in this section can be adapted to heterogenous PTSP. Section 3.2 describes the computational procedures to improve the solution found by the ACO/F-Race. We propose two local search algorithms for PTSP called *empirical 2-opt* and *empirical 2.5-opt* inspired by *2-opt* and *2.5-opt* local search algorithms designed to solve the TSP.

3.1 ACO/F-Race Algorithm

The family of *racing approaches* is inspired by the algorithm *Hoeffding race* introduced by Maron and Moore [71] for solving the model selection problem in machine learning. The problem of model selection can be thought of as trying to find the best student in a classroom using a series of tests. This problem, in different flavors, occurs repeatedly in the field of machine learning [73]. There are several ways to determine the solution to this problem as proposed by Weiss and Kulikowski [96]. The key idea behind *Hoeffding race* can be explained in the following way: All the students in the class are evaluated with the first set of tests at once. After a small number of tests, we can distinguish the best students, that is, those with the higher grades, from those with the lower grades. Using statistical bounds, the students whose grades are *significantly* worse than the best ones are discarded and not considered anymore for the further tests. Depending on the feedback from several evaluations, more students can be differentiated and eliminated. Quite similarly, the racing algorithm concentrates the computational effort on the best models by discarding the inaccurate models by not testing them unnecessarily.

In the context of metaheuristics, racing algorithms are designed to select the best performing settings of the parameter values [18], [17]. The solution quality of ACO and other metaheuristics is susceptible to the stochastic nature of the algorithms. Moreover, it is also depends on the values assigned to the parameters in the algorithm. Given a set of parameters and some instances of a CO problem, the racing procedure computes the best parameter configuration for the given CO problem in a limited amount of time. For example, let us consider the S-ACO [52] described in Chapter 2. Let us also consider a scenario in which we need to find the best parameters setting for this algorithm where each parameter has several values such as $\alpha \in \{1,1.25,1.5,2\}$, $\beta \in \{0,1,3,5\}$, $\rho \in \{0.6,0.7,0.8,0.9\}$, $c_1 \in \{0,0.005,0.002\ \}$ and $c_2 \in \{0,0.1,0.2,0.3\}$. Clearly, each

possible combination of the parameters setting represents one particular configuration, expressing a total number of 4x4x4x3x4=768 parameters configurations. Furthermore, the stochastic nature of the algorithm demands several runs for each configuration to hypothesize the best setting. In other words, we need to run the algorithm with each parameters configuration several times by changing the random seed used by the metaheuristic algorithm. A brute force approach to the problem consists of evaluating all the possible configurations on a *sufficiently large* number of samples [49] and selecting the one with the best average solution quality in all or most of samples. On the other hand, taking the inspiration from the *Hoeffding race*, the racing procedure computes the best configuration for the given instance of the CO problem by concentrating on the better configurations by leaving out the inferior parameters configurations. Intuitively, for a fixed computational time, the racing algorithm evaluates more samples only on the algorithms with superior parameters settings whereas brute force approach evaluates relatively less samples on all the possible settings.

The racing algorithm proposed for tuning metaheuristics is known as F-Race. This is based on Friedman two-way analysis of variance by ranks [67], a *statistical method*[1] for *hypothesis*[2] testing. F-Race procedure can be demonstrated in the following way: Let us assume that F-Race has evaluated k samples and $n = \{\theta_1, ..., \theta_n\}$ configurations are still available in the race. The assumption taken by the Friedman test is that all the observed solutions are from the same population, that is, there is no statistically significant difference [16] in the observed solutions. Mathematically, the observed solutions are mutually independent n-variate random variables $(C^k(\theta_1, i_l), C^k(\theta_2, i_l), ..., C^k(\theta_n, i_l))$ called *blocks* [28]. More precisely, the value $C^k(\theta_1, i_l)$ corresponds to the computed solution on the sample i_l for the configuration θ_1 in the race at step k.

Step	Observed Solution			
$k-3$	$C^{k-3}(\theta_1, i_{l-3})$	$C^{k-3}(\theta_2, i_{l-3})$	$\cdots$	$C^{k-3}(\theta_n, i_{l-3})$
$k-2$	$C^{k-2}(\theta_1, i_{l-2})$	$C^{k-2}(\theta_2, i_{l-2})$	$\cdots$	$C^{k-2}(\theta_n, i_{l-2})$
$k-1$	$C^{k-1}(\theta_1, i_{l-1})$	$C^{k-1}(\theta_2, i_{l-1})$	$\cdots$	$C^{k-1}(\theta_n, i_{l-1})$
k	$C^{k}(\theta_1, i_{l})$	$C^{k}(\theta_2, i_{l})$	$\cdots$	$C^{k}(\theta_n, i_{l})$

In each block, $C(\theta, i)$ are ranked from the smallest to the largest. Average ranks are used in case of ties. For each configuration $\theta_j \in \{\theta_1, ..., \theta_n\}$, R_{lj} denotes the rank of θ_j within block l, and $R_j = \sum_{l=1}^{k} R_{lj}$ represents the sum of the ranks over all instances i_l, with $1 \leq l \leq k$.

Step	Corresponding Rank			
$k-3$	$R_{1,k-3}$	$R_{2,k-3}$	$\cdots$	$R_{n,(k-3)}$
$k-2$	$R_{1,k-2}$	$R_{2,k-2}$	$\cdots$	$R_{n,(k-2)}$
$k-1$	$R_{1,k-1}$	$R_{2,k-1}$	$\cdots$	$R_{n,(k-1)}$
k	$R_{1,k}$	$R_{2,k}$	$\cdots$	$R_{n,(k)}$

The Friedman test considers the following statistic [26].

$$\mathbf{T} = \frac{(n-1)\sum_{j=1}^{n}\left(R_j - \frac{k(n+1)}{2}\right)^2}{\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4}} \tag{3.1}$$

To accept the assumption of the hypothesis that all possible rankings of the solutions within each block are equally likely, $\mathbf{T}$ is approximately χ^2 distributed with $n-1$ degrees of freedom. In

[1] Statistical methods are used to determine if the difference in the results (solution) from a set of experiments (several runs) are great enough to conclude that the difference is statistically significant.

[2] A hypothesis is an assumption about the distribution of the results(solution).

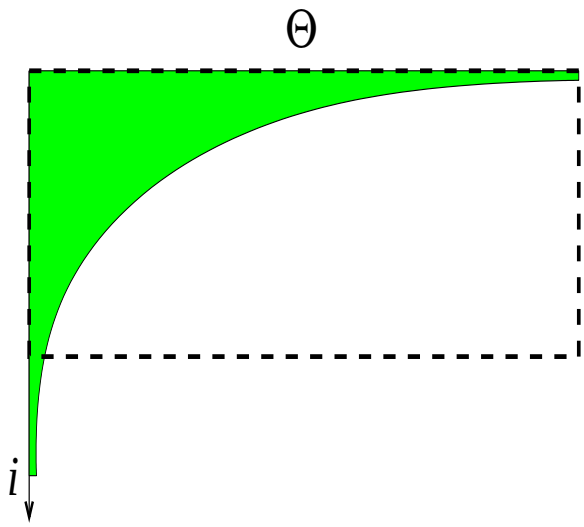

Figure 3.1: A visual representation of the amount of computation needed by brute force (rectangle comprised by dashed line) and racing approach(shadowed area).

F-Race, if the hypothesis is accepted at step k, then all the configurations from step k are allowed to race for the next step $k+1$. On the other hand, if the observed $\mathbf{T}$ exceeds the $1-\alpha$ quantile of such distribution, the hypothesis is rejected. As a consequence, at the approximate level α, there is a possibility of dropping at least a configuration θ_j from $\{\theta_1, ..., \theta_n\}$ which shows statistically significant difference from the other configurations.

Next step after rejecting the hypothesis is to identify which configuration has to be dropped from further racing. For this reason, pairwise comparison between individual candidates is performed. The candidates θ_j and θ_h are considered different if:

$$\frac{\mid R_j - R_h \mid}{\sqrt{\frac{2k(1-\frac{T}{k(n-1)})(\sum_{l=1}^{k}\sum_{j=1}^{n} R_{lj}^2 - \frac{kn(n+1)^2}{4})}{(k-1)(n-1)}}} > t_1 - \alpha/2 \tag{3.2}$$

where $t_1-\alpha/2$ is the $1-\alpha/2$ quantile of the student's t distribution. Therefore in F-Race, rejection of hypothesis is followed by pairwise comparisons which are executed between the best candidate and each other one. All the candidates, whose ranks are significantly worse than the best, at step k, are discarded and will not appear in the next step $k+1$.

The ranking of the quantiles $C(\theta, i)$ plays the vital role for the efficient computation of the best configuration by F-Race approach [18] [17] in the following way:

- *Nonparametric nature:* Nonparametric tests are powerful in detecting population differences when certain assumptions are not satisfied. On the other hand, if we consider the parametric tests, for example, t test assumes the values from each sample should come from a normal distribution. *Central limit theorem* states that when the samples are very large, then the sample means will follow the normal distribution even if the respective variable is not normally distributed in the observation. Hence, the parametric methods like t test are more powerful and appropriate for large samples. The smaller the sample size, the lesser will be the ability of t test to detect the differences. The striking feature of the nonparametric test is that it doesn't requires the *hypothesis* formulation and thus they are distribution-free. Further it is appropriate when the sample sizes are small. In F-Race, the number of candidates is reduced as soon as possible and the further evaluations take place only on a small

number of surviving candidates. Therefore, we cannot use parametric tests and we have to adopt nonparametric tests. More debates, discussions and facts about nonparametric and parametric methods were presented in Larson [67].

- *Blocking design:* Blocking design reduces the variability due to a known source of variation. Given a set of configuration for the stochastic algorithms such as metaheuristic algorithms and treating them with several different samples results in a large variation in the solution quality. Blocking is an efficient way of normalizing the solution quality observed on several different samples [28]. Therefore, the ranking of different configurations within each sample eliminates the risk of sample's influence in the accepting or rejecting a configuration.

Notwithstanding the inspiration from a number of machine learning algorithms such as Maron [70], Gratch et al. [50], Chien et al. [25], Moore and Lee [74], F-Race is the first racing algorithm to implement blocking through ranking and to adopt the Friedman test as the aggregate test [49] over all candidates, to be performed prior to any pairwise test. By means of F-Race, Birattari et al. [18] first proposed the formal definition for the problem of tuning metaheuristics from a machine learning perspective. A substantial explanation of the topic is given in Birattari [17].

3.1.1 Solution Methodology

ACO/F-Race algorithm, we propose in this thesis, is inspired by S-ACO [52] and F-Race [18] approaches. In this algorithm, F-Race is used in an original way as a component of the ACO algorithm. Similar to S-ACO, the proposed approach uses *empirical estimation* for tackling CO problems under uncertainty. Without loss of generality, ACO/F-Race can be adopted to solve the PTSP, in specific, to optimize the expected length of the *a posteriori tour*. The pseudo-code of the ACO/F-Race is presented in Algorithm 5.

Algorithm 5 ACO/F-Race Algorithm

input: an instance C of a PTSP problem
set $\tau_{ij} \leftarrow 1$ for all (i,j)
for round $r = 1, 2, 3.....$ **do**
  for ant $\sigma = 1, 2, 3.....m$ **do**
    place the ant in a random city i
    choose the next city by *random proportional rule* until all the cities were included in the tour
  end for
  if $(r = 1)$ **then**
    Based on N_m random scenarios ω_v, race m ant's tour s_m

$$s_{best} \leftarrow \textit{F-Race}(N_m, s_m)$$

  else
    Based on N_m random scenarios ω_v, race m ant's tour s_m with *best-so-far* ant's tour s_{best}

$$s_{best} \leftarrow \textit{F-Race}(N_m, s_m \cup s_{best})$$

  end if
  evaporation: set $\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \forall (i, j)$;
  best-so-far pheromone update: set $\tau_{ij} \leftarrow \tau_{ij} + c_1, \forall (i, j) \in s_{best}$;
  iteration-best pheromone update: set $\tau_{ij} \leftarrow \tau_{ij} + c_2, \forall (i, j) \in s_{best}$;
end for

Clearly, the algorithm reveals that most of the components are similar to S-ACO. Typically, the algorithm starts by initializing the pheromone on each arcs (i, j) of the PTSP to unity. Subsequently, the iterative *ContructSolution* phase of the ACO/F-Race starts. The iteration begins

by placing m ants randomly on the cities of the PTSP. Each ant construct the *a priori tour* according to the *random proportional rule* given by Equation 3.3 (which is same as the Equation 2.2). Therefore, an ant k, currently at city i moves to the city j probabilistically using Equation 3.3, where N_i^k is the set of all cities yet to be visited by the ant k.

$$\mathbf{p_{ij}^k} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}}, \text{if } j \in N_i^k \tag{3.3}$$

Once m ants finished constructing their respective *a priori tour*, the original element of the proposed approach, F-Race algorithm starts. During the first iteration, the function *F-Race* evaluates and compares m ants *a priori tour* to select the *iteration-best* solution. Since there is no *best-so-far* tour at the first iteration, the selected *iteration-best* solution is also considered as the *best-so-far* solution. From the subsequent iterations, the function *F-Race* evaluates m ant's solution together with the *best-so-far* solution.

Now we describe the way in which the *F-Race* approach is adopted as a component for selecting the *best-so-far* solution. As described earlier, once the ants constructed the tour, it's *a priori* tours are then passed to the function F-Race as described in Algorithm 6. In S-ACO, the value

Algorithm 6 Racing Function

```
Function F-Race(N_m,Θ)
M ← 2 * N_m{/*stopping criteria for each iteration in ACO/F-RACE */}
evaluated-candidates ← 0 {/* number of evaluated candidates */}
steps ← 0{/* number of steps performed */}
while (evaluated-candidates+surviving-candidates) ≤ M do
  steps ← steps + 1
  surviving-candidates ← | Θ |{/* number of surviving candidates */}
  generate a random scenario ω
  for each a priori tour θ_i ∈ Θ do
    compute the a posteriori tour θ_i by skipping strategy
  end for
  evaluated-candidates ← evaluated-candidates+surviving-candidates
  if (Failed_Significance_Friedman(steps, θ_1 ··· θ_|Θ|) then
    Θ ← Θ \ pairwise(θ_i ∈ Θ)
  end if
end while
return θ_best from surviving candidates of the set Θ
```

of N_m determines the number of random scenarios to be evaluated for each iteration to select the *best-so-far* solution. At each iteration (except first iteration), the *iteration-best* and *best-so-far* solution is evaluated on N_m random scenarios. As a consequence, the number of evaluations in S-ACO is twice that of N_m. In order to make a fair experimental evaluation and comparison of ACO/F-Race with S-ACO, we have adopted the same number of evaluations in the later as that of the former. Therefore, the termination condition of the *F-Race function* is set such that the number of evaluations should reach twice the value of N_m. Until this termination condition is encountered, at each step, a new step specific random scenario ω is generated according to the probability p_i associated with each city i in the PTSP. This ω is used for evaluating the solution which are still in the race. Intuitively, at each step, *a posteriori* tour length θ_i is computed for each *a priori* tour $\theta_i \in \Theta$ (the set Θ contains the surviving ant's solution) with ω (cities to be visited in the *a posteriori* tour) using skipping strategy.

Without loss of generality, the racing approach designed for tuning metaheuristic algorithms can be employed to discard the ants with inferior solutions. The function ($Failed\_Significance\_Friedman(steps, \theta_1 \cdots \theta_{|\Theta}$

test for the failure of the hypothesis using *Friedman-two-way analysis of variance by ranks*. More precisely, at each step, the *a posteriori* tour length of the surviving ants Θ referred as $\theta_1 \cdots \theta_{|\Theta|}$ is tested for the *statistically significant* difference by computing $\mathbf{T}$ using the Equation 3.1. As a consequence of such difference, the next step is to discard some ants from the race with the function *pairwise*($\theta_i \in \Theta$). Using the Equation 3.2, this function removes the ants whose solution's rank are significantly worse than the rank of best ant's solution. Racing of solutions stops when the termination condition becomes true. The *a priori* tour that wins the race is returned as the θ_{best} to the main algorithm.

The solution that wins the *race* is employed for updating the pheromone and stored as the *best-so-far* solution for the next iteration of the algorithm. Since the *best-so-far* solution from the previous iteration has to compete with the solutions generated at the current iteration, the concept of *iteration-best* solution doesn't occur in ACO/F-Race. Although the concept of *iteration-best* pheromone update doesn't have any sense in ACO/F-Race, a fair experimental comparison with S-ACO requires it. Therefore, the *best-so-far* solution is used as the candidate for updating pheromone both in the *best-so-far* and the *iteration-best* pheromone updates.

The significant difference between S-ACO and ACO/F-Race lies in the technique used to select the *iteration-best* tour for each iteration. In S-ACO, the solutions produced at a given iteration are compared on the basis of a single scenario ω to select the *iteration-best* tour. On the basis of larger sample size N_m, whose size is computed dynamically, the *iteration-best* tour is compared with *best-so-far* tour in the algorithm. The tour with least expected *a posteriori* tour length among two solutions is selected and stored as the new *best-so-far* tour for future comparisons and to update the pheromone matrix for the global pheromone update phase. More precisely, S-ACO exploits sampling techniques and parametric test whereas ACO/F-Race uses F-Race, an algorithm originally developed for tuning metaheuristics based on nonparametric tests. Although it is not necessary to have *iteration-best* pheromone update phase and large sample size, ACO/F-Race solution construction and pheromone update were implemented as described in [52] for the experimental comparison.

3.2 Local Search Methodology

Metaheuristic algorithms frequently encounter a sequence of states in which it is impossible to improve the solution quality by itself. Notwithstanding this inadequacy, various research works on metaheuristics tell us that a promising approach to extract high-quality solutions is to couple a local search mechanism within the metaheuristics framework. Local search algorithms operate on the solutions found by metaheuristics by introducing some modification to obtain locally optimal solutions [92]. Furthermore, the consideration of local search algorithm as a stand alone solution technique for CO problems suffers from the problem of finding good starting solutions. Therefore, the key idea of coupling metaheuristic algorithms with local search algorithms can be explained in the following way: On the one hand, the metaheuristic algorithms provide a high quality starting solutions. On the other hand, local search algorithms operate on these quality solutions to provide a higher quality solutions.

ACO framework has the flexibility of coupling local search in the definition. Once ants complete their solution construction phase, the local search algorithms are allowed to refine their solutions to yield a higher quality solutions. Afterwards, the pheromones are updated on the arcs with respect to the improved solutions found by local search procedures. With various experimental evidence, Dorigo and Stützle [34] proposed that combining the ant's solution construction phase with the local search procedures is a promising approach and there is a very high possibility and probability that the local search procedures can improve the solution constructed by the ants.

Fortunately enough, the number of possible choices when combining the local search with ACO algorithms is quite large [34]. Notwithstanding these possibilities, it is very important to understand the fundamental nature of local search procedure concerning effectiveness and efficiency. Most of the local search procedures demand high computational time to refine the solutions. Therefore, the decision is up to the algorithm designer to choose, for a given computation time, a frequent refining local search algorithm that slightly improves the solution quality of the initial solutions, or a less frequent, slow and effective local search algorithm that significantly improves the solution quality.

The number of ants, the necessity to use heuristic information, parameter settings and which ant(s) should be allowed to improve their solutions by a local search are some of the major elements to be set before adopting a particular local search algorithm in the ACO framework. Most effective and recommended local search in ACO algorithms is *best-so-far* solution refinement [34]. It is also interesting to know whether ACO algorithm with a best parameter setting will remain best after adding the local search.

Local search algorithms for the TSP are based on the k-exchange neighborhood relation, in which candidate solutions s and s' are direct neighbors if and only if, s' can be obtained from s by deleting a set of k edges and rewiring the resulting fragments into a complete tour by inserting a different set of k edges [92]. Within the ACO framework, the local search techniques such as *2-opt*, *2.5-opt* and *3-opt* [92] are used quite often in TSP to improve the solution found by ants. All three implementations exploit three standard speedup techniques: use of nearest neighbor list of limited length, the use of a fixed radius nearest-neighbor search, and the use of don't look bits. These techniques increases the computation time sub-quadratically, that is, $O(n^2)$ where n is the size of the instance. Large number of speed up techniques and its substantial explanation can be found in Bentley [5], Johnson and McGeoch [63] and Reinelt [84].

A quick and straightforward implementation [92] of a k-exchange iterative improvement algorithm considers, in each step, all possible combinations for the k edges to be deleted and replaced. After deleting k edges from a given candidate solution s, the number of ways in which the resulting fragments can be reconnected into a candidate solution different from s depends on k; For $k = 2$, after deleting two edges (u_i, u_j) and (u_k, u_l), the only way to reconnect the two partial tours into a different complete tour is by introducing the edges (u_i, u_k) and (u_l, u_j). It is interesting to note that after a 2-exchange move, one of the two partial tours is reversed.

Candidates for k-exchange moves can be identified quickly using the speed up techniques. For example, to enable an efficient access to the cities in the given tour that are connected to a given city u_i by edges with shorter distance can be achieved in the form of a list of neighboring cities u_k that is sorted according to the distance in ascending order. By using such candidate list for all the cities in the TSP, nearest neighbor search can be performed very efficiently. The use of lists for *2-opt* often leads to improvement in the quality of the local optima found by these algorithms. Recommended value for the number of neighbors ranges from 10 to 40 [92]. Full candidate list for each city incorporating all the $n - 1$ other cities in the TSP requires $O(n^2 log\ n)$ computational time and $O(n^2)$ memory. Therefore, to reduce memory requirements, it is often desirable to use fixed length nearest neighbor list. As a consequence, the tours obtained from *2-opt* using nearest neighbor list are no longer guaranteed to be locally optimal, because some improving moves may be missed. Alternative approachs to construct nearest neighbor lists were proposed by Peckny and Miller [81], Helsgaum [55].

Empirical 2-opt local search for PTSP

Bertsimas [8] first introduced the local search algorithms based on mathematical approximations, for CO problems under uncertainty known as *2-p-opt* and *1-shift* algorithms. Bianchi et al. [15]

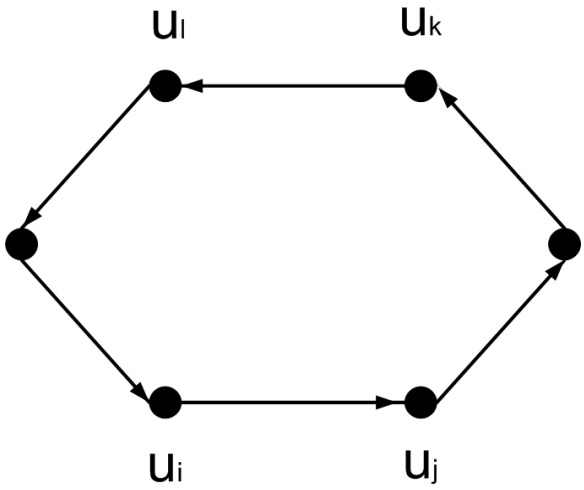

Figure 3.2: The *a priori* tour of a PTSP visiting all the cities once and only once. The edges (u_i, u_j) and (u_k, u_l) are selected for the *2-opt* move.

and Bianchi and Campbell [12] extended the *2-p-opt* and *1-shift* algorithms to the PTSP. The local search algorithm we propose in this thesis, *empirical 2-opt local search* for PTSP, is based on the *2-opt* local search in *best-so-far* solution using nearest neighbor list as the speed up technique. This technique, to the best of our knowledge, is the first local search for the PTSP which uses empirical estimation.

Before entering into the algorithmic details, it is important to understand the meaning of solution quality improvement in the context of TSP and PTSP. In the TSP, the improvement is straightforward, in the sense that the improvement can be calculated directly. Intuitively, if we consider the *2-opt* move between the edges (u_i, u_j) and (u_k, u_l) of the TSP, the only possible refinement is (u_i, u_k) and (u_l, u_j). Therefore, the improvement in the solution quality $\triangle$ is considered as the difference between the length of the solution components after and before *2-opt* move as shown in Equation 3.4:

$$\triangle = (d[u_i][u_k] + d[u_l][u_j]) - (d[u_i][u_j] + d[u_k][u_l]) \tag{3.4}$$

where, $d[u_i][u_k]$, $d[u_l][u_j]$, $d[u_i][u_j]$, $d[u_k][u_l]$ represents the distance between the respective cities in the TSP. The solution quality is said to be improved when $\triangle < 0$ and viceversa. On the other hand, for the PTSP, the *2-opt* move in the *best-so-far* solution is considered as an improved move only when it reduces the expected length of the *a posteriori* tour with respect to the set of all random scenarios ω_v. Therefore, in ACO/F-Race algorithm, to compute the improvement for a *2-opt* move, it is customary to remember the random scenarios ω_v which are evaluated to select the *best-so-far* solution for a given iteration.

In the context of PTSP, it should be noted that since the edges (u_i, u_j) and (u_k, u_l) for *2-opt* move are selected from the *a priori* solution, there are possibilities that the cities u_i, u_j, u_k, u_l may not occur in the random scenarios ω_v. A straight forward approach which re-computes the *a posteriori* tour length for each random scenario doesn't needs to consider this issue. But this re-computation increases the total computational time especially when the number of random scenarios is large. This case is trivial when the number of iteration increases. Therefore, we propose a simple way of computing the improvement similar to the Equation 3.4. For illustration, consider the *best-so-far* solution as shown in Figure 3.2. Assume that the edges (u_i, u_j) and

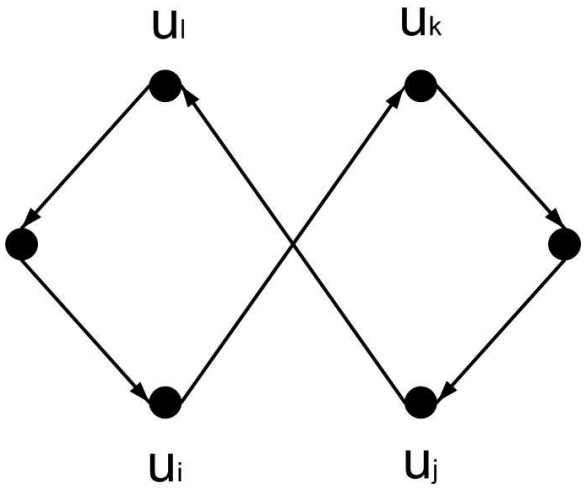

Figure 3.3: The modified *a priori* tour of a PTSP visiting all the cities once and only once. The edges (u_i, u_j) and (u_k, u_l) are modified as (u_i, u_k) and (u_j, u_l) with *2-opt* move

(u_k, u_l) in the *best-so-far* solution are selected for *2-opt* move. As a consequence, the new tour with interchanged edges (u_i, u_k) and (u_l, u_j) is shown in Figure 3.3. Now we have to compute the improvement measure $\triangle_{\omega_c}$, for each random scenario $\omega_c \in \omega_v$. We can use the same Equation 3.4 to compute $\triangle_{\omega_c}$, when all the cities in the selected edges for *2-opt* move are included in a random scenario ω_c. On the other hand, let us consider a case when the city u_i is not included in the random scenario ω_c. In such scenario, we have to find the previous city u_{prev} which has to be visited before visiting u_i in the modified *a priori* solution. Intuitively, in the modified *a posteriori* tour, the city u_{prev} is visited followed by visiting the city u_k by skipping the city u_i as shown in Figure 3.4. Therefore, the *2-opt* improvement measure $\triangle_{\omega_c}$ for the random scenario ω_c can be derived as:

$$\triangle_{\omega_c} = (d[u_{prev}][u_k] + d[u_l][u_j]) - (d[u_{prev}][u_j] + d[u_k][u_l]) \tag{3.5}$$

Similar argument can be made when the random scenario doesn't contain u_j as shown in Figure 3.5. Equation for this case is given as:

$$\triangle_{\omega_c} = (d[u_i][u_k] + d[u_l][u_{prev}]) - (d[u_i][u_{prev}] + d[u_k][u_l]) \tag{3.6}$$

For the random scenario that doesn't have cities u_k or u_l, the equation can be derived by finding the next city to be visited from u_k or u_l in the modified solution. The scenarios without u_k and u_l are shown in Figures 3.6 and 3.7 respectively. Equations for both the cases are established as:

$$\triangle_{\omega_c} = (d[u_i][u_{next}] + d[u_l][u_j]) - (d[u_i][u_j] + d[u_{next}][u_l]) \tag{3.7}$$

$$\triangle_{\omega_c} = (d[u_i][u_k] + d[u_{next}][u_j]) - (d[u_i][u_j] + d[u_k][u_{next}]) \tag{3.8}$$

The total improvement measure is the average of all improvement measure computed for each scenario $\omega_c \in \omega_v$. This is represented as:

$$\triangle_{\omega_v} = \frac{1}{v} \sum_{c=1}^{v} \triangle_{\omega_c} \tag{3.9}$$

Therefore, after the *2-opt* move, the solution quality is considered to be improved when $\triangle_{\omega_v} < 0$ and viceversa.

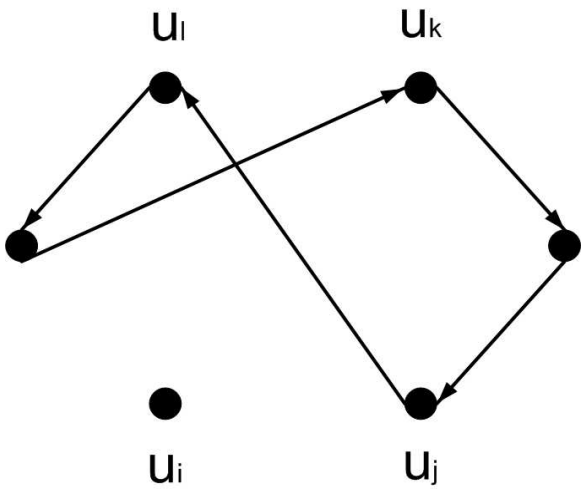

Figure 3.4: The *a posteriori* tour of a PTSP visiting the cities in the same order as modified *a priori* tour when the city u_i doesn't need to be visited.

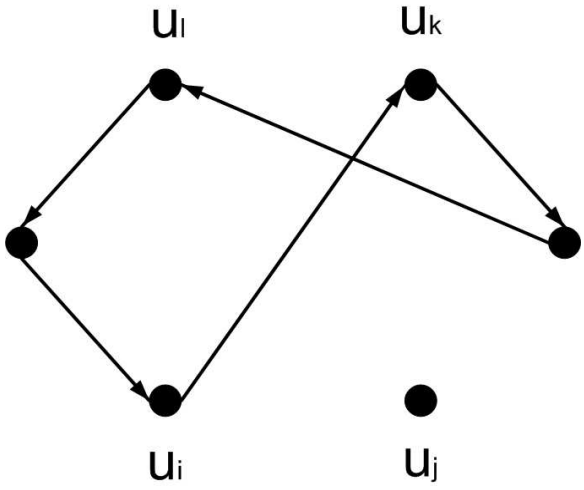

Figure 3.5: The *a posteriori* tour of a PTSP visiting the cities in the same order as modified *a priori* tour when the city u_j doesn't need to be visited.

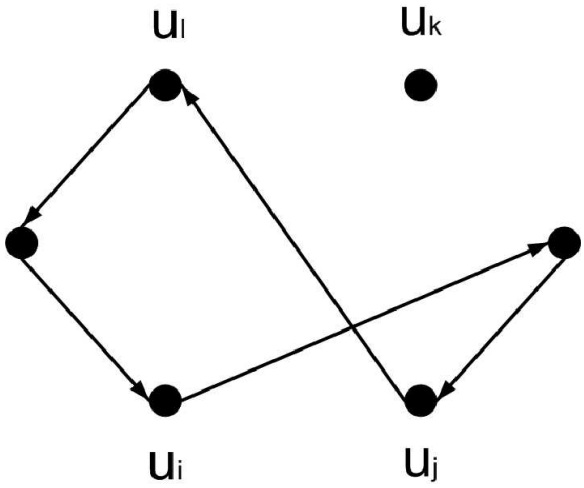

Figure 3.6: The *a posteriori* tour of a PTSP visiting the cities in the same order as modified *a priori* tour when the city u_k doesn't need to be visited.

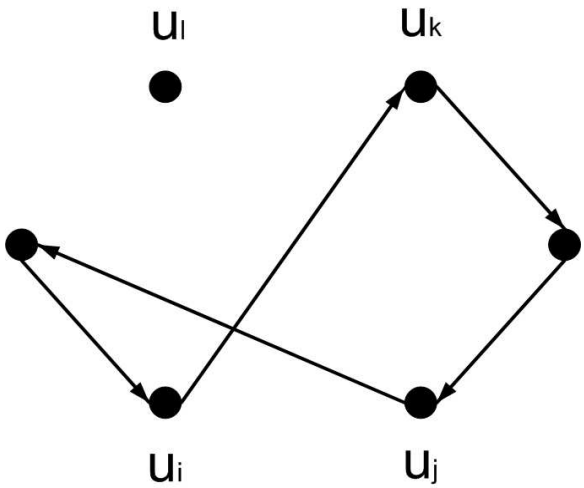

Figure 3.7: The *a posteriori* tour of a PTSP visiting the cities in the same order as modified *a priori* tour when the city u_l doesn't need to be visited.

Algorithm 7 Empirical 2-opt local search

Function **empirical 2-opt local search**(ω_v,s_{best})
{function *pick_edge_random()* randomly select an edge for the *2-opt* move}
$(u_i, u_j) \leftarrow$ *pick_edge_random()*
$(u_k, u_l) \leftarrow$ *pick_edge_random()* {function *excahnge_edge()* makes the *2-opt* move in s_{best}}
$s'_{best} \leftarrow$*exchange_edge*$((u_i, u_j), (u_k, u_l))$
for each random scenario $\omega_c \in \omega_v$ **do**
  if $u_i \in \omega_c$ **then**
    $first\_edge\_first \leftarrow u_i$
  else
    $first\_edge\_first \leftarrow previous\_city(u_i, s'_{best})$
  end if
  if $u_j \in \omega_c$ **then**
    $first\_edge\_second \leftarrow u_j$
  else
    $first\_edge\_second \leftarrow previous\_city(u_j, s'_{best})$
  end if
  if $u_k \in \omega_c$ **then**
    $second\_edge\_first \leftarrow u_k$
  else
    $second\_edge\_first \leftarrow next\_city(u_k, s'_{best})$
  end if
  if $u_l \in \omega_c$ **then**
    $second\_edge\_second \leftarrow u_l$
  else
    $second\_edge\_second \leftarrow next\_city(u_l, s'_{best})$
  end if
  $\triangle_{\omega_c} \leftarrow$ *(d[first_edge_first][second_edge_first]+d[first_edge_second][second_edge_second]) - (d[first_edge_first][first_edge_second]+d[second_edge_first][second_edge_second])*
end for
$\triangle_{\omega_v} = \frac{1}{v} \sum_{c=1}^{v} \triangle_{\omega_c}$
if $\triangle_{\omega_v} < 0$ **then**
  $s_{best} \leftarrow s'_{best}$
end if
return s_{best}

The pseudo code for the *empirical 2-opt* local search is presented in Algorithm 7. Once the *best-so-far* solution is computed by the *racing approach* in ACO/F-Race algorithm, the *2-opt* local search is employed to refine the solution.

The local search algorithm needs the *best-so-far* solution s_{best} and the set of all random scenarios ω_v used for selecting the *best-so-far* solution from the *F-Race algorithm*. The *2-opt* local search procedure starts by selecting two edges (u_i, u_j),(u_k, u_l) at random by using the function *pick_edge_random()* from the nearest neighbor list. It is customary to remember the selected list of edges to avoid selecting same edge more than once. The function *exchange_edge* applies the *2-opt* move between the edges to yield a modified solution s'_{best}. It is followed by evaluating the improvement measure for each random scenario $\omega_c \in \omega_v$. According to the cities that need to be visited in the random scenario, the improvement measure $\triangle_{\omega_c}$ is computed as illustrated earlier with the Figures 3.4, 3.5, 3.6, 3.7. Given a random scenario ω_c, the functions *previous_ city()* and *next_city()* are used to find the previous city and the next city in the modified *a priori* tour s'_{best} when the given city doesn't requires a visit in ω_c. The improvement measure $\triangle_{\omega_c}$ is computed for each random scenario. Followed by these evaluations, the average improvement $\triangle_{\omega_v}$ is calculated. Inexpensive computational time for *2-opt* local search gives the flexibility to have several refinements of *best-so-far* solution before updating the pheromone. In our case, *we employed the 2-opt local search repeatedly as long as the quality of the modified solution improves.* Finally, the *empirical 2-opt* local search algorithm returns the modified solution as s_{best} to the ACO/F-Race algorithm which is then used as the candidate for updating the pheromone.

Empirical 2.5-opt local search for PTSP

In this section, we present *empirical 2.5-opt* local search procedure for the PTSP based on empirical estimation and inspired by the *2.5-opt* local search procedure of TSP [92]. Let us consider that the algorithm wants to make a *2.5-opt* move between the edges (u_i, u_j) and (u_k, u_l). It first checks *2-opt* move between edges for the improvement. If it is not fruitful then it will explore the next improvement by inserting the city u_l between u_i and u_j. It is interesting to note that, in *2.5-opt* move *none* of the partial tour will be reversed. Obviously, the additional refinement phase in *2.5-opt* move requires extra computing time when compared to *2-opt* move. Notwithstanding the computational overhead, the experimental evidence of Bentley [5] shows that the amount of extra computational time is small and it leads to significantly better solution quality with respect to *2-opt* local search in the TSP. A substantial explanations for *2.5-opt* can be found in Stützle and Hoos [92], Dorigo and Stützle [34].

In the context of *2.5-opt* move, the only possible refinement is inserting the city u_l between the cities u_i and u_j. As a consequence, the edge starting from the city u_k is rewired to the city $u_{l_{next}}$ which has to be visited after visiting the city u_l. Figure 3.8 shows the re-fragmentation procedure of the *2.5-opt* move for the same *a priori* solution sketched in the Figure 3.2. The improvement in the solution quality is estimated by Equation 3.10 similar to the *2-opt* move.

$$\triangle = (d[u_i][u_l] + d[u_l][u_j] + d[u_k][u_{l_{next}}]) - (d[u_i][u_j] + d[u_k][u_l] + d[u_l][u_{l_{next}}]) \tag{3.10}$$

Considering different random scenarios and the cities to be visited in each of those random scenarios, the improvement measure $\triangle_{\omega_c}$ is computed. The pseudo code for *empirical 2.5-opt* is shown in Algorithm 8. The algorithm first checks for *2-opt* improvement, failure of which leads to *2.5-opt move*. The function *excahnge_edge2.5()* makes the *2.5-opt* re-fragmentation in the solution as illustrated in the Figure 3.8. The functions which are used in *empirical 2-opt* local search such as *previous_city()* and *next_city()* can be used for *2.5-opt* move without any modifications. The procedures for selecting the city and computing the improvement measure were same as that of the *empirical 2-opt* algorithm.

Algorithm 8 Emprical 2.5-opt local search

Function **empirical 2.5-opt local search**(ω_v,s_{best})
Perform the same operations as explained in the empirical 2-opt algorithm
If the empirical 2-opt move doesn't show any improvement then start the 2.5-opt *move in* s_{best}
{function *excahnge_edge2.5()* makes the *2.5-opt* move in s_{best}}
$s'_{best} \leftarrow$*exchange_edge2.5*$((u_i, u_j), (u_k, u_l))$
for each random scenario $\omega_c \in \omega_v$ **do**
  if $u_i \in \omega_c$ **then**
    $first\_edge\_first \leftarrow u_i$
  else
    $first\_edge\_first \leftarrow previous\_city(u_i, s'_{best})$
  end if
  if $u_j \in \omega_c$ **then**
    $first\_edge\_second \leftarrow u_j$
  else
    $first\_edge\_second \leftarrow previous\_city(u_j, s'_{best})$
  end if
  if $u_k \in \omega_c$ **then**
    $second\_edge\_first \leftarrow u_k$
  else
    $second\_edge\_first \leftarrow next\_city(u_k, s'_{best})$
  end if
  if $u_l \in \omega_c$ **then**
    $second\_edge\_second \leftarrow u_l$
  else
    $second\_edge\_second \leftarrow next\_city(u_l, s'_{best})$
  end if
  if $u_{l_{next}} \in \omega_c$ **then**
    $second\_edge\_second\_next \leftarrow u_{l_{next}}$
  else
    $second\_edge\_second\_next \leftarrow next\_city(u_{l_{next}}, s'_{best})$
  end if
  $\triangle_{\omega_c} \leftarrow$ *(d[first_edge_first][second_edge_second]+d[second_edge_second][first_edge_second]+ d[second_edge_first][second_edge_second_next]) -(d[first_edge_first][first_edge_second]+d[second_edge_first][second_ed d[second_edge_second][second_edge_second_next])*
end for
$\triangle_{\omega_v} = \frac{1}{v} \sum_{c=1}^{v} \triangle_{\omega_c}$
if $\triangle_{\omega_v} < 0$ **then**
  $s_{best} \leftarrow s'_{best}$
end if
return s_{best}

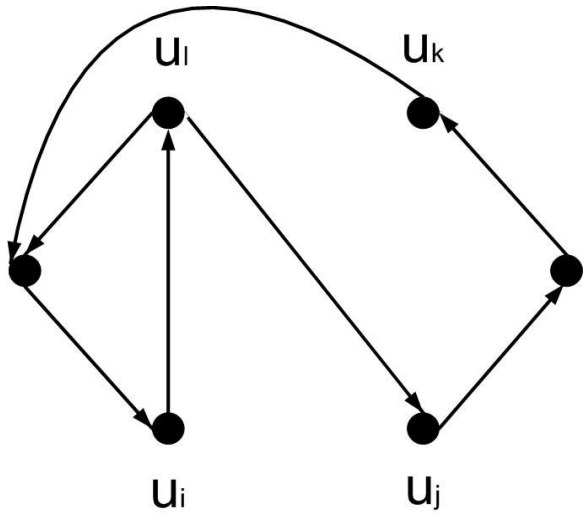

Figure 3.8: The modified *a priori* tour of a PTSP visiting all the cities once and only once. The edges (u_i, u_j), (u_k, u_l) and $(u_l, u_{l_{next}})$ are modified as (u_i, u_l), (u_l, u_j) and $(u_k, u_{l_{next}})$ by *2.5-opt* move which inserts the city u_l between u_i and u_j

Chapter 4

Experiments and Results

In this chapter we discuss the results of the experiments we conducted using the ACO/F-Race algorithm and *empirical local search* algorithms. Section 4.1 describes the experimental set-up, the problem sets, the hardware and software specifications and the output metrics for the experimental evaluations. Subsection 4.1.1 presents the experimental results concerning the behavior of ACO under uncertain conditions. More precisely, the computational experiments include the comparative study of the state-of-the-art ACO algorithms for solving PTSP such as explicit objective function based ACS (pACS) [13], simulation based ACO (S-ACO) [52] against the proposed ACO/F-Race approach. The next subsection 4.1.2 describes the results for empirical *2-opt and 2.5-opt* local search algorithms. We show how the empirical *2-opt and 2.5-opt* improves the solution quality of ACO/F-Race. We also present a set of preliminary results in which we compare the performance of empirical local search approaches against *1-shift local search* [15, 8], the state-of-the-art local search methodology for solving PTSP.

4.1 Experimental Framework

The primary goal of this thesis is to study the behavior of the ACO algorithms under uncertain conditions. More precisely, we are interested in evaluating the solution quality of the ACO algorithm which solves PTSP as TSP. In this context, we fixed Ant System [31] as the standard ACO algorithm with a slight modification, that is, the objective function is estimated on the basis of a single random scenario which is freshly sampled at each iteration of the algorithm. It is a standard practice to justify the proposal of a new algorithm by comparison of the new algorithm with the current state-of-the-art algorithm. The proposed algorithm, ACO/F-Race, based on the sampling and racing approach is inspired by S-ACO [52] which is based on the sampling technique. Quite naturally, it is more meaningful to compare the solution quality of the two algorithms to justify the significance of the proposed approach. The proposed empirical estimation approach is still relatively unstudied. We would, nevertheless like to make the first step towards the comparative study of ACO/F-Race with pACS [13] which is based on *mathematical approximation* and the state-of-the-art ACO algorithm designed to solve PTSP. In this thesis, we restrict our experiments to ACO. Henceforth we use the following notation.

Algorithm	Notation
Ant System	ACO-1
Simulation based ACO algorithm	S-ACO
Racing and Sampling ACO algorithm	ACO/F-Race
Explicit objective function based ACS	pACS

We use a standard benchmark to evaluate the quality of the algorithms. TSPLIB is one of the standard benchmarks with a library of sample instances for the TSP from various sources such

as cities in a country, drilling holes in a printed circuit board etc. These instances can be further subdivided into cases where the nodes are uniformly distributed and the cases where they are clustered. There are no standard benchmark for the PTSP. To gain some standardization, we carried out the experiments with the TSP instances. We used the TSP instances generated by the DIMACS [64][1] TSP generator. We are also interested in analyzing how the performance of the algorithm varies in terms of the organization of the nodes in the instances. In this context, we followed the experimental setup of Bertsimas and Howell [9]: case in which nodes are uniformly distributed and case in which nodes are clustered. For each case, we consider 100 TSP instances of 300 cities. From each TSP instance, we obtain 21 *homogeneous* PTSP instances by letting the probability associated with each city be $\{0,0.05,0.10,\cdots,0.95,1.0\}$. This resulted in 4200 PTSP instances for the experimental evaluation.

After the selection of the problem, we fixed the computational time for the algorithms as proposed by Dorigo and Stützle [34]. They compared various ACO algorithms to solve TSP instances from TSPLIB involving 198 and 783 cities using 100 seconds and 10000 seconds as the computational time respectively. Therefore, we allowed 120 seconds for each algorithm as the amount of computational time. It is interesting to see the quality of the solutions at the half way of this running time. For this purpose, we have also recorded the solution quality of the algorithms at 60 seconds.

The primary motivation behind this experimental study was to compare the evaluation procedure based on F-Race with the one proposed in S-ACO. Therefore, the algorithms were not fine-tuned, and the parameters adopted are those suggested by Gutjahr [52] for S-ACO. This might possibly introduce a bias in favor of S-ACO. In future, we will adopt F-Race approach [18] for tuning the algorithms.

Parameter	Notation	Value
Number of ants	σ	25
Pheromone exponent	α	1.0
Heuristic exponent	β	2.0
Pheromone evaporation factor	ρ	0.01
best-so-far update constant	c_1	0.04
iteration-best update constant	c_2	0.00
Number of nearest neighbors in tour construction	nn	25

Empirical Analysis of the Solution Quality

The goal of the optimal solution of the PTSP is to choose an *a priori tour* which minimizes the expected value of the *a posteriori* tour length. The *a priori* tour computed by each algorithm on each instance were then evaluated on 300 freshly generated random scenarios to compute the expected value of the *a posteriori* tour length. The random scenarios are sampled according to the probability associated with the instance.

The Computing Environment

The experiments presented in this chapter were performed on the cluster of personal computers called *Polyphemus* which is available at IRIDIA, Université Libre de Bruxells. More precisely, we used a group of clients composed of 16 nodes and each node features 2 AMD Opteron^{TM}244. All the nodes run the GNU/Linux operating system as distributed by Debian. All the algorithms mentioned in this chapter were implemented in C language based on the source code of Stützle

[1]DIMACS is a academic competition that aims create a reproducible picture of the state of the art in the area of TSP heuristics (their effectiveness, their robustness, their scalability, etc.)

[89][2] except pACS algorithm [13] which is implemented in C++. The source code was compiled for execution on *Polyphemus* using gcc, the GNU Compiler Collection of the Free Software Foundation, versions 3.2 and 3.3. The graphs and plots shown in this section were produced with the R package[3].

Interpreting Tables and Graphs

For every class of instances and then for the entire set of instances we verify if differences in cost of the solutions found by the algorithms are statistically significant. We use the *Pairwise Wilcoxon rank sum* test [26] with *p-values*[4] adjustment method by Holm [58]. In the tables associated to the graphics we report for every pair of algorithms (r, c) the *p-values* for the null hypothesis: "*The distributions of the solutions generated by A and by B are the same*".

The significance level with which we reject the null hypothesis is 0.99. The *p-values* smaller than 0.01 are sufficient to reject the null hypothesis in favor of the alternate hypothesis. The table features the names of the algorithms under consideration on its row *(r)* and column *(c)*. A number less than 0.01 as an entry in *(r,c)* represents the condition that the observed performance of the algorithms r and c has a sufficient statistical difference to reject the *null hypothesis* at a confidence level of at least 99%. The symbol * as an entry in *(r,c)* of the table refers to the condition that the algorithms are same $r{=}c$ whereas - as an entry in *(r,c)* represents that the comparison is already made in the form of *(c,r)*.

We represent the performance of the algorithms in *x-y plots* and present them in two forms for each case: the absolute performance graph and its corresponding relative performance graph. In the absolute performance graph, the *x-axis* denotes the probability that the cities require being visited in the PTSP and the *y-axis* represents the expected value of the *a posteriori* tour length averaged over 100 PTSP instances. The large scale of the *y-axis* makes the visualization of the difference in the solution quality difficult. The next graph illustrates the relative difference in the solution quality of the algorithms by taking a single algorithm as a normalized reference.

Analysis of Stochasticity

The stochasticity of the *homogeneous* PTSP can be measured with other simple heuristics. In this context, the term stochasticity refers to the variance associated with the *a posteriori* tour length of the PTSP. For illustration, consider a straight forward and simple heuristic algorithm that finds the *a priori* tour for the PTSP by treating it as TSP. It is possible to measure the quality of the solution with respect to the influence of the stochasticity in the following way: The solutions computed by each algorithm on a given PTSP were evaluated on a number of random scenarios to compute the *a posteriori* tour length associated with each random scenario. The mean and variance of the *a posteriori* tour length over the set of all random scenarios under consideration describe the stochasticity associated with the problem. More precisely, the larger the variance, the higher the stochasticity associated with the problem and viceversa.

In order to determine the stochastic nature of the problem, we test the *homogeneous* PTSP problems with simple heuristics such as Nearest Neighbor, Nearest Insertion and Furthest Insertion heuristics. The Nearest Neighbor algorithm (see Algorithm 9) always visits the nearest city

[2]This software package provides an implementation of various ACO algorithms applied to the TSP. The ACO algorithms implemented are Ant System, Elitist Ant System, MAX-MIN Ant System, Rank-based version of Ant System, Best-Worst Ant System, and Ant Colony System.

[3]R is a language and environment for statistical computing and graphics. It provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. It is also available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form.

[4]The *p-value* of a test is the probability of obtaining a value more extreme than the one that is obtained, in the direction of the alternate hypothesis, if the null hypothesis is true.

whereas other algorithms first find a tour on small subset of cities, and then extends this tour by inserting the remaining cities one after the other until all the cities have been inserted. More precisely, the Nearest Insertion approach (see Algorithm 10), from all cities yet to be visited, chooses a city whose insertion causes the lowest increase in the length of the tour. On the other hand, the Furthest Insertion approach (see Algorithm 11) inserts a city whose minimal distance to the visiting city is maximal. The idea behind this strategy is to fix the overall layout of the tour in the first few iterations of the insertion process. These algorithms are simple in conception and implementation. Figure 4.1 visualizes the stochasticity associated with the uniformly distributed and clustered *homogeneous* PTSP instances. The *x-axis* denotes the probability that the cities require being visited in the *homogeneous* PTSP. The *y-axis* represents the normalized standard deviation, that is, the standard deviation divided by the mean, for the *a posteriori* tour length computed on 300 random scenarios sampled according to the corresponding *x-axis* probability.

Algorithm 9 Nearest Neighbor

1. Select a random city
2. Select the nearest unvisited city and go there
3. Repeat step 2 until no more cities remain
4. Return to the first city

Algorithm 10 Nearest Insertion

1. Select the shortest edge and make a sub tour of it
2. Select the city not in the sub tour having the shortest distance to any one of the cities in the sub tour
3. Find the edge in the sub tour such that the cost of inserting the selected city between the edge's cities will be minimal
4. Repeat step 2 until no more cites remain

Algorithm 11 Furthest Insertion

1. Select the longest edge and make a sub tour of it
2. Select the city not in the sub tour having the farthest distance to any one of the cities in the sub tour
3. Find the edge in the sub tour such that the cost of inserting the selected city between the edge's cities will be minimal
4. Repeat step 2 until no more cities remain

From the results, we infer that the variance of the *a posteriori* tour length computed for the homogeneous PTSP is less for the probability values close to 1. This indicates that in these cases, most of the cites in the PTSP need to be visited in all the sampled random scenarios. On the other hand, as the probability decreases, the number of cities that need to be visited in the generated random scenarios also decreases. As a consequence, the lesser the probability, the higher the variance of the *a posteriori* tour length. Therefore, we informally conclude that by decreasing the probability values associated with each city, the stochasticity of the homogeneous PTSP increases. These results form the basis for other experiments described in the rest of this chapter.

4.1.1 Experiments on Solution Techniques

In this subsection, we present the computational results which describe the behavior of several ACO versions under uncertainty. Figures 4.2, 4.3 describe the absolute and the relative performance of the algorithms in terms of solution quality for the computational time of 60 seconds. The values in the *y-axis* denote the expected value of the *a posteriori* tour length averaged over

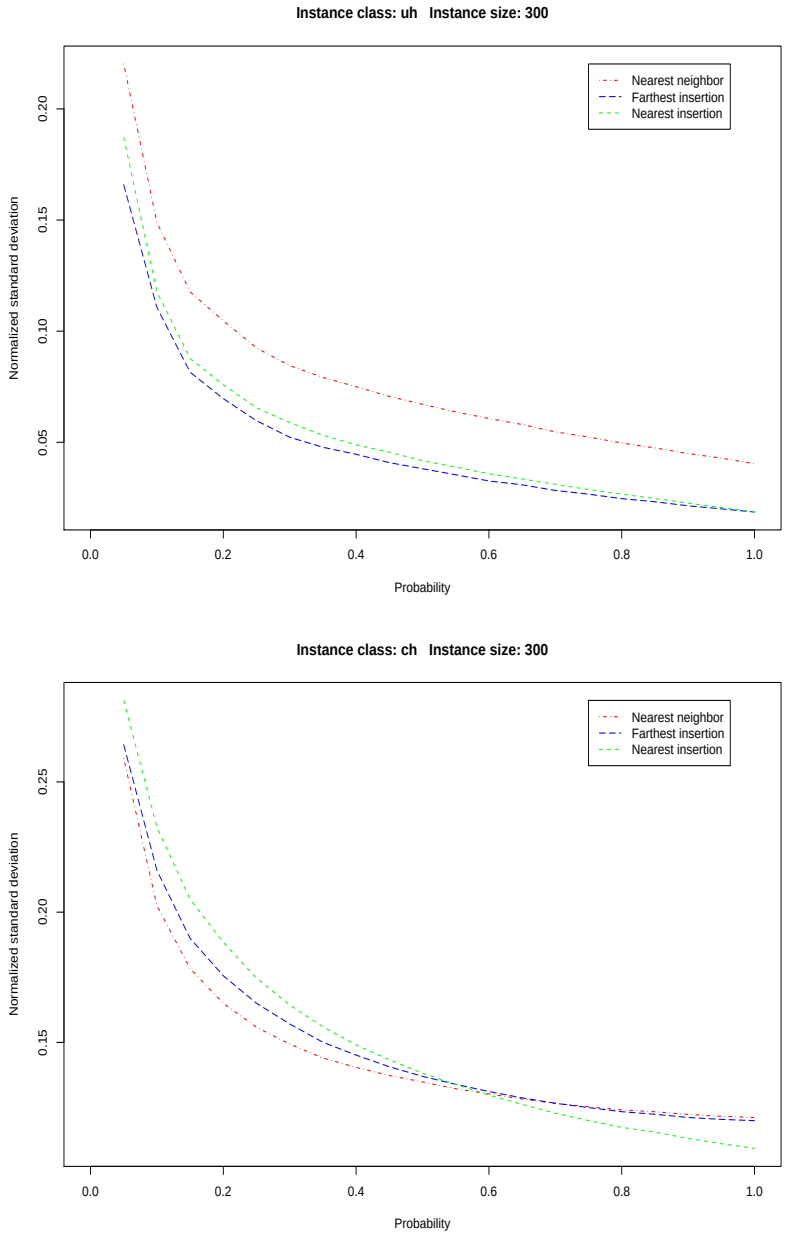

Figure 4.1: Experimental results for the stochasticity on the uniformly distributed and clustered homogeneous PTSP instances. The *x-axis* denotes the probability that the cities require being visited in the *homogeneous* PTSP. The *y-axis* represents the normalized standard deviation, that is, the standard deviation divided by mean, for the the *a posteriori* tour length computed on 300 random scenarios sampled according to the corresponding *x-axis* probability.

Table 4.1: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1.

Probability=0.25	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	$< 2.2e-16$
Probability=0.5	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=0.75	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	1
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=1.0	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1

100 *homogeneous* PTSP instances of 300 uniformly distributed cities. The values in the *x-axis* represent the corresponding probability that the cities require being visited in the *homogeneous* PTSP. Table 4.1 summarize the statistical significance for the observation, that is, the *p-values* for the Wilcoxon paired rank test for the probabilities {0.25, 0.50, 0.75, 1.0} that the cities require being visited in the PTSP.

From Figure 4.3, we can observe that the solution quality computed by ACO-1 is better than S-ACO and ACO/F-Race for the probability range greater than 0.4, that is, when the variance of the *a posteriori* tour length is small. This means that an algorithm designed to solve TSP is better than one specifically developed for PTSP. This confirms the results of the research work established by Rossi and Gavioli [86]. This is easily explained: Using large number of random scenarios for selecting the *best-so-far* solution is simply a waste of time when the variance of the objective function is very small.

On the other hand, for the probability range less than 0.4, the problem becomes more stochastic. Therefore, the *best-so-far* solution obtained using large number of samples plays a significant role. The rationale behind this technique is that unfortunate modifications to the pheromone matrix that can be caused by sampling an atypical random scenario at a given iteration, will not have a large impact on the overall result and will be corrected in following iterations. The risk we run by following a single sample strategy, in ACO-1, is that we might sample a particularly atypical random scenario which provides a misleading selection of the solutions. As a consequence, ACO/F-Race and S-ACO obtain better results than ACO-1. We can justify our observation with Table 4.3. The p value associated with ACO/F-Race(row) Vs. ACO-1(column) and S-ACO(row) Vs. ACO-1(column) is $< 2.2e-16$ for the probability range 0.0 to 0.4 and it becomes 1 for probability range 0.5 to 1.0 which refers to the condition that ACO-1 obtain better results than ACO/F-Race and S-ACO.

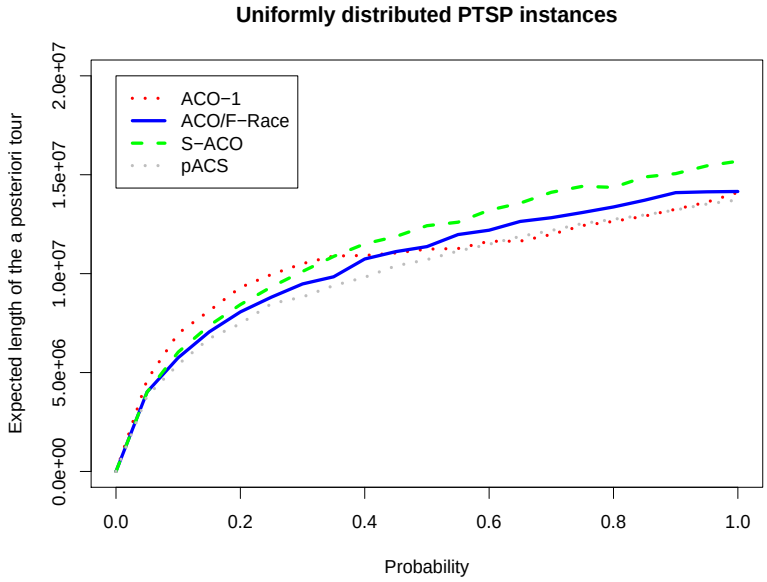

Figure 4.2: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1 for the computational time of 60 seconds.

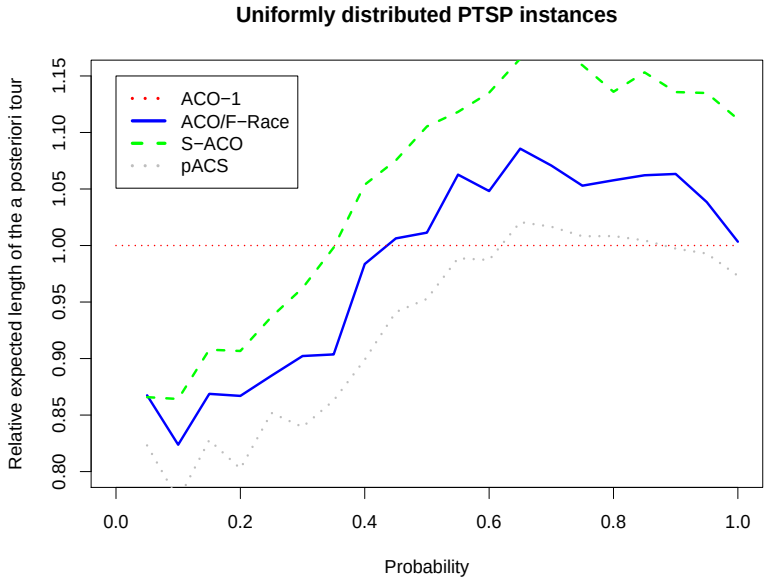

Figure 4.3: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race and S-ACO normalized by the one obtained by ACO-1 for the computational time of 60 seconds.

Another important observation concerning the results is the performance of the ACO/F-Race. Throughout the whole range of probabilities, the solution quality computed by ACO/F-Race is *significantly* better than S-ACO. Therefore, we infer that the nonparametric evaluation method adapted by ACO/F-Race computes better solution than parametric S-ACO. Notwithstanding the same number of evaluations, the S-ACO selects the *iteration-best* solution based on a single random scenario whereas ACO/F-Race use the racing approach to evaluate all the solutions obtained at each iteration. As a consequence, while selecting the *iteration-best* solution, the probability of neglecting a good solution is very high in S-ACO by sampling an atypical random scenario. In Table 4.1, for the entire probability range, the p value of the Wilcoxon test associated with ACO/F-Race(row) Vs. S-ACO(column) is $< 2.2e-16$. To ensure ACO/F-Race is significantly better than S-ACO it is enough to have the p value < 0.01.

Further insight into the results revels that pACS performs better than ACO/F-Race and S-ACO. The statistical significance for this observation is shown in Table 4.1. The *p-value* is $< 2.2e-16$ for all the probabilities in pACS(row) Vs. ACO/F-Race(column). The same holds true for pACS (row) Vs. S-ACO (column). It should be noted that pACS is fine tuned for parameter configuration and it exploits the pseudo random proportional rule[5] in the solution construction process. On the other hand, ACO/F-Race and S-ACO are not tuned for the parameter and employ the random proportional rule[6]. It is known in the literature [34] that an ACO algorithm which employs pseudo random proportional rule computes better solution than one which uses random proportional rule. Therefore, pACS computes better solution than ACO/F-Race and S-ACO. The primary goal of this experiment was to compare the evaluation procedure based on F-Race with the one proposed in S-ACO. For this reason, solution construction (with random proportional rule) was implemented as described in S-ACO [52]. In future, we will combine F-Race with better performing ACO.

Figure 4.5 shows the results obtained for the uniformly distributed *homogeneous* PTSP for the computational time of 120 seconds. We can observe a similar behavior to the results obtained for the computational time of 60 seconds. The interesting element to observe from these results is that the difference in the relative performance among different algorithms becomes less with respect to the relative performance at 60 seconds. This behavior can be justified with the property of *convergence in solution* of ACO. It is a state of the algorithm that keeps on generating the same optimal solution. ACO-F/Race and pACS reach this optimal solution in shorter computational time due to the improved technique which they adopt to select the *iteration-best* solution. On the other hand, ACO-1 and S-ACO attain this state by reaping the benefits of higher computational time. The statistical significance for this observation is given in Table 4.2.

The experimental evaluations for the homogeneous PTSP with clustered cities, for the computational time of 60 and 120 seconds are shown in Figures 4.7 and 4.9 respectively. The Wilcoxon paired test and the p values are summarized in Tables 4.3 and 4.4. From the results, we observe that the behavior of the algorithms do not change significantly with the clustered distribution. Therefore, the arguments and justifications which are explained for the uniformly distributed cities can be adapted to the clustered PTSP.

[5] See Equation 2.9 described in page 18
[6] See Equation 2.2 described in page 15

Table 4.2: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1.

Probability=0.25	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	$< 2.2e-16$
Probability=0.5	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	0.8757
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=0.75	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	0.9583
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=1.0	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1

Table 4.3: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1.

Probability=0.25	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	*	-	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	$< 2.2e-16$
Probability=0.5	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	*	-	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	1
Probability=0.75	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	1
ACO/F-Race	*	-	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=1.0	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	*	-	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	1

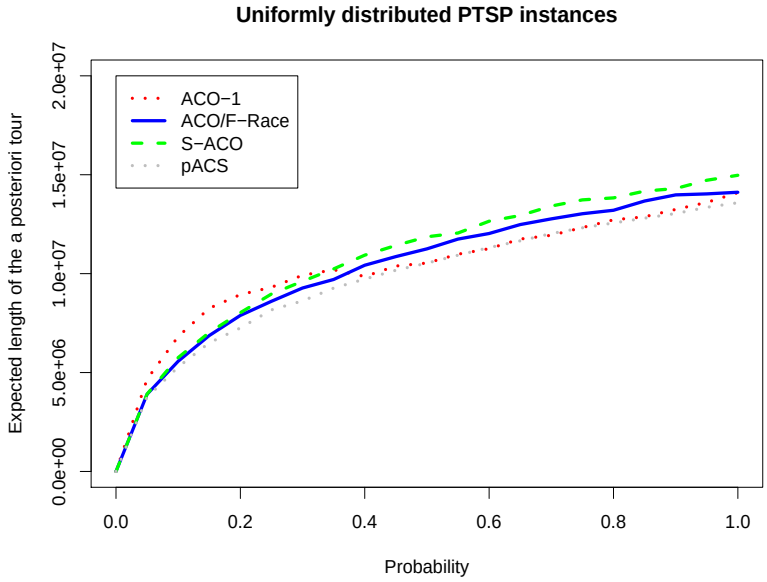

Figure 4.4: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1 for the computational time of 120 seconds.

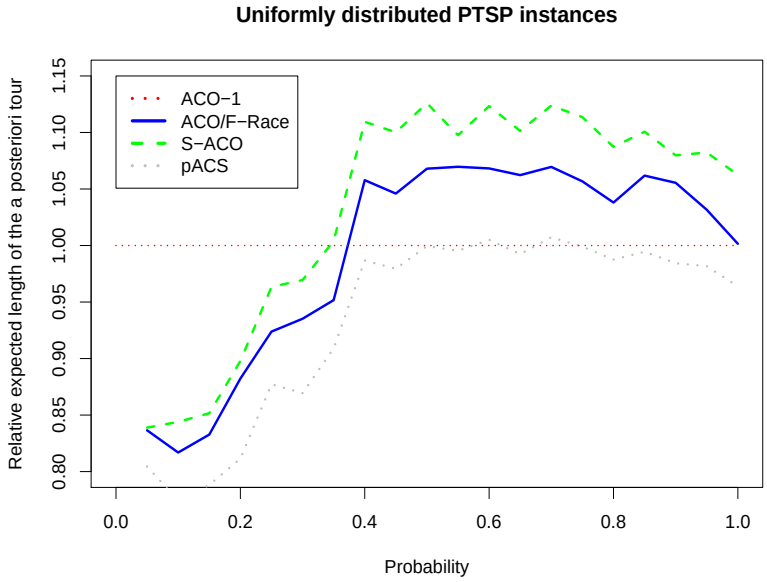

Figure 4.5: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race and S-ACO normalized by the one obtained by ACO-1 for the computational time of 120 seconds.

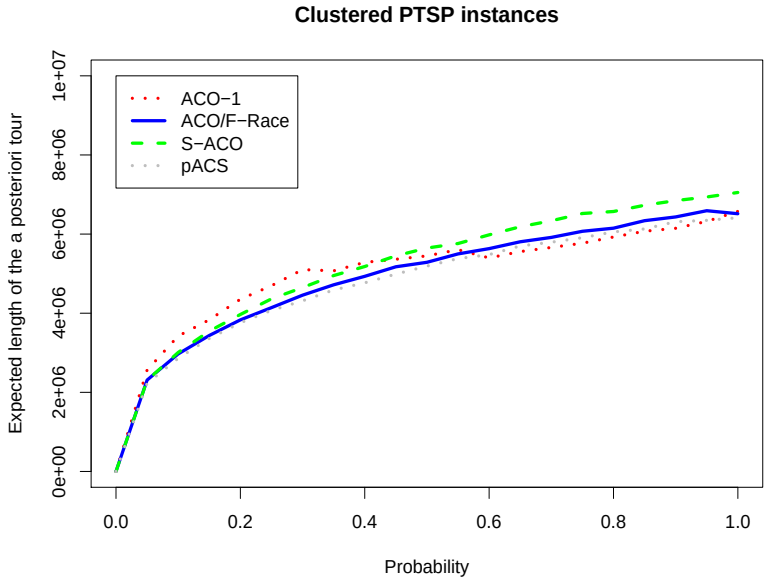

Figure 4.6: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1 for the computational time of 60 seconds.

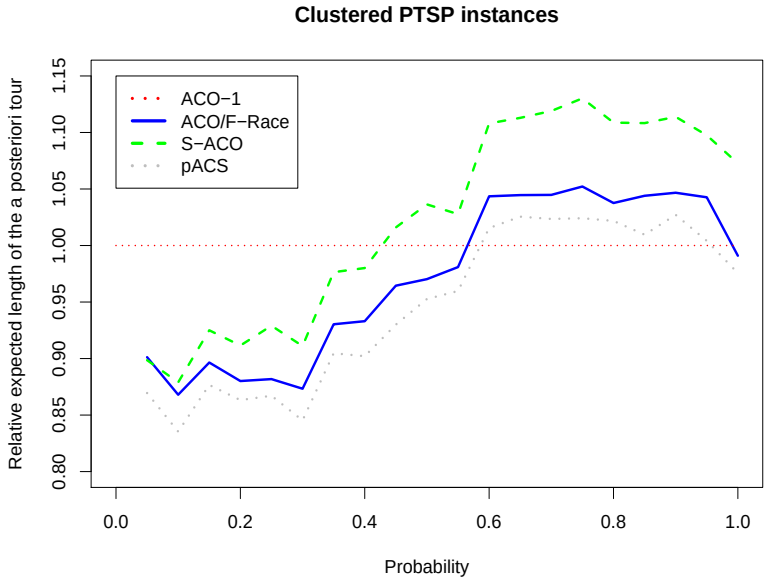

Figure 4.7: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race and S-ACO normalized by the one obtained by ACO-1 for the computational time of 60 seconds.

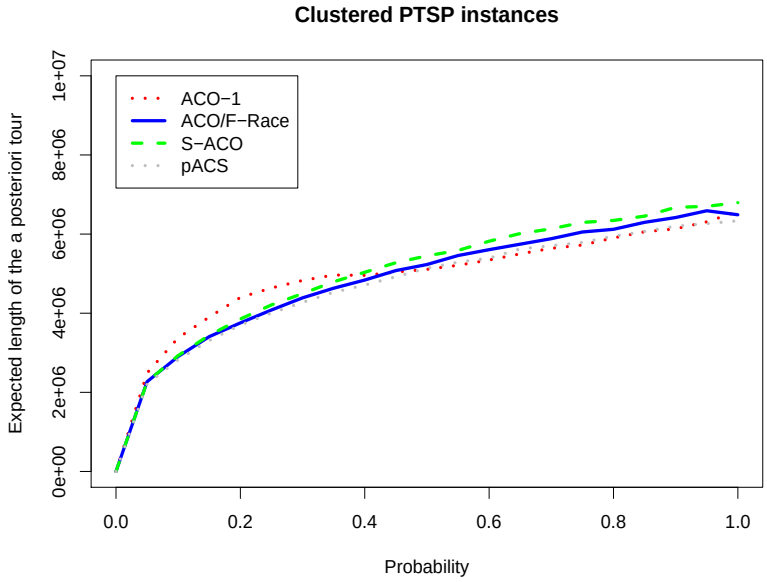

Figure 4.8: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1 for the computational time of 120 seconds.

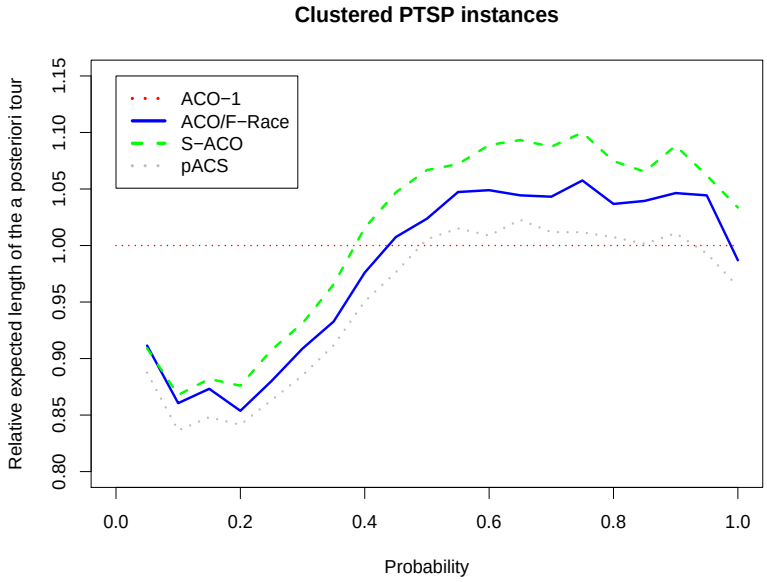

Figure 4.9: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race and S-ACO normalized by the one obtained by ACO-1 for the computational time of 120 seconds.

Table 4.4: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, S-ACO and ACO-1.

Probability=0.25	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	$< 2.2e-16$
Probability=0.5	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	1
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=0.75	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	1
ACO/F-Race	-	*	$< 2.2e-16$	1
SACO	-	-	*	1
Probability=1.0	pACS	ACO/F-Race	SACO	ACO-1
pACS	*	$< 2.2e-16$	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race	-	*	$< 2.2e-16$	$< 2.2e-16$
SACO	-	-	*	1

4.1.2 Experiments on Local Search techniques

This subsection describes the potentialities of the *empirical 2-opt and 2.5-opt* local search procedures proposed in this thesis. More precisely, we will present the computational experiments to illustrate the phenomenon and the significance of the empirical local search techniques when applied to ACO/F-Race. The second part of this subsection describes some preliminary experimental results in which we compare our local search techniques with the state-of-the-art local search approach.

Effectiveness of the empirical local search algorithms

We followed the same experimental methodology with respect to the problem type and the computational time as explained in the previous section. We examined the solution refinement phenomenon of the proposed local search approach by solving the PTSP with three different algorithms:

- ACO/F-Race
- ACO/F-Race with *empirical 2-opt* local search
- ACO/F-Race with *empirical 2.5-opt* local search

For the *homogeneous* PTSP that contains uniformly distributed cites, Figures 4.10, 4.11 describes the absolute and the relative performance of the algorithms for the computational time of 60 seconds. Table 4.5 summarize the *p-values* for the Wilcoxon paired rank test to show the statistical significance for the experimental observations.

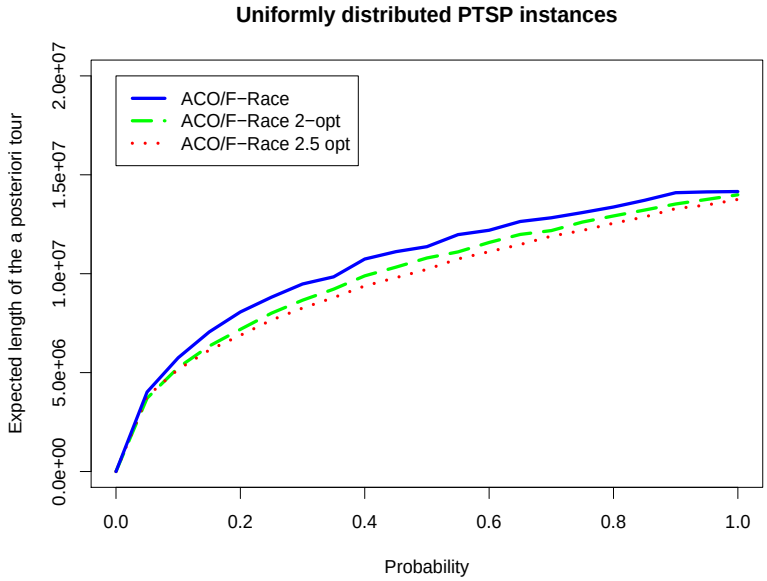

Figure 4.10: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* for the computational time of 60 seconds.

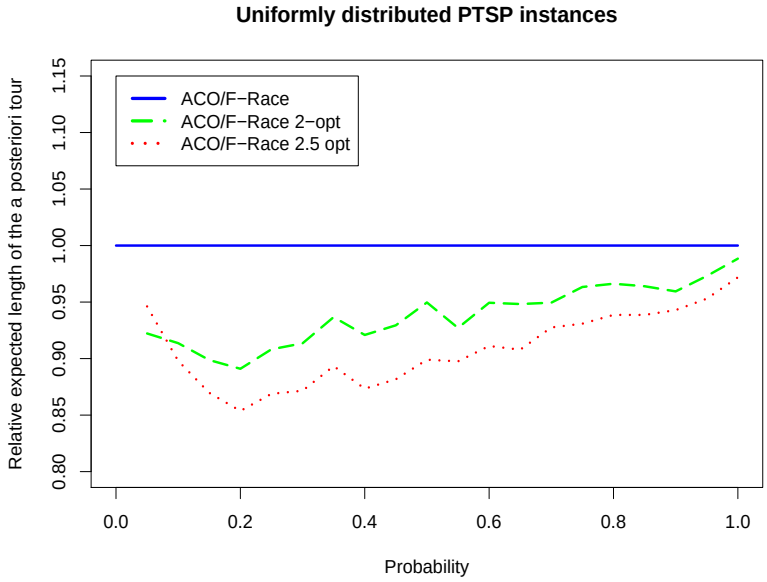

Figure 4.11: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* normalized by the one obtained by ACO/F-Race for the computational time of 60 seconds.

Table 4.5: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt*.

Probability=0.25	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.75	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$

The results from Figure 4.11 clearly show that the *empirical 2-opt and 2.5-opt significantly* increase the quality of the solution computed by the ACO/F-Race for all the probability values. This is easily explained: The *empirical 2-opt* local search approach evaluates the improvement in the expected length of the *a posteriori tour* that results from reversing the tour between each pair of nodes. The *empirical 2.5-opt* local search computes the improvement by removing each city from its current position in the tour and inserting it at all other points in the tour. As a consequence, the newly computed solution is always better. Table 4.5 provides statistical significance for the observation.

Further insight into the results reveals that the solution quality of *empirical 2.5-opt* is *significantly* better than *empirical 2-opt* for probabilities greater than 0.1. The superior performance of the *empirical 2.5-opt* local search can be justified by the technique employed for the solution refinement. For the same set of nearest neighbors, the *2.5-opt* chooses the best swap when compared to *2-opt* [92]. The same argument holds true for the empirical local search too. But the poor performance of *2.5-opt* for the probabilities from 0.0 to 0.1 is surprising. Here the problem becomes highly stochastic. In *2.5-opt*, the search for the best swap with very few cities using large number of samples results in a waste of computational time. On the other hand, *2-opt* move contributes this extra time to the ACO/F-Race to perform more iterations which results in better quality solutions.

Another interesting observation from this result is the difference in the relative performance of the algorithms. The relative difference of solutions computed by local search versions of the ACO/F-Race (with respect to ACO/F-Race) decreases with the increase of the probability associated with the *homogeneous* PTSP. For highly stochastic problems, using large samples to estimate the improvement measure enables empirical local search to compute better quality solutions. It is simply a waste of time when the problem is less stochastic.

Figures 4.12, 4.13 show the results obtained for the uniformly distributed cities in the *homogeneous* PTSP for the computational time of 120 seconds. Table 4.6 lists the *p-values* for the

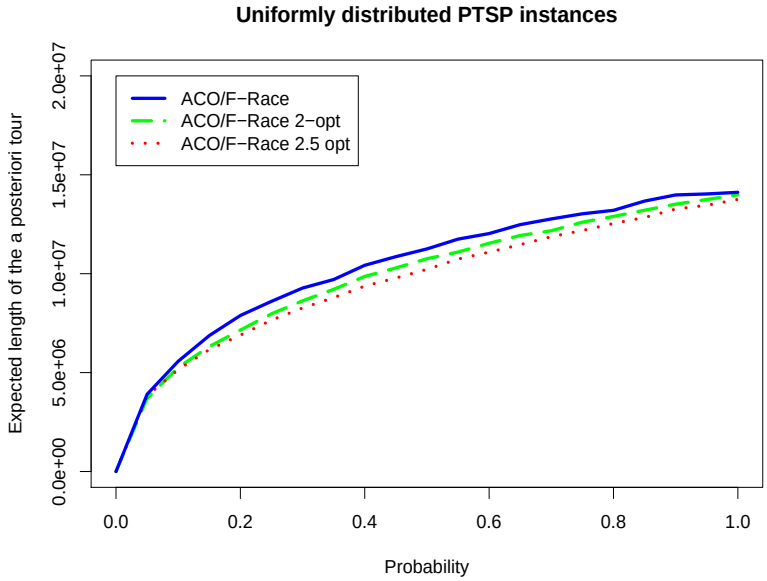

Figure 4.12: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* for the computational time of 120 seconds.

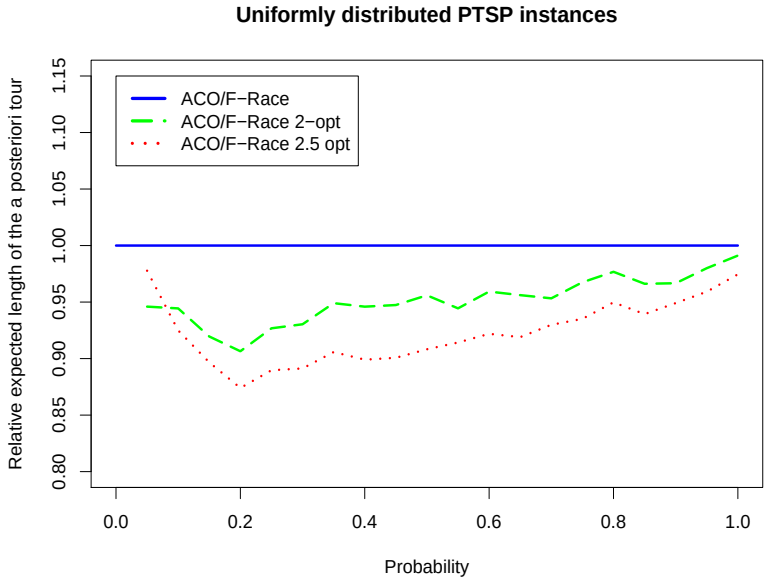

Figure 4.13: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* normalized by the one obtained by ACO/F-Race for the computational time of 120 seconds.

Table 4.6: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt*.

Probability=0.25	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.75	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$

Wilcoxon paired rank test for this observation. The only difference that can be found in Figure 4.13 is that the difference in relative performance among different algorithms becomes less with respect to the relative performance at 60 seconds due to the property of *convergence in the solution*.

The experimental evaluations for the homogeneous PTSP with clustered cities, for the computational time of 60 and 120 seconds are shown in Figures 4.14, 4.15 and 4.16, 4.17 respectively. The Wilcoxon paired test and the p values are summarized in Tables 4.7 and 4.8. The behavior of the algorithms is similar to the uniformly distributed instances. Therefore, the arguments and justifications which are explained for the uniformly distributed cities can be adapted to the clustered PTSP.

Preliminary Results Towards the State-of-the-Art Perspective

In this subsection, we present the results of the computational experiments which compare the ACO/F-Race *empirical 2-opt and 2.5-opt* local search against pACS *1-shift* local search algorithm. The *1-shift* local search is a solution improvement technique that computes the improvement measure using *mathematical approximation* by removing each city from its current position in the tour and inserting it at all other points in the tour (in the same way as that of *2.5-opt* move). The experimental setup was same as stated before.

The absolute and relative behavior of the algorithms, on the uniformly distributed instances, for the computational time of 60 seconds, are shown in Figures 4.18 and 4.19 respectively. Table 4.9 contains the *p values* of the Wilcoxon paired test for the observation.

Table 4.7: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt*.

Probability=0.25	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.75	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$

Table 4.8: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt*.

Probability=0.25	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=0.75	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	ACO/F-Race 2-opt	ACO/F-Race
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
ACO/F-Race 2-opt	-	*	$< 2.2e-16$

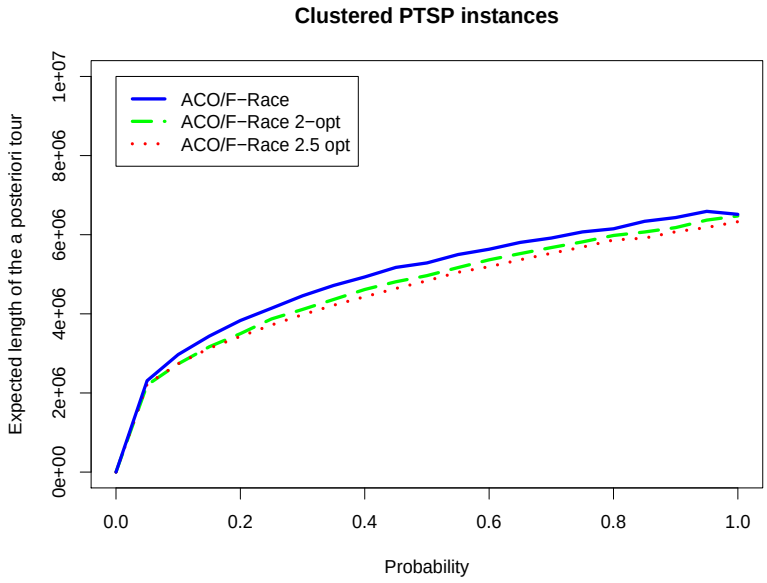

Figure 4.14: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* for the computational time of 60 seconds.

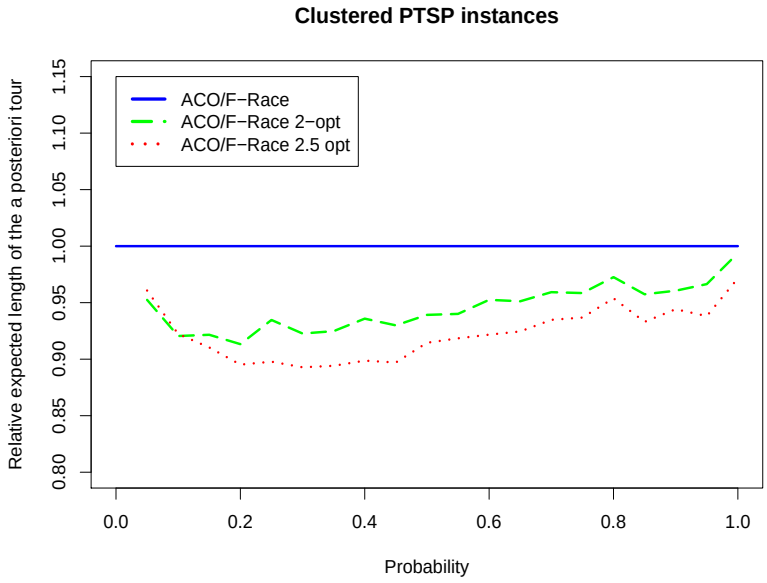

Figure 4.15: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* normalized by the one obtained by ACO/F-Race for the computational time of 60 seconds.

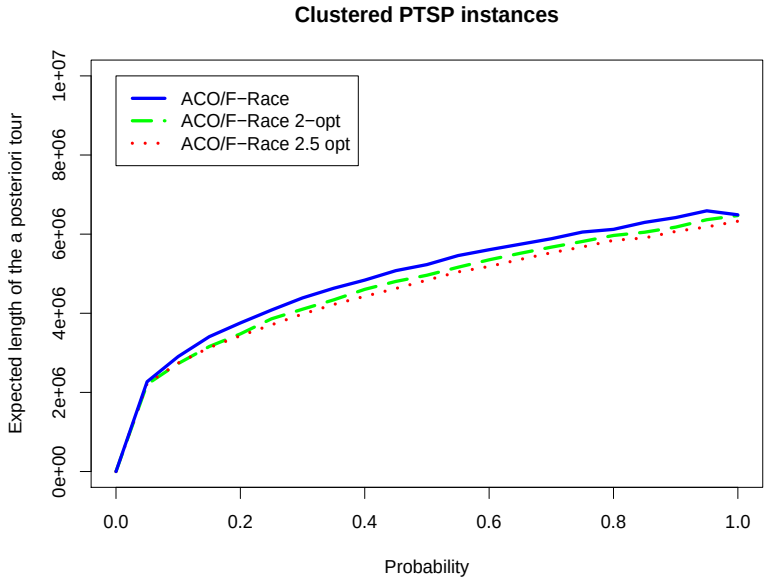

Figure 4.16: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race, ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* for the computational time of 120 seconds.

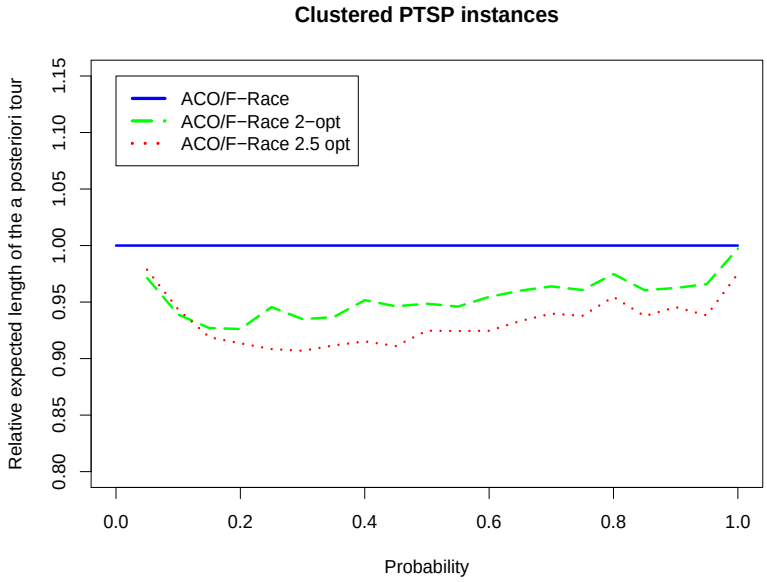

Figure 4.17: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and ACO/F-Race empirical *2.5-opt* normalized by the one obtained by ACO/F-Race for the computational time of 120 seconds.

Table 4.9: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift*

Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$

The results from Figure 4.19 clearly show that ACO/F-Race with *empirical 2.5-opt* local search performs better than the other two algorithms for the probability values from 0.15 to 1.0. We found, much to our surprise, that the solution quality of pACS *1-shift* algorithm is *significantly* worse than the quality of the solution computed by ACO/F-Race with *empirical 2.5-opt* local search. From these results, we conclude that the evaluation method adapted by empirical approach is *significantly* better than the mathematical approximation. On the other hand, when the problem is highly stochastic (for probability value less than 0.15) the quality of the solution computed by ACO/F-Race *empirical 2.5-opt* local search is worse than pACS *1-shift* approach. As established with previous experiments for the *2.5-opt*, the search for the best swap with very few cities in the large samples results in wasted computational time. On the other hand, *1-shift* use this computational time to perform more recursions and iterations. For the computational time of 120 seconds, it is clear from Figure 4.21 that the solution quality of pACS *1-shift* is *significantly* better than the ACO/F-Race *empirical 2-opt* local search. The pACS exploit the pseudo random proportional rule which favors towards high quality solution. The *1-shift* local search takes place in such high quality solutions to obtain better solutions than ACO/F-Race *empirical 2-opt*. On the other hand, the pACS *1-shift* algorithm exhibits different behavior with the clustered instances. It is more evident in Figure 4.25 where the difference in the relative performance between the pACS *1-shift* and ACO/F-Race with *empirical 2-opt* is minimal. We derive the justification for this behavior from the research work made by Bertsimas and Howell [9] where they concluded that *1-p shift* local search scheme works well only for the uniformly distributed data set.

Table 4.10: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for uniformly distributed homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift*

Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$

Table 4.11: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 60 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift*

Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$

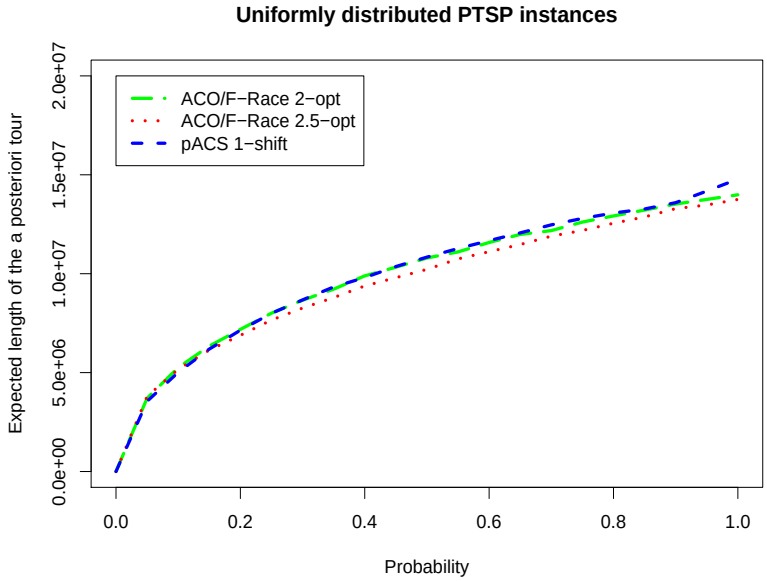

Figure 4.18: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift* for the computational time of 60 seconds.

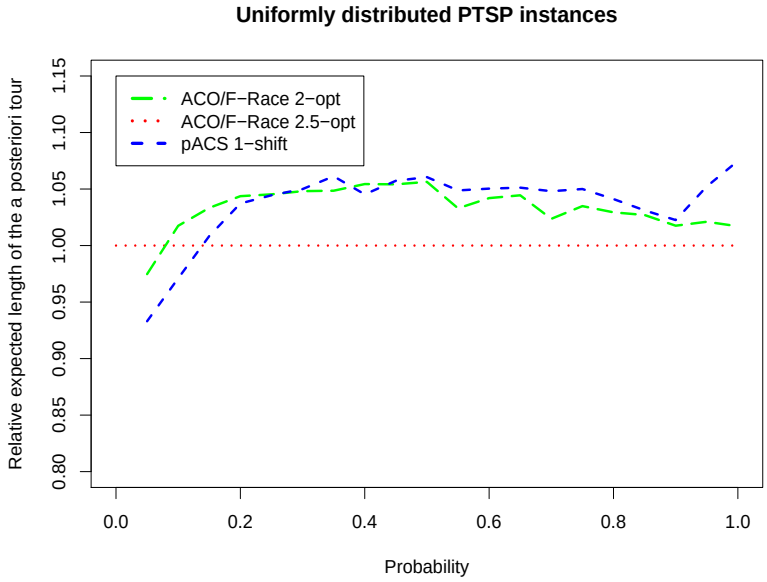

Figure 4.19: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and pACS *1-shift* normalized by the one obtained by ACO/F-Race empirical *2.5-opt* for the computational time of 60 seconds.

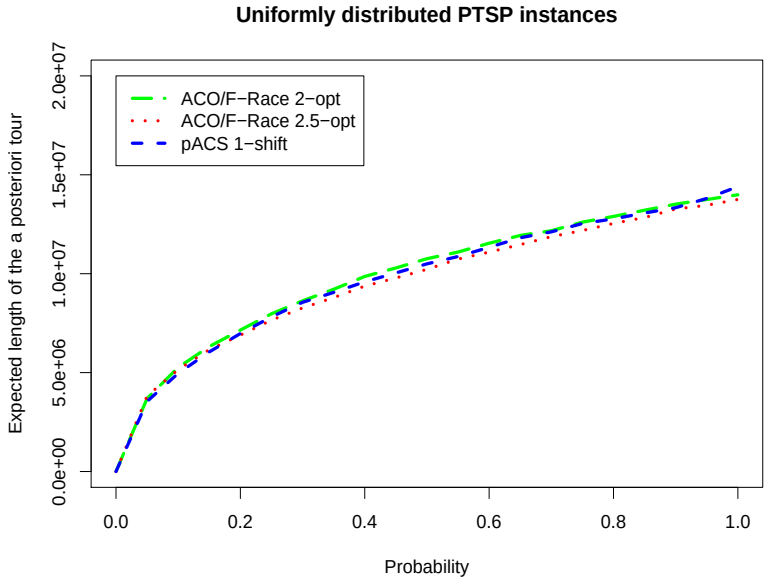

Figure 4.20: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift* for the computational time of 120 seconds.

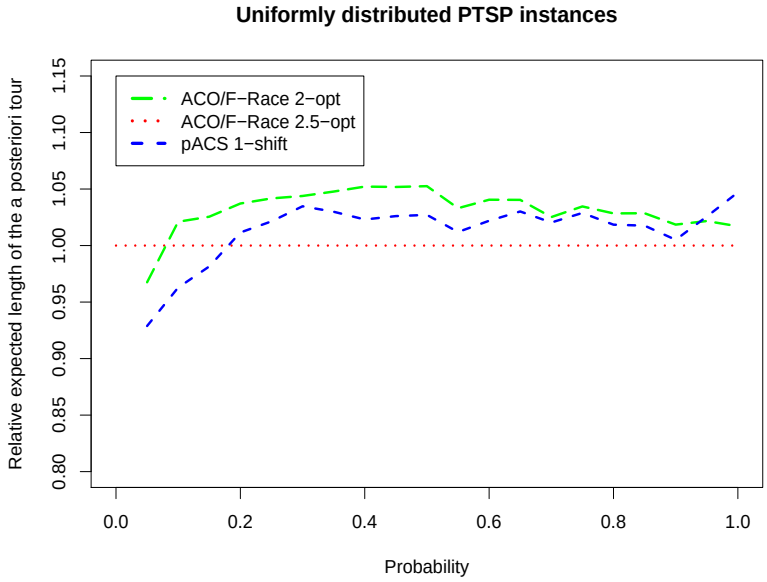

Figure 4.21: Experimental results on the uniformly distributed homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and pACS *1-shift* normalized by the one obtained by ACO/F-Race empirical *2.5-opt* for the computational time of 120 seconds.

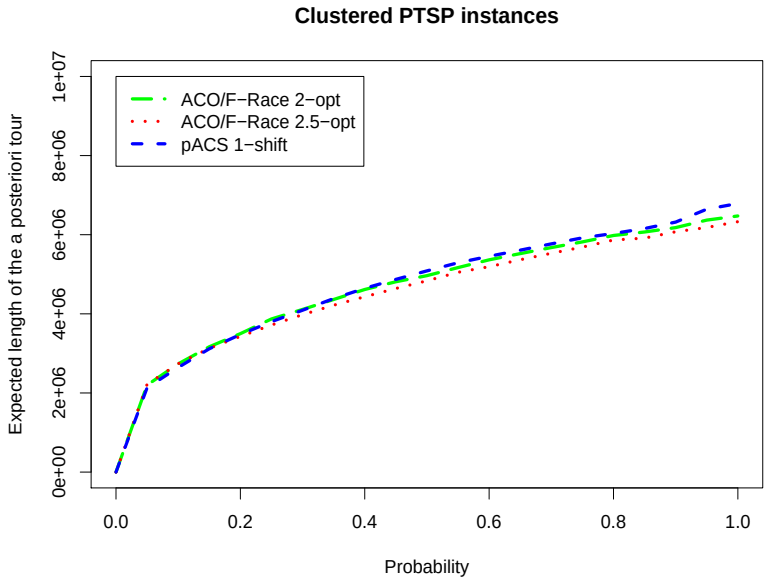

Figure 4.22: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift* for the computational time of 60 seconds.

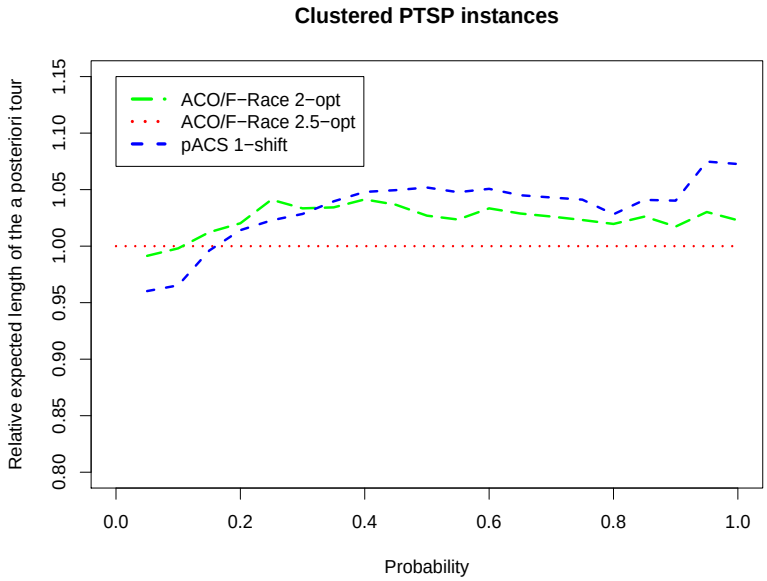

Figure 4.23: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and pACS *1-shift* normalized by the one obtained by ACO/F-Race empirical *2.5-opt* for the computational time of 60 seconds.

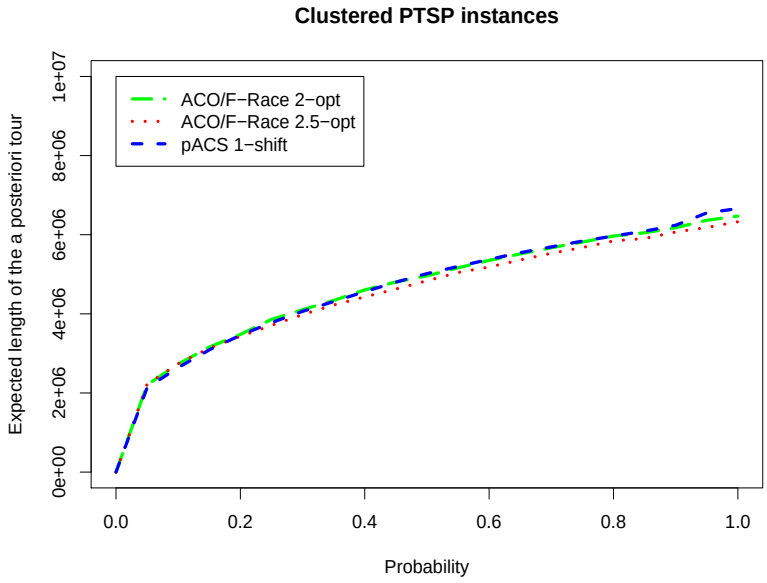

Figure 4.24: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift* for the computational time of 120 seconds.

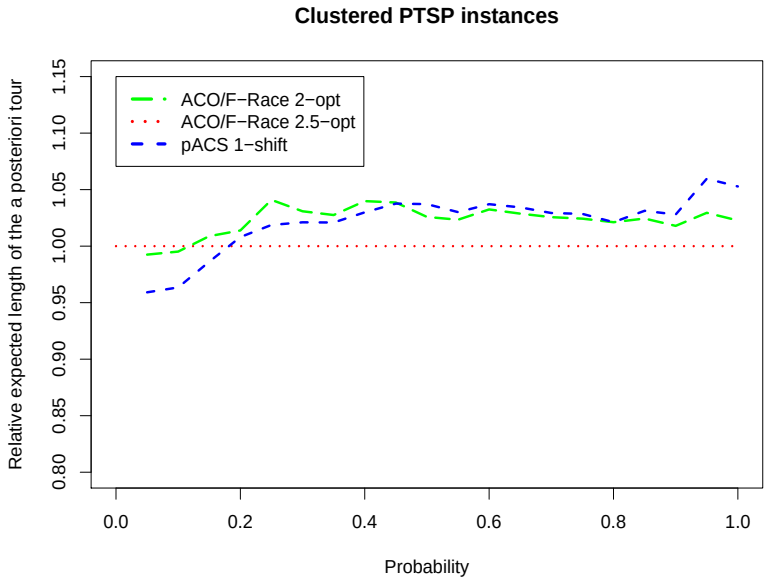

Figure 4.25: Experimental results on the clustered homogeneous PTSP. The plot represents the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt* and pACS *1-shift* normalized by the one obtained by ACO/F-Race empirical *2.5-opt* for the computational time of 120 seconds.

Table 4.12: The Wilcoxon paired test and p-values for the null hypothesis: *The distributions of the solutions are the same* for clustered homogeneous PTSP for the computational time of 120 seconds. The significance level with which we reject the null hypothesis is 0.99. p-values smaller than 0.01 are sufficient to reject the null hypothesis. The quantities under analysis are the expected length of the *a posteriori* tour obtained by ACO/F-Race empirical *2-opt*, ACO/F-Race empirical *2.5-opt* and pACS *1-shift*

Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.50	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=0.25	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$
Probability=1.0	ACO/F-Race 2.5-opt	pACS 1-shift	ACO/F-Race 2 opt
ACO/F-Race 2.5-opt	*	$< 2.2e-16$	$< 2.2e-16$
pacs 1-shiftt	-	*	$< 2.2e-16$

Chapter 5

Conclusion and Future Work

This chapter presents the summary of the thesis with the contribution of the proposed approach followed by the conclusion of the thesis, suggestions for the improvement and future work.

Summary

In a large number of real-world combinatorial optimization problems, the objective function is affected by uncertainty. In order to tackle these problems, it is customary to resort to a probabilistic model of the value of each feasible solution. In other words, a setting is considered in which the cost of each solution is a random variable, and the goal is to find the solution that minimizes some statistics of the latter. For a number of practical and theoretical reasons, it is customary to optimize with respect to the expectation. For a given probabilistic model, the expectation can always be computed but this typically involves particularly complex analytical manipulations and computationally expensive procedures. Two alternatives have been discussed in the literature: analytical approximation and empirical estimation. While the former explicitly relies on the underlying probabilistic model for approximating the expectation, the latter estimates the expectation through sampling or simulation. Computing a profitable approximation is a problem specific issue and requires a deep understanding of the underlying probabilistic model. The main advantage of the estimation approach over the one based on approximation is generality: Indeed, a sample estimate of the expected cost of a given solution can be simply obtained by averaging a number of realizations of the cost itself.

We adopted the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) as a test-bed for stochastic optimization problems, in much the same way as the TRAVELING SALESMAN PROBLEM (TSP) has been considered a standard amongst deterministic optimization problems.

Metaheuristics is a general algorithmic framework to solve different optimization problems. They compute fairly good solution in a reasonable amount of computational time. Ant colony optimization is a metaheuristic which takes inspiration from foraging behavior of ant colonies.

We have developed a computational technique called ACO/F-Race algorithm for tackling highly stochastic PTSP. The proposed approach is inspired by S-ACO [52] and F-Race [18, 17]. The former is an ACO algorithm based on simulation/sampling techniques to tackle the stochastic optimization problems and the later is an algorithmic framework originally developed for tuning metaheuristics. F-Race is itself inspired by a class of racing algorithms proposed in the machine learning literature for tackling model selection problem. The peculiarity of the F-Race approach compared to other racing algorithms is the adoption of the *Friedman two-way analysis of variance by ranks*, a nonparametric statistical test that appears particularly suitable for the highly stochastic PTSP.

In general, ACO algorithms frequently encounter a sequence of states in which it is impossible to improve the solution quality by itself. Local search algorithms improve the solutions computed by ACO by introducing some modifications [92]. The state-of-the-art local search procedures for PTSP such as *1-shift* and *2-p-opt* local search are based on complex mathematical approximations. We proposed *empirical 2-opt* and *empirical 2.5-opt* for PTSP inspired by *2-opt* and *2.5-opt* local search algorithms which are explicitly designed to solve TSP. We derived computationally inexpensive equations to compute the improvement in the objective function for the proposed improvement techniques. As a consequence, the solution computed by ACO/F-Race algorithm is subjected to several refinements in a short computational time to obtain a high quality solution.

Conclusion

The experimental results proposed in the subsection 4.1.1 confirm the generality of the approach proposed in Gutjahr [52]. More precisely, from the computational experiments, we conclude that it is important to use an ACO technique specifically developed for PTSP when the probability associated with each city in the PTSP is *significantly* less than 1. This confirms the results established by Rossi and Gavioli [86] and Bertsimas and Howell [9]. In the experimental analysis proposed in the subsection 4.1.1, the goal was to compare the evaluation procedure based on F-Race with the one proposed in S-ACO [52] and with the trivial one based on a single sample ACO-1. For this reason, solution construction and pheromone update were implemented as described in [52]. As a consequence, ACO/F-Race exhibits inferior solution quality than the state-of-the-art pACS algorithm [13]. On the other hand, ACO/F-Race computes *significantly* better solutions than S-ACO by reaping the benefits of the nonparametric selection phenomenon of the F-Race approach. The general conclusions that we can draw from these results are:

- ACO/F-Race algorithm is extremely useful when the stochasticity associated with the PTSP is significantly large as suggested by Rossi and Gavioli [86].
- We propose to use ACO algorithms designed for TSP to solve PTSP when the stochasticity associated with the problem is small. This is because the computational time is wasted by the evaluation method adopted by ACO/F-Race (which results in inferior solution quality).

The experimental results on the proposed local search procedures such as *empirical 2-opt* and *empirical 2.5-opt* reveal that the solution computed by ACO/F-Race can be significantly improved by coupling it with the empirical local search. The inexpensive computational time of the proposed improvement procedures enables ACO/F-Race to have several refinements to produce significantly high quality solutions. The results from the preliminary experiments with the state-of-the-art local search procedure indicate that the *empirical estimation* based evaluation method is more profitable than *mathematical approximation*. Therefore, in the context of PTSP, we conclude the following:

- ACO/F-Race can be employed to identify good quality solutions in the search space. The empirical local search techniques operate on these solutions to compute *significantly* better quality solutions.
- Local search approach which uses *empirical estimation* can obtain better solutions than *mathematical approximation*.

Finally, We conclude that the ACO/F-Race algorithm and the empirical estimation based local search can be profitably adopted for computing high quality solutions in the framework of applications of ant colony optimization to combinatorial optimization problems under uncertainty.

Original Contributions

The original contributions of this DEA thesis are:

- We proposed the ACO/F-Race algorithm inspired by F-Race [18, 17] and S-ACO [52] as the computational technique for computing quality solutions in the framework of ant colony optimization algorithms when applied to combinatorial optimization problems under uncertainty. More precisely, the novelty of the proposed technique lies in the *nonparametric* approach for selecting the *best-so-far* ant, that is, the ant that is appointed for updating the pheromone matrix. In contrast, S-ACO employ the *parametric* procedure for the same. ACO/F-Race algorithm for optimization under uncertainty (Birattari et al. [19]) has been presented at the Sixth Metaheuristics International Conference 2005.

- pACS [13], the state-of-the-art ACO algorithm designed to solve the PTSP, is based on *mathematical approximations* whereas the proposed ACO/F-Race approach is based on *empirical estimation*. This thesis described an experimental evaluation method to analyze the empirical estimation against mathematical approximation techniques.

- Local search methods are used in general to improve the solution found by ACO algorithms. The state-of-the-art local search methods for the PTSP are based on mathematical approximations and recursions [8, 12]. In this thesis, we proposed *empirical 2-opt and 2.5-opt* local search for the PTSP which is based on empirical estimation. This approach, to the best of our knowledge, is the first local search technique that uses empirical estimation to compute the objective function. This thesis also proposed a comparative analysis of the estimation approach against mathematical approximation.

Future work

Further research is needed for properly assessing the quality of ACO/F-Race. We plan to study the behavior of ACO/F-Race on non-homogeneous problems. We intend to enrich the solution construction phase of the ACO/F-Race by exploring other possibilities, such as construction and update as defined in $\mathcal{MAX} - \mathcal{MIN}$ ant system [91].

In the context of local search techniques, we are currently developing *empirical 3-opt* for PTSP inspired by the *3-opt* local search for TSP [92]. Preliminary experiments with this local search technique indicate that significant improvements in the solution quality can be attained with the proposed approach.

We plan to design a unified framework which contains the ACO/F-Race algorithm enriched by empirical estimation based local search procedures such as *empirical 2-opt, 2.5-opt and 3-opt* local search procedures.

Applications to other problems, in particular of the vehicle routing class, will be considered, too.

Bibliography

[1] E.H.L. Aarts, J.H.M. Korst, and P.J.M. vanLaarhoven. Simulated Annealing. In *Local Search in Combinatorial Optimization*, pages 91–120. John Wiley & Sons, Chichester, UK, 1997.

[2] M.H. Alrefaei and S. Andradottir. A modification of the stochastic ruler method for discrete stochastic optimization. *European Journal of Operational Research*, 133:160–182, 2001.

[3] R.C. Ball, T.M Fink, and N.E Bowler. Stochastic Annealing. Accepted in Physical Review letter and unpublished. Available from: `http://arXiv.org/abs/cond-mat/0301179`, 2002.

[4] J.J. Bartholdi and L.K. Platzman. An $O(n \log n)$ planar travelling salesman heuristic based on spacefilling curves. *Operations Research Letters*, 1:121–125, 1982.

[5] J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

[6] O. Berman and D. Simchi-Levi. Finding optimal a priori tour and location of traveling salesman with nonhomogenous customers. *Trasnportation Science*, 22:148–154, 1988.

[7] D. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40: 574–585, 1992.

[8] D. Bertsimas. *Probabilistic Combinatorial Optimization Problems*. PhD thesis, MIT, Cambridge, MA, 1988.

[9] D. Bertsimas and L. Howell. Further results on the probabilitsic traveling salesman problem. *European Journal of Operations Research*, 65:68–95, 1993.

[10] D. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Operations Research*, 38: 1019–1033, 1990.

[11] D. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic routing problems. *Transportation Science*, 29:342–352, 1995.

[12] L. Bianchi and A. Campbell. Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. Technical report, IDSIA, 2004.

[13] L. Bianchi, L. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic travelling salesman problem by the ACO metaheuristic. In G. Di Caro M. Dorigo and M. Sampels, editors, *ANTS-III*, volume 2463 of *LNCS*, pages 176–187, Berlin, Germany, 2002. Springer-Verlag.

[14] L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the vehicle routing problem with stochastic demands. In X. Yao, E. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. Rowe, P. Tino, A. Kabán, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, 8th International Conference, PPSN VIII*, volume 3242 of *LNCS*, pages 450–460, Berlin, Germany, 2004. Springer-Verlag.

[15] L. Bianchi, J. Knowles, and N. Bowler. Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operational Research*, 162:206–219, 2005.

[16] P. Billingsley. *Probability and Measure*. John Wiley & Sons, New York, NY, USA, second edition, 1986.

[17] M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[18] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, San Francisco, CA, USA, 2002. Morgan Kaufmann.

[19] M. Birattari, P. Balaprakash, and M. Dorigo. ACO/F-Race: Ant colony optimization and racing techniques for combinatorial optimization under uncertainty. In R. F. Hartl, editor, *MIC 2005: The 6th Metaheuristics International Conference*, Vienna, Austria, 2005.

[20] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[21] A. M. Campbell. Aggregation for the probabilistic traveling salesman problem. Unpublished, Available from:
`http://www.myweb.uiowa.edu/acampbll/aggregation`, 2004.

[22] D.J. Cavicchio. *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, Ann Arbor, MI, 1970.

[23] V. Cerný. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[24] P. Chardaire, J.L. Lutton, and A. Sutter. Thermostatical persistency: a powerful improving concept for simulated annealing algorithms. *European Journal of Operational Research*, 86: 565–579, 1995.

[25] S. Chien, J. Gratch, and M. Burl. On the efficient allocation of resources for hypothesis evaluation: A statistical approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):652–665, 1995.

[26] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, third edition, 1999.

[27] T. Homem de Mello. Variable sample methods for stochastic optimization. *ACM Trans. on Modelling and Computer Simulation*, 13:108–133, 2003.

[28] A. Dean and D. Voss. *Design and Analysis of Experiments*. Springer-Verlag, New York, NY, USA, 1999.

[29] K.A. DeJong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[30] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behaviour*, 3:159–168, 1990.

[31] M. Dorigo. *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale*. PhD thesis, Politecnico di Milano, Milan, Italy, 1992.

[32] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, UK, 1999.

[33] M. Dorigo and L.M. Gambardella. Ant Colony System: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1): 53–66, April 1997.

[34] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[35] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[36] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[37] W. Feller. *An introduction to Probability Theory and its Applications*. John Wiley & Sons, New York, NY, 1968.

[38] J.M. Fitzpatrick and J.J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1998.

[39] M. Fleischer. Simulated Annealing: past, present and future. In C. Alexopoulos, K. Kang, W. R. Lilegdon, and G. Goldsam, editors, *Proceedings of the 1995 Winter Simulation Conference*, pages 155–161, 1995.

[40] T. A. Foe and M.G.C. Resende. A probabilisitc heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[41] T. A. Foe and M.G.C. Resende. Greedy randomized adaptive search procedures. *Jouranl of Global Optimization*, 6:109–133, 1995.

[42] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, New York, NY, 1966.

[43] L.J. Fogel. Toward inductive inference automata. In *Proceedings of the International Federation for Information Processing Congress*, pages 395–399, Munich, Germany, 1962.

[44] M. Fu. Optimization for simulation: theory vs practice. *INFORMS Journal on Computing*, 14:192–215, 2002.

[45] M.R. Garey and D.S. Johnson. *Computers and Intractability. A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, NY, USA, 1979.

[46] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.

[47] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, second edition, 1997.

[48] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[49] P. I. Good. *Resampling Methods*. Birkhauser, Boston, MA, USA, second edition, 2001.

[50] J. Gratch, S. Chien, and G. DeJong. Learning search control knowledge for deep space network scheduling. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 135–142, San Francisco, CA, USA, 1993. Morgan Kaufmann.

[51] W.J. Gutjahr. A converging ACO algorithm for stochastic combinatorial optimization. In A.Albrecht and K.steinhofl, editors, *Stochastic Algorithms: Foundations and Applications*, volume 2827, pages 10–25. SAGA 2003, LNCS, 2003.

[52] W.J. Gutjahr. S-ACO: An ant based approach to combinatorial optimization under uncertainity. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *ANTS-IV*, volume 3172 of *LNCS*, pages 238–249, Berlin, Germany, 2004. Springer-Verlag.

[53] W.J. Gutjahr and G. Pflug. Simulated Annealing for Noisy Cost Functions. *Journal of Global Optimization*, 8:1–13, 1996.

[54] P. Hansen and N. Mladenović. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, pages 433–458. Kluwer Academic Publisher, Boston, MA, USA, 1999.

[55] K. Helsgaum. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

[56] A. Hertz and D. Kobler. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126:1–12, 2000.

[57] J.H. Holland. *Adaptation in natural artificial systems.* University of Michigan Press, Ann Arbor, MI, 1975.

[58] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[59] L. Ingber. Adaptive simulated annealing (asa): Lessons learned. *Control and Cybernetics - Special Issue on Simulated Annealing Applied to Combinatorial Optimization*, 25(1):33–54, 1996.

[60] P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, MIT, Cambridge, MA, 1985.

[61] P. Jaillet. A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36(6):929–936, 1988.

[62] P. Jaillet and A. Odoni. The probabilistic vehicle routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies.* Elsevier, Amsterdam, The Netherlands, 1988.

[63] D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, Chichester, United Kingdom, 1997.

[64] D. S. Johnson, L. A. McGeoch, C. Rego, and F. Glover. 8th DIMACS implementation challenge, 2001.

[65] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[66] G. Laporte, F.V. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research,*, 42:543–549, 1994.

[67] H. Larson. *Introduction to Probability Theory and Statistical Inference.* John Wiley & Sons, New York, NY, USA, 1982.

[68] S. Lin. Computer solution of the travelling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[69] S.W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1995.

[70] O. Maron. Hoeffding races, model selection for mri classification. Master's thesis, Dept. of Electrical and Computer science, MIT, 1994.

[71] O. Maron and A.W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 59–66, San Francisco, CA, USA, 1994. Morgan Kaufmann.

[72] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A.Teller, and E.Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[73] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, USA, 1997.

[74] A.W. Moore and M.S. Lee. Efficient algorithms for minimizing cross validation error. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 190–198, San Francisco, CA, USA, 1994. Morgan Kaufmann.

[75] P. Moscatos. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, CA, 1989.

[76] H. Mühlenbein and H.-M. Voigt. Gene pool recombination in genetic algorithms. In I. H. Osman and J. P. Kelly, editors, *Proceedings of the Metaheuristics Conference*, Norwell, MA, 1995. Kluwer Academic Publishers.

[77] G.I. Nemhauser and A.L. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, NY, 1988.

[78] V. I. Norkin, Y. M. Ermoliev, and A. Ruszczynski. On optimal allocation of indivisibles under uncertainity. *Operations Research*, 46:381–395, 1998.

[79] I. Osman and G. Laporte. Metaheuristics: A Bibliography. *Annals of Operations Research*, 63:513–623, 1996.

[80] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity*. Dover Publications, Inc., New York, NY, 1982.

[81] J.F. Peckny and D.L. Miller. A staged primal-dual algorithm for finding a minimum cost perfect two-matching in an undirected graph. *ORSA Journal on Computing*, 6:68–81, 1994.

[82] N.J. Radcliffe. Formal analysis and random respectful recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms, ICGA 1991*, pages 222–229, San Mateo, CA, 1991. Morgan Kaufmann Publishers.

[83] I. Rechenberg. *Evolution strategy: Optimization of technical systems by means of biological evolution*. Fromman-Holzboog, Stuttgart, Germany, 1973.

[84] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*, volume 840 of *LNCS*. Springer-Verlag, Berlin, Germany, 1994.

[85] H. Robbins and S. Munro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[86] F. Rossi and I. Gavioli. Aspects of heuristic methods in the probabilisitc traveling salesman problem. In *Advanced School on Stochastics in Combinatorial Optimization*, pages 214–227. World Scientific, 1987.

[87] L. Shi and S. Olafsson. Nested partition method for global optimization. *Operations Research*, 48:390–407, 2000.

[88] L. Shi and S. Olafsson. New parallel randomized algorithms for the traveling salesman problem. *Computers & Operations Research*, 26:371–394, 1999.

[89] T. Stützle. ACOTSP software package provides an implementation of various ant colony optimization (ACO) algorithms applied to the symmetric traveling salesman problem (TSP). the ACO algorithms implemented are ant system, elitist ant system, max-min ant system, rank-based version of ant system, best-worst ant system, and ant colony system., 2002.

[90] T. Stützle. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications.* PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1998. Published in 1999 - Infix, Sankt Augustin, Germany - volume 220 of DISKI.

[91] T. Stützle and H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer System*, 16(8):889–914, 2000.

[92] T. Stützle and H. Hoos. *Stochastic Local Search: Foundations and Applications.* Morgan Kaufmann, 2005.

[93] G. Syswerda. Simulated crossover in genetic algorithms. In *Proceedings of the second workshop on Foundations of Genetic Algorithms*, pages 239–255, San Mateo, CA, 1993. Morgan Kaufmann Publishers.

[94] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, 24:289–333, 2002.

[95] S. Voss, S. Martello, I. Osman, and C. Roucairol. *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers, Boston, MA, 1999.

[96] S.M Weiss and C. Kulikowski. *Computer systems that learn Classification and prediction methods from statistics neural nets machine learning and expert systems.* Morgan Kaufmann, 1991.

[97] D. Yan and H. Mukai. Stochastic discrete optimization. *SIAM Journal on Control and Optimization*, 30:594–612, 1992.