New approaches for solving the Probabilistic Traveling Salesman Problem

by

Leonora Bianchi
Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
lbianchi@ulb.ac.be

supervised by

Marco Dorigo
Maître de Recherches du FNRS
Université Libre de Bruxelles, IRIDIA
Avenue Franklin Roosevelt 50, CP 194/6, 1050 Brussels, Belgium
mdorigo@ulb.ac.be

and by

Luca Maria Gambardella
Scientific Director of
IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Via Cantonale, Galleria 2, CH-6928 Switzerland
luca@idsia.ch

May, 2003

A thesis submitted in partial fulfillment of the requirements of the *Université Libre de Bruxelles, Faculté de Sciences Appliquées* for the
DIPLOME D'ETUDES APPROFONDIES (DEA)

Preface

This thesis has been submitted to the Université Libre de Bruxelles as required for the Diplome d'Études Approfondies (DEA), which, for the author, represents an intermediate step for obtaining the PhD in computer science. Therefore, the work described here reports an ongoing research.

This work has been mainly done at IDSIA [22], Switzerland, and partly at IRIDIA [23], Belgium. The research has been supported by IDSIA, by the Swiss National Science Foundation project titled "On-line fleet management", grant 16R10FM, and by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this thesis 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.

Abstract

The Probabilistic Traveling Salesman Problem (PTSP) is a TSP problem in which each customer requires a visit only with a given probability, independently of the others. The goal is to find an a priori tour of minimal expected length over all customers, with the strategy of visiting a random subset of customers in the same order as they appear in the a priori tour. The PTSP is NP-hard, and only very small instances may be solved by exact methods. Here, we concentrate on the design of algorithms (heuristics) for finding good suboptimal solutions to the problem.

Given the analogies between the TSP and the PTSP, it is reasonable to expect that, like in the TSP, a good heuristic for the problem is composed by two components. The first component is a solution construction algorithm, which finds an initial solution. The second component is a local search algorithm, which tries to improve as much as possible the starting solution. A good heuristic algorithm usually integrates these two components, and repeats the sequence construction-improvement of a solution several times, untill a good solution or some other termination criterion is not satisfied.

In this thesis we propose new (for the PTSP) computational approaches for both solution construction and local search algorithms. As far as the solution construction algorithm is concerned, we investigate the potentialities of the ant colony optimization metaheuristic for the PTSP, which is a new approach for this type problem. This choice is motivated by the observation that, for the TSP, when adding to ant colony optimization algorithms local search procedures, the quality of the results obtained was close to that obtainable by other state-of-the-art methods. Also for the PTSP, it is reasonable to expect good performance of ant colony optimization algorithms with local search. As far as local search is concerned, we show that the current algorithms in the literature, 2-p-opt and 1-shift, are inadequate. This results pose the question whether a new fast local search operator for the PTSP can be derived. One possibility to design fast local search is to use an approximation of the objective function, which is faster to be computed than the exact one. Two types of approximations are proposed and analyzed in this thesis.

Part of the research developed in this thesis has been published, or is about to be published, in the following papers:

[6] L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*. Springer,

Berlin, Germany, 2002.

[7] L. Bianchi, L. M. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 176–187. Springer, Berlin, Germany, 2002.

[8] L. Bianchi and J. Knowles. Local search for the probabilistic traveling salesman problem: a proof of the incorrectness of bertsimas' proposed 2-p-opt and 1-shift algorithms. Technical Report IDSIA-21-02, IDSIA, Lugano, Switzerland, 2002. Submitted to European Journal of Operational Research.

Contents

Chapter 1

Introduction

1.1 Vehicle routing problems involving uncertainty

In typical distribution systems, vehicles provide delivery, customer pick-up, or repair and maintenance services to customers that are geographically dispersed in a given area. In most applications, a common objective is to find a set of routes for the vehicles which satisfies a variety of constraints and so as to minimize the total fleet operating cost. The problem of minimizing total cost has traditionally been called the vehicle routing problem (VRP).

The deterministic VRP is well known in the operations research literature (see [9, 11, 19, 21, 5, 31] for reviews). By 'deterministic', we mean routing problems in which the number of customers, their locations, the size of their demands, and all other elements defining the problems are known with certainty before the routes, or solutions, are designed. One can identify, however, a practically endless variety of problems in which one or more of these parameters are random variables, that is, subject to uncertainty in accordance to some probability distribution. VRPs involving uncertainty, are more appropriate models for the many real-world problems in which randomness is not only present, but a major concern. This type of VRPs belong to the wider class of probabilistic combinatorial optimization problems (PCOPSs, according to the terminology used in [4]).

In VRPs involving uncertainty one finds that, after solving a given instance realization involving certain values of the random variables, it becomes necessary to repeatedly solve many other instance realizations of the same problem, corresponding to different values of the random variables. These other instance realizations are therefore variations of the instance solved originally. Yet, they may be sufficiently different from one another to necessitate a reconsideration of the entire problem on part of the analyst. The most obvious approach to deal with this type of problems is the *re-optimization* strategy. It consists in attempting to optimally solve (or near-optimally with a good heuristic) every potential instance of the original problem. This approach, however, has several disadvantages. For example, if the combinatorial optimization is NP-hard (like in the case of VRPs), one might have to solve exponentially many instances of a hard problem. Moreover, in many applications it is necessary to find a solution to each instance realization quickly, but one might not have the required computing or other resources for doing so.

Rather than reoptimizing every potential instance realization, a completely different strategy consists in finding an *a priori* solution to the original problem, to be updated in a simple and fast way after each instance realization is known.

All the intermediate strategies between a priori and re-optimization are called *mixed* optimization strategies. These strategies allow for arbitrarily sophisticated and possibly computationally demanding modifications of one or more a priori solutions.

The above discussion of optimization strategies is general, in the sense that it applies to any PCOP. In the following, we focus on the particular problem which has been the subject of our research.

1.2 Definition of the Probabilistic Traveling Salesman Problem

Consider a completely connected graph $G = (V, E)$ where $V = \{i = 1, 2, ..., n\}$ is the set of vertices, representing a set of customers, and $E = \{e_{ij}\}$ is the set of edges joining each couple of customers i, j. To each edge e_{ij} is associated a cost d_{ij}, which represents the travel distance between customers i and j. A tour is a sequence of customers which starts and ends at one customer, and visits all the other customers exactly once. The traveling salesman problem (TSP) consists in finding a tour of minimal length.

Consider now a problem related to the TSP, but where customers are stochastic. (TSP with stochastic customers, or TSPSC): suppose that each customer i requires a visit only with probability p_i, independently of the others. If S denotes a set of customers that require a visit, then every random subset $S \subseteq V$ has a certain probability to happen. One can see the TSP with stochastic customers as a set of TSP problems, one for each random set S, each one having a certain probability of being the actual set of customers to be visited. The goal is to find, for each random

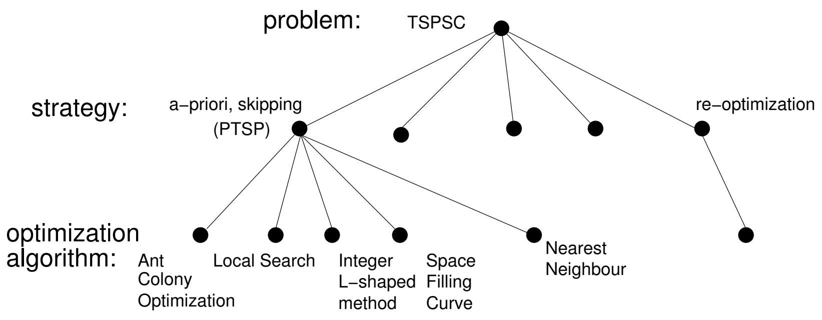


Figure 1.1: Logical tree representing the relation among a probabilistic combinatorial optimization problem (root), solution strategies (first level) and optimization algorithms (second level). The traveling salesman problem with stochastic customers (TSPSC) may be attacked with three types of strategies, a priori, mixed and re-optimization strategy. The TSPSC, if attacked with the a priori strategy is called probabilistic traveling salesman problem (PTSP), which is the subject of this research. Optimization algorithms should be designed according to the chosen solution strategy.

subset $S \subseteq V$, a traveling salesman tour over the customers in S in such a way as to minimize the expected tour length over all random subsets $S \subseteq V$.

The TSP with stochastic customers models a delivery context where a set of customers has to be visited on a regular (e.g., daily) basis, but all customers do not always require a visit. In general, every day only a subset of customers requires a visit, and the composition of this subset is only known the same day as the required deliveries. Therefore, every day there is a different traveling salesman problem to solve, over the subset of customers requiring a visit on that day. The objective, in this context, is to minimize the average distance traveled over many days of service.

As we have seen in section 1.1, given a PCOP, different strategies may be used to solve the problem. The definition and choice of a strategy is an important step in the solution process of the problem, since it influences the design of specific optimization algorithms. Figure 1.1 schematically shows different alternative possibilities to address the TSPSC. In this research, we focus on the a priori solution strategy for the TSPSC. The problem, when solved by an a priori strategy, is known as probabilistic traveling salesman problem (PTSP, see Figure 1.1). The a priori strategy consists in the definition of an a priori solution and of an updating method, which, for the PTSP, are the following. The a priori solution is defined as a tour visiting the complete set V of n customers. A very simple updating method modifies the a priori tour in order to have a particular tour for each subset of customers $S \subseteq V$:

for every subset of customers, visit them *in the same order* as they appear in the a priori tour, skipping the customers that do not belong to the subset. The strategy related to this method is also called the 'skipping strategy'. The objective of the PTSP is finding an a priori tour of minimum expected length under the skipping strategy. The concept of a priori solution, and the updating of the a priori solution when only a subset of customers require a visit, is exemplified in Figure 1.2, for a PTSP instance with 10 customers.

The PTSP approach models a delivery context where re-optimizing vehicle routes from scratch every day is unfeasible. In this context the delivery man would follow a standard route (i.e., an a priori tour), leaving out customers that on that day do not require a visit. The standard route of least expected length corresponds to the optimal PTSP solution.

1.2.1 Notation and objective function

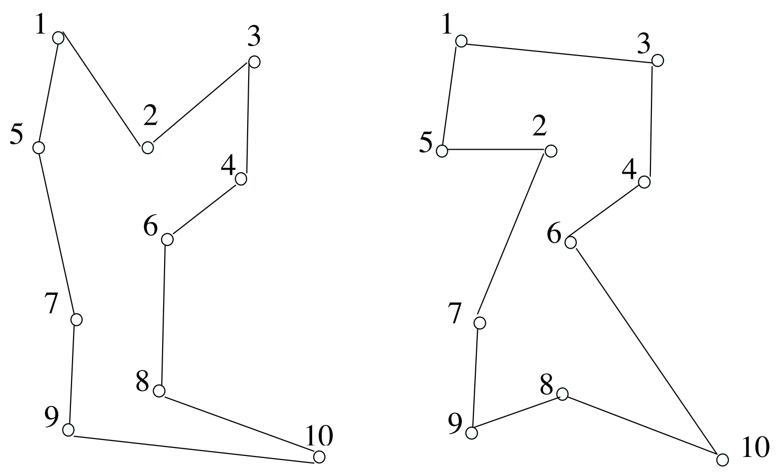
Let us consider an instance of the PTSP. We have a completely connected graph whose nodes form a set $V = \{i = 1, 2, ..., n\}$ of customers. Each customer has a probability p_i of requiring a visit, independently of the others. A solution for this instance is an a priori tour λ over all nodes in V,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

$$E[L(\lambda)] = \sum_{S \subseteq V} p(S) L(\lambda_{|S}) \ . \tag{1.1}$$

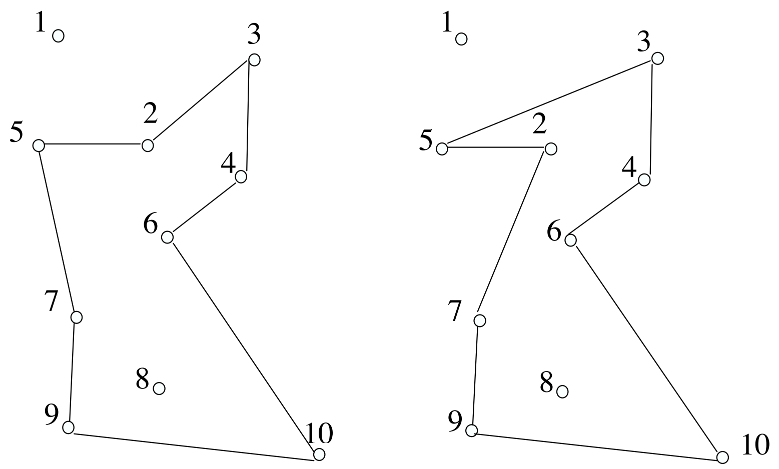
In the above expression, S is a subset of the set of nodes V, $L(\lambda_{|S})$ is the distance required to visit the subset of customers S (in the same order as they appear in the a priori tour), and $p(S)$ is the probability for the subset of customers S to require a visit:

$$p(S) = \prod_{i \in S} p_i \prod_{i \in V-S} (1 - p_i) \ . \tag{1.2}$$

The evaluation of expression (1.1) for the objective function requires a sum over a number of terms which is exponential in n. Jaillet [24, 25] showed that the evaluation of the PTSP objective function (eq.(1.1)) can be done in $O(n^2)$. In fact, let us consider (without loss of generality) an a priori tour $\lambda = (1, 2, \ldots, n)$; then its



(a) Example of two a priori tours



(b) The two tours resulting from the a priori tours of Fig.1.2(a) when customers 1 and 8 do not require a visit

Figure 1.2: Graphical examples of a PTSP instance and two a piori solutions.

expected length is

$$E[L(\lambda)] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij} p_i p_j \prod_{k=i+1}^{j-1} (1-p_k) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} d_{ij} p_i p_j \prod_{k=i+1}^{n} (1-p_k) \prod_{l=1}^{j-1} (1-p_l) \ . \quad (1.3)$$

This expression is derived by looking at the probability for each arc of the complete graph to be used, that is, when the a priori tour is adapted by skipping a set of customers which do not require a visit (see, for instance, Figure 1.2). For instance, an arc (i, j) is actually used only when customers i and j do require a visit , while customers $i+1, i+2, ..., j$ do not require a visit. This event occurs with probability $p_i p_j \prod_{k=i+1}^{j-1}(1-p_k)$ (when $j \leq n$).

1.3 Brief literature review

The PTPS was introduced in Jaillet's PhD thesis [24]. This problem is an NP-hard [4, 1], and instances only up to 50 customers have been solved to optimality [27]. This is why a lot of effort has been done in the literature in designing heuristic algorithms for finding good suboptimal solutions for the PTSP. Heuristics using a nearest neighbor criterion or savings criterion were implemented and tested by Jézéquel [26] and by Rossi-Gavioli [30]. Later, Bertsimas-Jaillet-Odoni [4] and Bertsimas-Howell [3] have further investigated some of the properties of the PTSP and have proposed some heuristics for the PTSP. These include tour construction heuristics (space filling curves and radial sort), and tour improvement heuristics (probabilistic 2-opt edge interchange local search and probabilistic 1-shift local search). Most of the heuristics proposed are an adaptation of a TSP heuristic to the PTSP, or even the TSP heuristic itself, which in some cases gives good PTSP solutions. Recently, Branke-Guntsch [10] have applied ant colony optimization algorithms to the PTSP, extending the results of this research thesis and using an approximated objective function to speed up the computation.

1.4 Outline of the thesis

Given the analogies between the TSP and the PTSP, it is reasonable to expect that, like in the TSP, a good heuristic for the problem is composed by two components.

The first component is a solution construction algorithm, which finds an initial solution. The second component is a local search algorithm, which tries to improve as much as possible the starting solution. A good heuristic algorithm usually integrates these two components, and repeats the sequence construction-improvement of a solution several times, until a good solution or some other termination criterion is not satisfied. In this thesis we propose new (for the PTSP) computational approaches for both solution construction and local search algorithms.

The thesis is organized as follows. In section 2 we address the study of a solution construction algorithm for the PTSP, in particular, we investigate the potentialities of ant colony optimization (ACO) algorithms for the PTSP. This choice is motivated by the observation that, for the TSP, when adding to ACO algorithms local search procedures based on 3-opt [28], the quality of the results obtained [20, 17] was close to that obtainable by other state-of-the-art methods. Given the analogies between the TSP and the PTSP, it is reasonable to expect good performance of ACO algorithms with local search also on the PTSP. In section 3 we address the issue of local search for the PTSP, by showing that the current algorithms in the literature, 2-p-opt and 1-shift [4, 3], are inadequate. This results pose the question whether a new fast local search operator for the PTSP can be derived. One possibility to design fast local search is to use an approximation of the objective function, which is faster to be computed than the exact one. Two types of approximations are proposed and analyzed in section 4. The three appendices of this thesis contain additional details about the local search algorithms we addressed in section 3.

Chapter 2

Solution construction algorithms for the PTSP

This chapter explores the possibility of using the Ant Colony Optimization meta-heuristic (ACO) for solving the PTSP. For the TSP, when adding to ACO algorithms local search procedures based on 3-opt [28], the quality of the results obtained [20, 17] was close to that obtainable by other state-of-the-art methods. Given the analogies between the TSP and the PTSP, it is reasonable to expect good performance of ACO algorithms with local search also on the PTSP.

Section 2.1 introduces the ACO meta-heuristic as solution framework for optimization problems in general. The second section (section 2.2) describes the particular implementations of ACO we chose for solving the PTSP, that is, ACS and pACS. The last section reports the experimental analysis of ACS and pACS, and comparisons with other solution construction algorithms in the PTPS literature.

2.1 The ACO metaheuristic

ACO is a metaheuristic approach proposed by Dorigo [13] and later by Dorigo et al.[18]. A complete description and review of ACO applications may be found in [15, 16, 14]. Here, we follow the description of ACO given in [14].

The inspiring source of ACO is the foraging behavior of real ants. This behavior, described by Deneubourg et al in [12], enables them to find shortest paths between food sources and their nest. While walking from food sources to the nest and vice versa, ants deposit a substance called *pheromone* on the ground. When they decide about a direction to go they choose, with higher probability, paths marked by strong pheromone concentrations. This basic behavior is the basis for a cooperative

```
procedure ACO metaheuristic for combinatorial optimization problems
    while (termination condition not met) do
        schedule_activities
            ants_activity()
            pheromone_evaporation()
            daemon_actions()
        end schedule_activities
    end while
end ACO metaheuristic
```

Figure 2.1: High level pseudocode for the ACO metaheuristic.

interaction which leads to the emergence of shortest paths. In fact, if a path is shorter than the others, an ant that luckily finds it will walk through satisfying it in a shorter time than the time required by the other ants on longer paths. This implies that the shorter path may be marked by pheromone earlier than other paths, and it is therefore preferred, in probability, by other ants.

In order for a problem to be solvable by ACO, it must be modeled in the following way. Given a graph $G = (C, L, W)$, a feasible solution is a path on this graph, satisfying a set Ω of constraints. The optimal solution is a minimum cost path (for minimization problems) over G. The graph $G = (C, L, W)$ and the constraints Ω are defined as follows: $C = \{c_1, c_2, \ldots c_n\}$ is a finite set of possible *components*, $L = \{l_{c_i c_j} | c_i, c_j \in C\}$ a finite set of possible *connections* among the elements of C, W a set of weights associated either to the components C or to the connections L or both, and $\Omega(C, L, \theta)$ is a finite set of *constraints* assigned over the elements of C and L (θ indicates that the constraints can change over time). For example, in the TSP C is the set of customers, L is the set of edges connecting the customers, W the length of the edges in L, and the constraints Ω impose that in any feasible solution each customer appears once and only once. A feasible path over G is solution ψ and a minimum cost path is an optimal solution and it is indicated by ψ^*; $f(\psi)$ is the cost of solution ψ, and $f(\psi^*)$ is the cost of the optimal solution. In the TSP for example, a solution ψ is an Hamiltonian circuit and ψ^* the shortest feasible Hamiltonian circuit.

In ACO algorithms a colony of artificial ants iteratively and stochastically constructs solutions for the problem defined by G and Ω using artificial pheromone trails and heuristic information. Most ACO algorithms for combinatorial optimization problems follow the algorithmic scheme given in Fig. 2.1.

During the procedure *ants_activities()* each ant k starts with a partial solution

$\psi_k(1)$ consisting of one of the components in C, and adds components to $\psi_k(h)$ till a complete feasible solution ψ is built, where h is the step counter. Components to be added to $\psi_k(h)$ are stochastically chosen in an appropriately defined neighborhood of the last component added to $\psi_k(h)$. The ants stochastic choice is made by applying a stochastic local decision policy that makes use of local information available at the visited components. Once an ant has built a solution, or while the solution is being built, the ant evaluates the (partial) solution and adds pheromone on the components and/or connections it used. Pheromone is added in such a way to give information about the quality of the (partial) solution. This pheromone information will direct the search of the ants in the following iterations.

Besides ants' activity, an ACO algorithm includes a *pheromone_evaporation()* and an optional *daemon_action()* procedure. Pheromone evaporation is the process by which the pheromone trail automatically decreases over time. Evaporation allows the exploration of new solutions, and should avoid early convergence of the ACO algorithm to a poor solution. "Deamon" actions can be used to implement centralized actions which cannot be performed by single ants. Examples are the activation of a local optimization procedure, or the collection of global information that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a non-local perspective. For instance, the deamon can observe the path found by each ant in the colony and choose to deposit extra pheromone on the edges used by the ant that made the shortest path. In most applications to combinatorial optimization problems the *ant_activity(), pheromone_evaporation()* and *deamon_actions()* procedures are scheduled sequentially. Nevertheless, the **schedule_activity** construct of the ACO metaheuristic (Fig.2.1) leaves the decision on how these three procedures must be synchronized to the user.

2.2 Solving the PTSP by the ACO metaheuristics

We apply to the PTSP the Ant Colony System (ACS) [20, 17], a particular ACO algorithm which was successfully applied to the TSP. We also consider a modification of ACS which explicitly takes into account the PTSP objective function (we call this algorithm probabilistic ACS, that is, pACS). Another ACO approach to the PTSP has been recently proposed by Branke et al. [10], where an approximate objective function and the use of a new heuristic guidance scheme seem to give promising results. In the following, we describe how ACS and pACS construct a solution (procedure *ant_activity()* of Fig.2.1) and how they update pheromone trails (procedure *pheromone_evaporation()* and *deamon_action()* of Fig.2.1).

2.2.1 Solution construction in ACS and pACS

A feasible solution for an n-city PTSP is an a priori tour which visits all customers. Initially m ants are positioned on their starting cities chosen according to some initialization rule (e.g., randomly). Then, the solution construction phase starts (procedure *ant_activity()* in Fig.2.1). Each ant progressively builds a tour by choosing the next customer to move to on the basis of two types of information, the pheromone τ and the heuristic information η. To each arc joining two customers i, j it is associated a varying quantity of pheromone τ_{ij}, and the heuristic value $\eta_{ij} = 1/d_{ij}$, which is the inverse of the distance between i and j. When an ant k is on city i, the next city is chosen as follows.

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

- With probability $1 - q_0$, a city j is chosen randomly with a probability given by

$$p_k(i,j) = \begin{cases} \frac{\tau_{ij} \cdot \eta_{ij}^{\beta}}{\sum_{r \in J_k(i)} \tau_{ir} \cdot \eta_{ir}^{\beta}}, & \text{if } j \in J_k(i) \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

This decision rule has a double function: with probability q_0 the decision rule exploits the knowledge available about the problem, that is, the heuristic knowledge about distances between cities and the learned knowledge memorized in the form of pheromone trails, while with probability $1 - q_0$ it operates a biased exploration. Tuning q_0 allows to modulate the degree of exploration and to choose whether to concentrate the activity of the system on the best solutions so far or to explore the search space.

2.2.2 Pheromone trails update in ACS and pACS

Pheromone trails are updated in two stages, the first is part of the *ant_activity()*, while the second is part of the *deamon_actions()* (Fig.2.1). In the first stage of pheromone update, each ant, after it has chosen the next city to move to, applies the following local update rule:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0, \tag{2.2}$$

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

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

$$\tau_{ij} \leftarrow (1-\alpha) \cdot \tau_{ij} + \alpha \cdot \Delta\tau_{ij}, \tag{2.3}$$

where

$$\Delta\tau_{ij} = ObjectiveFunc_{best}^{-1} \tag{2.4}$$

with $0 < \alpha \leq 1$ being the pheromone decay parameter, and $ObjectiveFunc_{best}$ is the value of the objective function of the best-so-far tour. This pheromone updating rule is a “deamon” action, since ants do not have the knowledge about the best-so-far tour. ACS and pACS only differ by the use of this “deamon” action. In fact, in ACS the objective function is the a priori tour length, while in pACS the objective function is the PTSP expected length of the a priori tour. In the next section we discuss in more detail the differences between ACS and pACS.

2.2.3 Discussion of differences between ACS and pACS

Differences between ACS and pACS are due to the fact that the two algorithms exploit different objective functions in the pheromone updating phase. The global updating rule of equations (2.3) and (2.4) implies two differences in the way ACS and pACS explore the search space. The first and most important difference is the set of arcs on which pheromone is globally increased, which is in general different in ACS and pACS. In fact, the ‘best tour’ in eq. (2.4) is relative to the objective function. Therefore in ACS the search will be biased toward the shortest tour, while in pACS it will be biased toward the tour of minimum expected length. The second difference between ACS and pACS is in the quantity $\Delta\tau_{ij}$ by which pheromone is increased on the selected arcs. This aspect is less important than the first, because ACO in general is more sensitive to the difference of pheromone among arcs than to its absolute value.

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

$$\Delta = L_\lambda - E[L_\lambda] = (1-p^2)L_\lambda - p^2 \sum_{r=1}^{n-2}(1-p)^r L_\lambda^{(r)}, \tag{2.5}$$

which implies

$$\Delta \sim O(q \cdot L_\lambda) \tag{2.6}$$

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

In the heterogeneous PTSP, it is not easy to see the relation between the two objective functions, since each arc of the a priori tour L_λ is multiplied by a different probabilistic weight (see eq.(1.3)), and a term with L_λ cannot be isolated in the expression of $E[L_\lambda]$, as in the homogeneous case.

ACS and pACS also differ in time complexity. In both algorithms one iteration (i.e., one cycle through the *while* condition of Fig. 2.1) is $O(n^2)$ [18], but the constant of proportionality is bigger in pACS than in ACS. To see this one should consider the procedure *UpdateTrail* of Fig. 2.1, where the best-so-far tour must be evaluated in order to choose the arcs on which pheromone is to be updated. The evaluation of the best-so-far tour requires $O(n)$ time in ACS and $O(n^2)$ time in pACS. ACS is thus faster and always performs more iterations than pACS for a fixed CPU time.

2.3 Experimental tests

2.3.1 Homogeneous PTSP Instances

Homogeneous PTSP instances were generated starting from TSP instances and assigning to each customer a probability p of requiring a visit. TSP instances were taken from two benchmarks. The first is the TSPLIB [32], from which we considered instances with a number of customers between 50 and 200. The second benchmark is a group of instances where customers are randomly distributed on the square $[0, 10^6]$. Both uniform and clustered distributions where considered in this case, and the number of cities varied between 50 and 350. For generating random instances we used the Instance Generator Code of the 8^{th} DIMACS Implementation Challenge at http://research.att.com/dsj/chtsp/download.html.

2.3.2 Computational environment and ACS parameters

Experiments were run on a Pentium Xeon, 1GB of RAM, 1.7 GHz processor. In order to asses the relative performance of ACS versus pACS independently from the details of the settings, the two algorithms were run with the same parameters. We chose the same settings which yielded good performance in earlier studies with ACS on the TSP [17]: $m = 10$, $\beta = 2$, $q_0 = 0.98$, $\alpha = \rho = 0.1$ and $\tau_0 = 1/(n \cdot Obj)$, where n is the number of customers and Obj is the value of the objective function evaluated with the nearest neighbor heuristic [17].

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

2.3.3 Comparison between pACS and ACS

For each TSP instance tested, nine experiments were done varying the value of the customer probability p from 0.1 to 0.9 with a 0.1 interval. Fig. 2.2 summarizes results obtained on 21 symmetric PTSP instances, one third of the instances were taken from the TSPLIB, the others were random instances (half of them uniform and half clustered). The figure shows the relative performance of pACS versus ACS, averaged over the tested instances. A typical result for a single instance is reported in Fig. 2.3.

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

The reason why pACS does not always perform better than ACS is clear if we consider two aspects: the complexity (speed) of ACS versus pACS, and the goodness of the PTSP objective function approximation as p approaches 1. When p is near to 1, a good solution to the TSP is also a good solution to the PTSP; therefore, ACS, which performs more iterations than pACS, has a better chance to find a good solution.

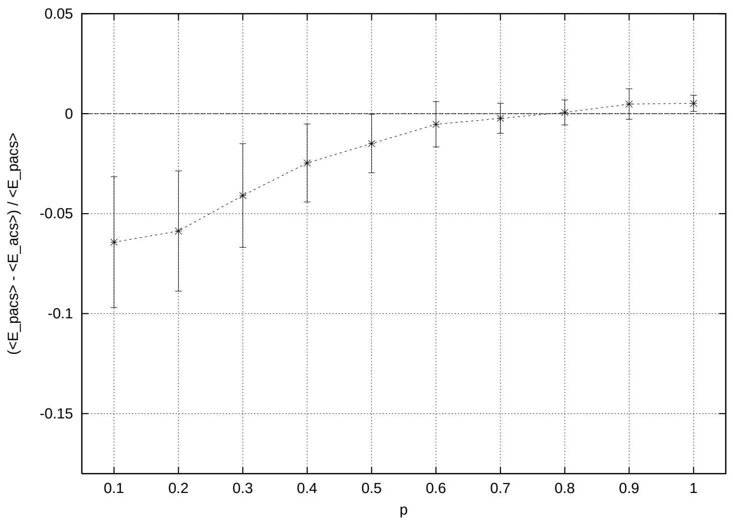


Figure 2.2: Relative performance of pACS versus ACS for the homogeneous PTSP. The vertical axis represents $(E[L_\lambda(pACS)] - E[L_\lambda(ACS)])/E[L_\lambda(pACS)]$. On the horizontal axis there is the customer probability p. Each point of the graph is an average over 21 symmetric homogeneous PTSP instances. Error bars represent average deviation, defined as $\sum_{i=1}^{n} |x_i - <x> |/n$, with $n = 21$. Note that for $p = 1$ ACS outperforms pACS, since for a fixed CPU stopping time ACS makes more iterations.

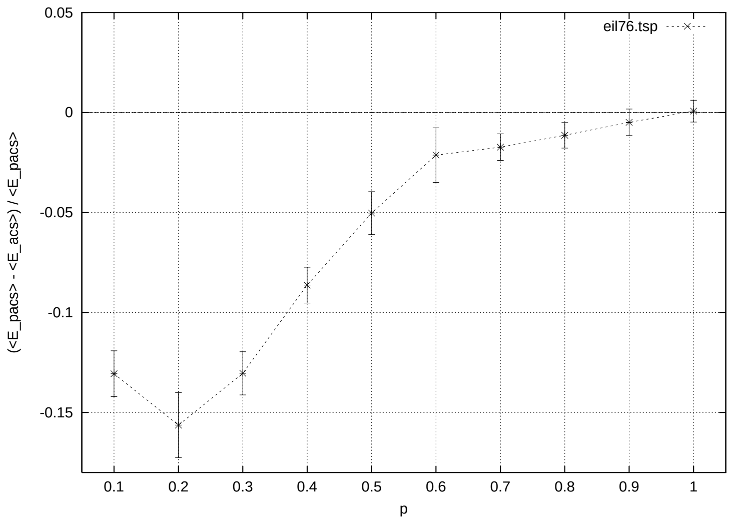


Figure 2.3: Relative performance of pACS versus ACS for the eil76.tsp instance of the TSPLIB. Error bars represent the average deviation of the result over 5 independent runs of the algorithms.

2.3.4 Comparison between pACS and other tour construction heuristics

We compared pACS with two simple tour construction heuristics, the radial sort and the random best heuristic. The random best heuristic generates random tours and selects the one with the shortest expected length. Random best and pACS were run on the same machine (a Pentium Xeon, 1GB of RAM, 1.7 GHz processor) for the same CPU time ($stoptime = k \cdot n^2$ CPU seconds, with $k = 0.01$). For pACS we chose the same settings which yielded good performance in earlier studies with ACS on the TSP [17]: $m = 10$, $\beta = 2$, $q_0 = 0.98$, $\alpha = \rho = 0.1$ and $\tau_0 = 1/(n \cdot Obj)$, where n is the number of customers and Obj is the value of the objective function evaluated with the nearest neighbor heuristic [17]. For each experiment, we run 5 independent trials of pACS.

Radial sort builds a tour by sorting customers by angle with respect to the 'center of mass' of the customer spatial distribution. The 'center of mass' coordinates have been computed here by averaging over the customers coordinates. The a priori tour which radial sort builds does not depend on the customer probabilities, and a unique tour is thus used as a priori tour for all probabilities of the PTSP. Even if very simple, this heuristic is interesting for the PTSP, because of the conjecture [3] that the tour generated by radial sort is near optimal for small customer probabilities. Moreover, the combination of radial sort and the 1-shift local search have shown to be the best combination of tour construction and tour improvement heuristics in [3]. A disadvantage of radial sort is that it is only applicable to those PTSP instances where the coordinates of customers are known. In general, this is not the case for asymmetric PTSP instances, where the arc weights may have, for instance, the meaning of travel times.

The average relative performance of pACS with respect to radial sort and random best heuristics is shown in Fig. 2.4. The first observation is that pACS always performs better than radial sort and random best, for each probability and for each type of instance, while random best is always very poor both with respect to pACS and to radial sort. Secondly, radial sort and pACS are equivalent for small probabilities (prob = 0.1). This results supports the conjecture of near-optimality of radial sorted tours for small probability, and it is interesting that this also applies to non uniform instances, such as TSPLIB and random clustered instances.

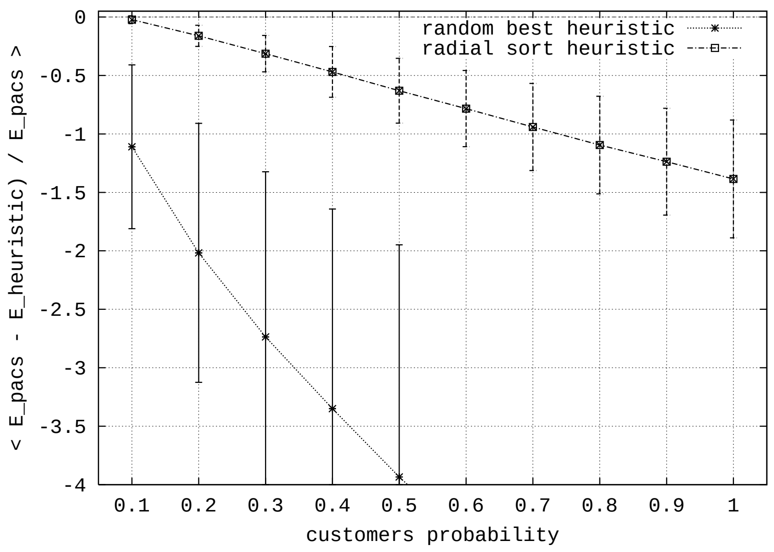


Figure 2.4: Relative performance of pACS with respect to radial sort and random best heuristics. On the horizontal axis there is the customers probability. Each point is an average over 21 symmetric PTSP instances. Error bars represent average deviation, defined as $\sum_{i=1}^{n} |x_i - <x> |/n$, with $n = 21$.

2.3.5 Absolute performance

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

$$E[L_{\lambda^*}] \geq LB \; . \tag{2.7}$$

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

$$\frac{E[L_\lambda] - E[L_{\lambda^*}]}{E[L_{\lambda^*}]} \leq \frac{E[L_\lambda] - LB}{LB} \; . \tag{2.8}$$

In the following we apply two different techniques for evaluating a lower bound to the optimal PTSP solution (and thus for evaluating the absolute performance of pACS). In the first case a theoretical lower bound is used while in the second case the lower bound is estimated by using Monte Carlo sampling.

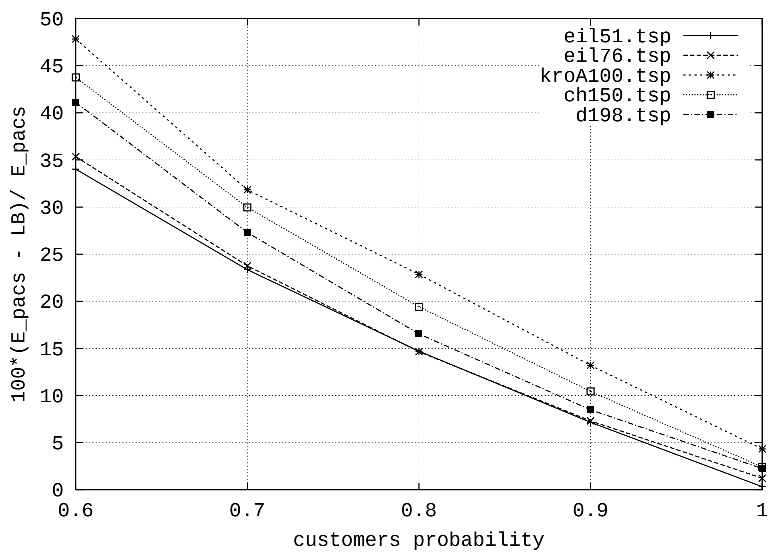


Figure 2.5: Upper bound of relative percent error of pACS for 5 TSPLIB instances. Note that, for decreasing customers probability, the upper bound to the relative error becomes bigger at least partially because the lower bound to the optimum becomes less tight.

Theoretical lower bound to the PTSP optimum.

For the homogeneous PTSP and for instances where the optimal length L_{TSP} of the corresponding TSP is known, it is possible to use the following lower bound to the optimal expected length, as was proved in [3]

$$LB = pL_{TSP}(1 - (1 - p)^{n-1}) \ . \tag{2.9}$$

If we put this lower bound into the right side of equation (2.8), we obtain an upper bound of the relative error of pACS. Fig. 2.5 shows the absolute performance of pACS, evaluated with this method, for a few TSPLIB instances. From the figure we see that, for example, pACS finds a solution within 15% of the optimum for a homogeneous PTSP with customers probability 0.9. This technique for evaluating the absolute performance of a PTSP heuristic is rigorous, but has the limitation that the lower bound for small probabilities is not tight, so that it produces big overestimates of the error. The following technique is more flexible and gives better estimates of the error, even if, as we will see, it also has some limitations.

Estimated a posteriori optimum.

The expected tour length under re optimization is defined as the average of the lengths of the optimal TSP solution to each subset of customers, and it is also called a posteriori optimum, since it is the value obtained by solving a TSP problem once the set of customers requiring a visit on a certain day is known. The a posteriori optimum is a lower bound on the optimal PTSP solution, because the length induced by the PTSP a priori tour on a subset of customers cannot be smaller than the optimal TSP solutions for that subset of customers.

The exact evaluation of the a posteriori optimum is impractical, because it requires the solution of 2^n instances of the TSP to optimality. The technique proposed in [3], consists in making two approximations. First, only a random sample of the 2^n subsets of customers is selected, by means of a stratified Monte Carlo sampling (see [3] for a detailed description). Second, each random sample of customers S is solved to near optimality as a TSP by choosing the best of $|S|/\gamma$ random tours (γ is a parameter) and applying to it the 3-opt local search.

This technique can be applied easily only to small instances (say, up to 100 customers). Otherwise, care must be taken in order to avoid overflow when generating the stratified samples from a set of 2^n subsets of customers. We report average results for 10 random uniform and clustered instances in Fig. 2.6. In our tests we used $\gamma = 0.5$ and about 400 samples. From the figure we see that pACS is within 8% of the optimum when applied to uniform random instances, while it is within 14% of the optimum if the instances are clustered.

2.3.6 Conclusions

In this chapter we investigated the potentialities of pACS, a particular ACO algorithm, for the homogeneous PTSP. We showed that the pACS algorithm is a promising tour construction heuristic for the PTSP. We compared pACS with other tour construction heuristics and we provided an estimation of the absolute error with respect to the optimal PTSP solution for some instances. We also compared pACS to ACS, and we showed that for customers probability close to 1, the ACS heuristic is a better alternative than pACS.

In this chapter the ACO metaheuristic was applied without a local search for improving the a priori tour. The study of an efficient local search for the PTSP, which should greatly improve the solution quality of pACS and of any tour construction heuristic in general, is an important issue, and it is analyzed in chapter 3.

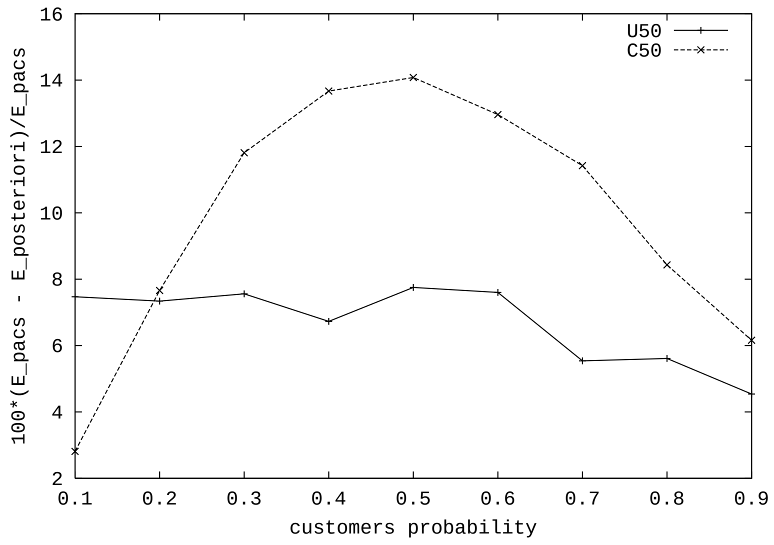


Figure 2.6: Relative percent error with respect to the estimated a posteriori optimum for random uniform instances (U50) and for random clustered instances (C50) of 50 customers. Each point is an average over 10 random instances.

Chapter 3

The issue of local search for the PTSP

In order for a metaheuristic to achieve state-of-the art results in an optimization problem, the local search or solution improvement part of the algorithm is very important.

In the literature there has been one major attempt to apply TSP-inspired local search algorithms to the homogeneous PTSP: the 2-p-opt and 1-shift algorithms introduced by Bertsimas [1, 3], who proposed for these two algorithms fast evaluation operators, which compute the cost evaluations of the entire neighborhood of a tour in $O(n^2)$ time. Unfortunately, derivations for these expressions were not given in [3], and it has since come to our attention that they do not correctly calculate the change in expected tour length. This fact has two consequences. First, results published in several papers, which used the 2-p-opt and 1-shift local search, should be re-examined because they may be wrong. Second, it poses the question whether a new fast local search operator for the PTSP can be derived. In fact, using the $O(n^2)$ full evaluation of the objective function for the cost of a move gives an $O(n^4)$ local search algorithm (since the neighborhood size of 1-shift and 2-p-opt is $O(n^2)$), so any improvement on this would be most useful.

This chapter is organized as follows. In section 3.1 we describe some basic concepts about local search algorithms in general. Section 3.2 highlights the difficulties to be dealt with when trying to apply TSP-inspired local search algorithms to the PTSP. In section 3.3 we describe the 2-p-opt move and give the delta evaluation expressions proposed in [3], followed by a proof of their incorrectness. Section 3.4 follows a similar course but with respect to the 1-shift operator. While our focus here is on the expressions published in [3], we note that the expressions given in

Bertsimas' PhD thesis [1] are also incorrect and given without derivation. Since the 2-p-opt expression given in [1] differs slightly from that in [3], we include a separate appendix (appendix A) to address this expression. Two further appendices (appendix B and C) present a full calculation of the cost of local search moves for a simple PTSP tour, when the 2-p-opt [3] and 1-shift operators are applied, respectively. The calculations demonstrate, concretely, the different results obtained using Bertsimas' delta expressions, compared with the full evaluation. These calculations were generated by a computer program written in C which is also available at http://www.idsia.ch/∼leo/.

3.1 Local search algorithms

Local search is one of the most effective heuristics in many combinatorial optimization problems [29], and it is based on a very simple method: trial and error. Local search algorithms repeatedly perform simple modifications of a solution (moves), accepting a move only when the cost of the move is negative. More formally, given an instance (F, c) of an optimization problem, where F is the set of feasible solutions and c is the cost mapping, we choose a neighborhood

$$N : F \to 2^F \tag{3.1}$$

which is searched at point $t \in F$ for improvements by the procedure

$$\textit{improve(t)} = \begin{cases} \text{any } s \in N(t) \text{ with } c(s) < c(t), & \text{if such an } s \text{ exists} \\ \text{“no”}, & \text{otherwise.} \end{cases} \tag{3.2}$$

The general local search procedure is shown in Fig.3.1. We start at some initial feasible solution $t \in F$ and use the procedure *improve(t)* to search for a better solution in its neighborhood. So long as an improved solution exists, we adopt it and repeat the neighborhood search from the new solution; when we reach a local optimum we stop. A local optimum is a solution $t^* \in F$ such that $c(t^*) \leq c(t)$, for all neighbor solutions $t \in N(t^*)$.

To apply the local search approach to a problem, one should make a number of choices. First, one should obtain an initial feasible solution. Sometimes the initial solution is obtained by using some heuristic algorithm, or even a random choice could be useful. Next, one should choose a 'good' neighborhood for the problem at hand, and a method for searching it. This choice is usually guided by intuition. In general, there is a trade-off between large and small neighborhoods. A larger

```
procedure local search
    t:= some initial starting solution in $F$
    while (improve(t) $\neq$ ‘no’) do
        t := improve(t)
    end while
end local search
```

Figure 3.1: High level pseudo-code for the local search.

neighborhood might provide better local optima, but will take more time to search. Is it better to generate fewer ‘stronger’ local optima or more ‘weaker’ ones? There is no general rule to answer this question. Another aspect related to this trade-off is wether to search the all neighborhood for the best neighbor of the current solution (best improvement local search), or to be happy with the first improving solution (first improvement local search), like in the local search sketched in Fig.3.1.

More aspects related to the implementation of a good local search and applications to a diversity of problems may be found in [29]. In the following, we focus on the main difficulties to be overcome when designing a local search for the PTSP problem.

3.2 TSP-inspired local search algorithms

Because of the analogy between the TSP and the PTSP, it may seem reasonable to apply TSP local search algorithms to the PTSP. Nevertheless, a local search that is efficient for the TSP may be inefficient when applied to the PTSP. In this section, we will analyze some possible advantages and disadvantages of applying TSP-inspired local search algorithms to the PTSP.

The run time efficiency of a local search depends, among many factors, on three aspects:

- the time to *evaluate* a neighbor solution in order to check if it is an improving one,
- the time to *find* an improving solution (or to verify that none exists),
- the time to *perform* the change from the present to the new improving solution.

The last point only depends on the dimension of the neighborhood and the second point only depends on the data structure representing a solution, which are same for the TSP and for the PTSP. In fact, an a priori solution to the PTSP may also represent a solution to the corresponding TSP, that is, to the problem obtained only by considering the distance matrix of the PTSP and disregarding the customers probabilities (see the definition of the PTSP in section1). Therefore, if a TSP local search perform these two operations efficiently, the same is true for the PTSP.

The critical point is the first one, that is, the time to *evaluate* a neighbor solution. In fact, in successful TSP local searches this process typically requires $\Theta(1)$ time, while in the PTSP it is not trivial to find such an efficient evaluation, due to the fact that the PTSP objective function is much more complex than the TSP one. Let us reconsider the form of the objective function for the homogeneous PTSP.

Given an a priori tour $\lambda = \lambda(1), \lambda(2), ..., \lambda(n)$, its expected length (that is, the objective function), is computed in $O(n^2)$ by the following expression [24]:

$$E[L(\lambda)] = \sum_{j=1}^{n} \sum_{r=1}^{n-1} p^2(1-p)^{r-1} d(\lambda(j), \lambda(j+r)). \tag{3.3}$$

Throughout the thesis we use the convention that, $\forall i \in \{1, 2, ..., n\}$ and $\forall r \in \{1, 2, ..., n-1\}$,

$$\lambda(i \pm r) = \begin{cases} \lambda((i \pm r) \bmod n), & \text{if } i \pm r \neq 0 \text{ and if } i \pm r \neq n \\ \lambda(n), & \text{otherwise .} \end{cases} \tag{3.4}$$

We recall that the meaning of equation (3.3) is as follows. In each term of the sum the distance between the j^{th} city $\lambda(j)$ and the $(j+r)^{\text{th}}$ city $\lambda(j+r)$ is weighted by the probability that the two nodes require a visit (p^2) while the $r-1$ nodes between them do not require a visit ($(1-p)^{r-1}$). In other words, each arc $(\lambda(j), \lambda(j+r))$ is weighted by a probability coefficient $p^2(1-p)^{r-1}$.

The 2-p-opt and 1-shift algorithms introduced by Bertsimas [1, 3], use fast evaluation operators, which compute the cost evaluations of the entire neighborhood of a tour in $O(n^2)$ time. Unfortunately, as we will show in the next sections, they do not correctly calculate the change in expected tour length.

3.3 The 2-p-opt

The 2-p-opt local search was introduced by Bertsimas in [1] and later published by this journal in an article by Bertsimas and Howell [3]. In this chapter we refer to [3]. Given an a priori tour λ, its 2-p-opt neighborhood is the set of tours obtained by reversing a section of λ (that is, a set of consecutive nodes) and adjusting the arcs adjacent to the reversed section, as for example in Figure 3.2.

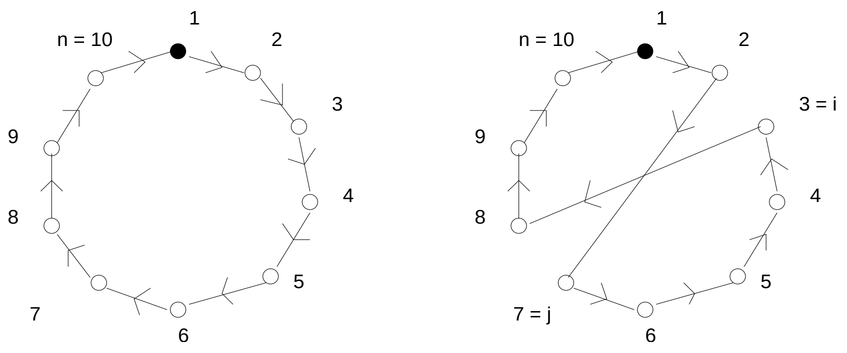


Figure 3.2: Tour $\lambda = 1, 2, ..., i, i+1, ..., j, ..., n$ (left) and tour $\tilde{\lambda} = 1, 2, ..., i-1, j, j-1, ..., i, j+1, ..., n$ (right) obtained from λ by reversing the section $(i, i+1, ..., j)$, with $n = 10$, $i = 3$, $j = 7$.

3.3.1 Expressions proposed in [3] for the cost of a 2-p-opt move

Consider, without loss of generality, a tour $\lambda = 1, 2, ..., i, i+1, ..., j, ..., n$ and a tour $\tilde{\lambda} = 1, 2, ..., j, j-1, ..., i, j+1, ..., n$ obtained by reversing a section $(i, i+1, ..., j)$ of λ. Let $\Delta E_{i,j}$ denote the change in the expected length $E[L(\tilde{\lambda})] - E[L(\lambda)]$. In the following we reproduce verbatim the expressions given in [3] for the computation of $\Delta E_{i,j}$. Let $j = i + k$. Given the two dimensional matrices of partial results A and B:

$$A_{i,k} = \sum_{r=k}^{n-1} (1-p)^{r-1} d(i, i+r) \quad \text{and} \quad B_{i,k} = \sum_{r=k}^{n-1} (1-p)^{r-1} d(i-r, i), \quad 1 \leq k \leq n-1, \quad 1 \leq i \leq n, \tag{3.5}$$

$\Delta E_{i,j}$ is computed by means of the following recursive equations. Letting $q = 1 - p$, for $k = 1$,

$$\Delta E_{i,i+1} = p^3[q^{-1}A_{i,2} - (B_{i,1} - B_{i,n-1}) - (A_{i+1,1} - A_{i+1,n-1}) + q^{-1}B_{i+1,2}], \tag{3.6}$$

and for $k \geq 2$,

$$\begin{aligned} \Delta E_{i,j} = \Delta E_{i+1,j} \quad & +p^2[(q^{-k} - 1)A_{i,k+1} + (q^k - 1)(B_{i,1} - B_{i,n-k}) \\ & +(q^k - 1)(A_{j,1} - A_{j,n-k}) + (q^{-k} - 1)B_{j,k+1} \\ & +(q^{n-k} - 1)(A_{i,1} - A_{i,k}) + (q^{k-n} - 1)B_{i,n-k+1} \\ & +(q^{k-n} - 1)A_{j,n-k+1} + (q^{n-k} - 1)(B_{j,1} - B_{j,k})]. \end{aligned} \tag{3.7}$$

The 2-p-opt local search proposed proceeds in two phases. The first phase consists of computing $\Delta E_{i,i+1}$ for every value of i, and at the same time accumulating the

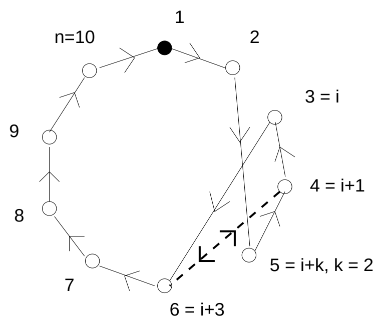


Figure 3.3: Tour $\tilde{\lambda}$, with $\tilde{\lambda}(i) = i+2$, $\tilde{\lambda}(i+1) = i+1$, $\tilde{\lambda}(i+2) = i$, and $\tilde{\lambda}(i+3) = i+3$. The dashed line represents arcs $(i+1, i+3)$ and $(i+3, i+1)$.

two matrices of partial results A and B. Note that since the computation of $\Delta E_{i,i+1}$ only involves two rows of A and B, one can proceed to the next pair of nodes without recomputing each entire matrix. Each time a negative $\Delta E_{i,i+1}$ is encountered, one immediately switches the two nodes involved. The n calculations of phase one require $O(n)$ time apiece, or $O(n^2)$ time in all. At the end of this phase, an a priori tour is reached for which every $\Delta E_{i,i+1}$ is positive, and the matrices A and B are complete and correct for that tour. The second phase of the local search consists of computing $\Delta E_{i,j}$ recursively by means of equation (3.7). Since each $\Delta E_{i,j}$ in phase two is computed in $O(1)$ time, this phase, and thus the entire 2-p-opt checking sequence, is performed in $O(n^2)$.

Note that this procedure would be much faster than using the full evaluation function (equation (3.3)) for the cost of each 2-p-opt move. In fact, the full evaluation would require $O(n^2)$ time for the cost calculation of each move, and $O(n^4)$ time for the entire 2-p-opt checking sequence (since the neighborhood size of 2-p-opt is $O(n^2)$). Unfortunately, as we show in the following section, the delta evaluation expressed by equation (3.7) is not correct.

3.3.2 Incorrectness of the proposed 2-p-opt recursive equation from [3]

Theorem 1 *Given an instance of the homogeneous PTSP, an a priori tour* $\lambda = (1, 2, ..., i, i+1, ..., i+k, ..., n)$*, and an a priori tour* $\tilde{\lambda} = (1, 2, ..., i-1, i+k, i+k-1, ..., i, i+k+1, ..., n)$*, then* $E[L(\tilde{\lambda})] - E[L(\lambda)] \neq \Delta E_{i,i+k}$ *for* $k \geq 2$*.*

The proof is by example. Let $k = 2$, then, according to (3.7)

$$\begin{aligned}\Delta E_{i,i+2} = \Delta E_{i+1,i+2} \quad & +p^2[(q^{-2}-1)A_{i,3} + (q^2-1)(B_{i,1} - B_{i,n-2}) \\ & +(q^2-1)(A_{i+2,1} - A_{i+2,n-2}) + (q^{-2}-1)B_{i+2,3} \\ & +(q^{n-2}-1)(A_{i,1} - A_{i,2}) + (q^{2-n}-1)B_{i,n-1} \\ & +(q^{2-n}-1)A_{i+2,n-1} + (q^{n-2}-1)(B_{i+2,1} - B_{i+2,2})].\end{aligned} \tag{3.8}$$

We show that in this expression there are arcs with an incorrect probability coefficient. Consider for instance arc $(i+1, i+3)$ and arc $(i+3, i+1)$. We examine the probability coefficients of these arcs, as computed by the recursive equation (3.8). The terms inside square brackets of equation (3.8) only take into account arcs joining respectively i and $i+2$ with other nodes. Therefore, the arcs $(i+1, i+3)$ and $(i+3, i+1)$ are only computed in the first term of equation (3.8):

$$\Delta E_{i+1,i+2} = p^3[q^{-1}A_{i+1,2} - (B_{i+1,1} - B_{i+1,n-1}) - (A_{i+2,1} - A_{i+2,n-1}) + q^{-1}B_{i+2,2}]. \tag{3.9}$$

In the above expression, arc $(i+1, i+3)$ is computed by the term corresponding to $r = 2$ of $A_{i+1,2} = \sum_{r=2}^{n-1}(1-p)^{r-1}d(i+1, i+1+r)$, and arc $(i+3, i+1)$ is computed by the term corresponding to $r = n-2$ of $B_{i+1,1} = \sum_{r=1}^{n-1}(1-p)^{r-1}d(i+1-r, i+1)$. From this observation and from equation (3.9), we see that arc $(i+1, i+3)$ is computed as follows:

$$\frac{p^3}{(1-p)}(1-p)d(i+1, i+3) = p^3 d(i+1, i+3), \tag{3.10}$$

and arc $(i+3, i+1)$ is computed as follows:

$$p^3(1-p)^{n-3}d(i+3, i+1). \tag{3.11}$$

Now we consider the objective function (3.3), and we write the probability coefficients that arcs $(i+1, i+3)$ and $(i+3, i+1)$ must have in $E[L(\lambda)]$ and in $E[L(\tilde{\lambda})]$. When the a priori tour is λ, arc $(i+1, i+3) = (\lambda(i+1), \lambda(i+3))$ and it has a probability coefficient of $p^2(1-p)$, that is, the probability that $\lambda(i+1)$ and $\lambda(i+3)$ require a visit while the single node $\lambda(i+2)$ between them does not require a visit. The same coefficient is valid when the a priori tour is $\tilde{\lambda}$, because $(i+1, i+3) = (\tilde{\lambda}(i+1), \tilde{\lambda}(i+3))$, and there is only one node $(\tilde{\lambda}(i+2))$ between $\tilde{\lambda}(i+1)$ and $\tilde{\lambda}(i+3)$. Figure 3.3 may be of help in visualizing this. Similarly, arc $(i+3, i+1)$ has probability coefficient $p^2(1-p)^{n-3}$ both in λ and in $\tilde{\lambda}$ (since $i+3 = \tilde{\lambda}(i+1-(n-2))$). Therefore arc $(i+1, i+3)$ is computed as follows in $E[L(\tilde{\lambda})] - E[L(\lambda)]$:

$$[p^2(1-p) - p^2(1-p)]d(i+1, i+3) = 0, \tag{3.12}$$

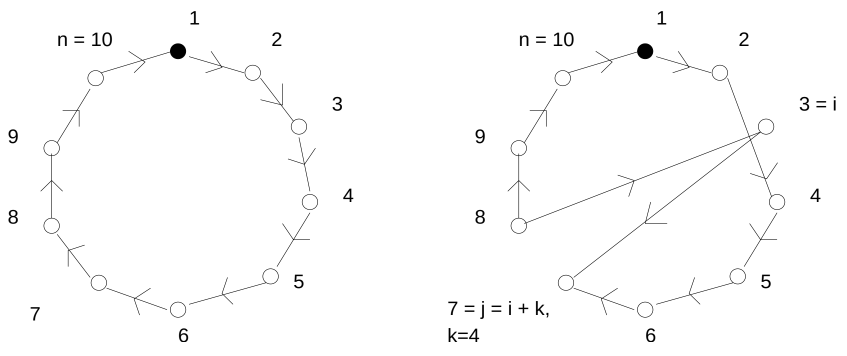


Figure 3.4: Tour $\lambda = 1, 2, ..., i, i+1, ..., j, ..., n$ (left) and tour $\tilde{\lambda} = 1, 2, ..., i-1, i+1, i+2, ..., j, i, j+1, ..., n$ (right) obtained from λ by moving node i to position j and shifting backwards one space the nodes $i+1, ..., j$, with $n = 10$, $i = 3$, $j = 7$.

in contrast with (3.10), and the arc $(i+3, i+1)$ is computed as follows in $E[L(\tilde{\lambda})] - E[L(\lambda)]$:

$$[p^2(1-p)^{n-3} - p^2(1-p)^{n-3}]d(i+3, i+1) = 0, \tag{3.13}$$

in contrast with (3.11). ■

Appendix B presents the step-by-step evaluation of a 2-p-opt move, first using the full evaluation of the objective function (3.3), and second using Bertsimas' proposed expression (3.7).

Note: a proof of the incorrectness of the 2-p-opt expression appearing in [1] is given in appendix A.

3.4 The 1-shift

The 1-shift local search was introduced by Bertsimas in [1] and later published by this journal in an article by Bertsimas and Howell [3]. Given an a priori tour λ, its 1-shift neighborhood is the set of tours obtained by moving a node i to position j of the tour, with the intervening nodes being shifted backwards one space accordingly, as for example in Figure 3.4.

3.4.1 Expressions proposed in [3] for the cost of a 1-shift move

Consider, without loss of generality, a tour $\lambda = 1, 2, ..., i, i+1, ..., j, ..., n$ and a tour $\tilde{\lambda} = 1, 2, ..., i-1, i+1, i+2, ..., j, i, j+1, ..., n$ obtained from λ by moving node i to position j and shifting backwards one space the nodes $i+1, ..., j$. Let $\Delta E'_{i,j}$ denote the change in the expected length $E[L(\tilde{\lambda})] - E[L(\lambda)]$. In the following we reproduce verbatim the expressions given in [3] for the computation of $\Delta E'_{i,j}$.

Let $j = i + k$. Note that, for $k = 1$, the tour $\tilde{\lambda}$ obtained by 1-shift is the same as the one obtained by 2-p-opt. Therefore, for $k = 1$, $\Delta E'_{i,i+1} = \Delta E_{i,i+1}$ and the computation is done according to (3.6). For $k \geq 2$, the proposed expression is:

$$\begin{aligned}\Delta E'_{i,j} = \Delta E'_{i,j-1} \quad & +p^2[(q^{-k} - q^{-(k-1)})A_{i,k+1} + (q^k - q^{k-1})(B_{i,1} - B_{i,n-k}) \\ & +(q-1)(A_{j,1} - A_{j,n-k}) + (q^{-1} - 1)B_{j,k+1} \\ & +(q^{n-k} - q^{n-(k-1)})(A_{i,1} - A_{i,k}) + (q^{k-n} - q^{(k-1)-n})B_{i,n-k+1} \\ & +(1 - q^{-1})A_{j,n-k+1} + (1-q)(B_{j,1} - B_{j,k})],\end{aligned} \tag{3.14}$$

where A and B are the same matrices of partial results considered for the 2-p-opt and defined by (3.5). This algorithm follows the same lines as the 2-p-opt algorithm: all phase one computations, including the accumulation of matrices A and B, proceed in the same way. The second phase of the local search consists of computing $\Delta E'_{i,j}$ recursively by means of equation (3.14). Like 2-p-opt, since each $\Delta E'_{i,j}$ in phase two is computed in $O(1)$ time, this phase, and thus the entire 1-shift checking sequence, is performed in $O(n^2)$. Similarly to the 2-p-opt, note that this procedure would be much faster than using full evaluation function (equation (3.3)) for the cost of each 1-shift move. In fact, the full evaluation would require $O(n^2)$ time for the cost calculation of each move, and $O(n^4)$ time for the entire 1-shift checking sequence (since the neighborhood size of 1-shift is $O(n^2)$). Unfortunately, as we show in the following section, the delta evaluation expressed by equation (3.14) is not correct.

3.4.2 Incorrectness of the proposed 1-shift recursive expression

Theorem 2 *Given an instance of the homogeneous PTSP, an a priori tour* $\lambda = (1, 2, ..., i, i+1, ..., i+k, ..., n)$*, and an a priori tour* $\tilde{\lambda} = (1, 2, ..., i-1, i+1, i+2, ..., i+k, i, i+k+1, ...n)$*, then* $E[L(\tilde{\lambda})] - E[L(\lambda)] \neq \Delta E'_{i,i+k}$ *for* $k \geq 2$.

The proof is by example. Let $k = 2$, then, according to (3.14)

$$\begin{aligned}\Delta E'_{i,i+2} = \Delta E'_{i,i+1} \quad & +p^2[(q^{-2} - q^{-1})A_{i,3} + (q^2 - q)(B_{i,1} - B_{i,n-2}) \\ & +(q-1)(A_{i+2,1} - A_{i+2,n-2}) + (q^{-1} - 1)B_{i+2,3} \\ & +(q^{n-2} - q^{n-1})(A_{i,1} - A_{i,2}) + (q^{2-n} - q^{1-n})B_{i,n-1} \\ & +(1 - q^{-1})A_{i+2,n-1} + (1-q)(B_{i+2,1} - B_{i+2,2})].\end{aligned} \tag{3.15}$$

We show that in this expression there are arcs with an incorrect probability coefficient. Consider for instance arc $(i, i+1)$ and arc $(i+1, i)$. We examine the

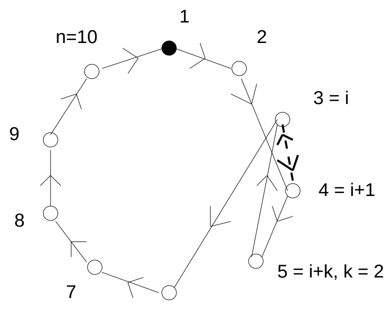


Figure 3.5: Tour $\tilde{\lambda}$, with $\tilde{\lambda}(i) = i+1$, $\tilde{\lambda}(i+1) = i+2$, and $\tilde{\lambda}(i+2) = i$. The dashed line represents arcs $(i, i+1)$ and $(i+1, i)$.

probability coefficients of these arcs, as computed by the recursive equation (3.15). The first term of (3.15) is $\Delta E'_{i,i+1}$ and it is equal to (3.6). By an inspection of the terms A and B in equation (3.6) it is not difficult to see that arcs $(i, i+1)$ and $(i+1, i)$ are not computed by $\Delta E'_{i,i+1}$. Arc $(i, i+1)$ is computed by the term $(q^{n-2} - q^{n-1})(A_{i,1} - A_{i,2})$ of equation (3.15), that is,

$$p^2((1-p)^{n-2} - (1-p)^{n-1})d(i, i+1), \tag{3.16}$$

and arc $(i+1, i)$ is computed by the term $(q^{2-n} - q^{1-n})B_{i,n-1}$ of equation (3.15), that is,

$$\frac{p^3}{(1-p)}d(i+1, i). \tag{3.17}$$

Now we consider the objective function (3.3), and we write the probability coefficients that arcs $(i, i+1)$ and $(i+1, i)$ must have in $E[L(\lambda)]$ and in $E[L(\tilde{\lambda})]$. When the a priori tour is λ, arc $(i, i+1) = (\lambda(i), \lambda(i+1))$ and it has a probability coefficient of p^2, that is, the probability that $\lambda(i)$ and $\lambda(i+1)$ require a visit. When the a priori tour is $\tilde{\lambda}$, we have $(i, i+1) = (\tilde{\lambda}(i+2), \tilde{\lambda}(i)) = (\tilde{\lambda}(i-(n-2)), \tilde{\lambda}(i))$, which leads to the probability coefficient $p^2(1-p)^{n-3}$ (see Figure 3.5). The difference of the probability coefficients of $(i, i+1)$ in $\tilde{\lambda}$ and λ leads to

$$p^2((1-p)^{n-3} - 1)d(i, i+1), \tag{3.18}$$

which is different from (3.16). Let us now focus on arc $(i+1, i)$. When the a priori tour is λ, arc $(i+1, i) = (\lambda(i-(n-1)), \lambda(i))$ and it has a probability coefficient of $p^2(1-p)^{n-2}$. When the a priori tour is $\tilde{\lambda}$, we have $(i+1, i) = (\tilde{\lambda}(i), \tilde{\lambda}(i+2))$ (see Figure 3.5), and the probability coefficient is $p^2(1-p)$. The difference of the probability coefficients of $(i+1, i)$ in $\tilde{\lambda}$ and λ leads to:

$$p^2(1-p-(1-p)^{n-2})d(i+1, i), \tag{3.19}$$

which is different from (3.17). ■

Appendix C presents the step-by-step evaluation of a 1-shift move, first using the full evaluation of the objective function (3.3), and second using Bertsimas' proposed expression (3.14).

Chapter 4

Approximations of the objective function

As we have seen, the objective function of the PTSP is computationally expensive, since it needs to consider all the $O(n^2)$ edges joining couples of customers. For this reason, it is worth investigating the design of fast approximations of the objective function, to be used in optimization algorithms for speeding up, and thus possibly improving, the optimization process.

In this chapter we present and analyze two types of objective function approximations. The first approximation (section 4.1) is based on the observation that, for a given a priori solution, some edges have a very small probability of being actually travelled, and therefore they may be neglected. We call this type of approximation 'depth-approximation' (this naming convention will become clear in the remainder of this chapter). The second type of approximation (section 4.2) is based on the observation that the objective function computes the expected value of a quantity, and therefore it may be estimated by a sampling technique. We call this second type of approximation 'sampling-approximation'. In section 4.3 we experimentally analyze the accuracy of the two types of approximations under different conditions.

4.1 Depth-approximation

We say that and edge e_{ij} has a depth $\theta \in [0, n-2]$ with respect to a given a priori tour λ if there are exactly θ cities on the tour between i and j. A high depth θ for an edge implies a small probability for the edge to be actually part of a realized tour, since this would mean that a large number θ of consecutive customers do not require a visit. Therefore, edges with higher depth should impact less on the objective value of an a priori tour. Let us now be more precise.

Given an a priori tour λ, and depth values $\theta = 0, 1, ..., n-2$, the edges of the PTSP instance under consideration may be grouped in sets $\lambda^{(\theta)}$, each set containing edges of the same depth. Note that the edge set $\lambda^{(\theta)}$ contains a number of subtours (that is, tours which visit a subset of the customers) equal to gcd(n, $\theta + 1$) [1]. For instance, the set $\lambda^{(0)}$ contains exactly the edges of the a priori tour. In general, however, it may be the case that the subtours formed in such a way can never be realized under the a priori strategy. As an example of edge partition according to the depth, Fig. 4.1 shows, from left to right, an a priori tour λ (which coincides with the set of edges $\lambda^{(0)}$) , $\lambda^{(1)}$ and $\lambda^{(2)}$ for a PTSP with 8 customers. Note that $\lambda^{(1)}$ contains two subtours which could be actually realized under the a priori strategy, but $\lambda^{(2)}$ contains one single tour that visits all customers, and therefore cannot be realized under the a priori strategy (since, if all customers are to be visited, then they are visited according to the a priori tour λ).

Given the partition of edges according to their depth $\theta = 0, 1, 2, ..., n-2$, the PTSP objective function may be written as a sum of terms of increasing depth:

$$E[L(\lambda)] = \sum_{\theta=0}^{n-2} \left[\sum_{j=1}^{n} d_{\lambda(j),\lambda(j+\theta+1)} \cdot p_{\lambda(i)} p_{\lambda(i+\theta+1)} \prod_{k=1}^{\theta} (1 - p_{\lambda(i+k)}) \right] , \tag{4.1}$$

where $\lambda = (\lambda(1), \lambda(2), ..., \lambda(n))$ is the a priori tour. In the special class of PTSP instances where $p_i = p$ for all customers $i \in V$ (the homogeneous PTSP), equation (4.1) may be written as

$$E[L(\lambda)] = \sum_{\theta=0}^{n-2} L(\lambda^{(\theta)}) p^2 (1-p)^{\theta} \tag{4.2}$$

where $L(\lambda^{(\theta)}) \equiv \sum_{j=1}^{n} d(j, (j+1+r))$. The $L(\lambda^{(\theta)})$'s have the combinatorial interpretation of being the sum of lengths of the collection subtours in $\lambda^{(\theta)}$.

It is now easy to see how the probability of an edge of depth θ to be used decreases with θ. In the homogeneous PTSP (equation 4.2) this probability is $p^2(1-p)^{\theta}$, and in the heterogeneous PTSP (equation 4.1) it also involves a product over $\theta + 2$ probability coefficients. Therefore, the probability of an edge of depth θ to be used (and therefore the weight of such and edge in the objective value) decreases *exponentially* with θ. The idea, in the depth-approximation, is to stop the evaluation

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

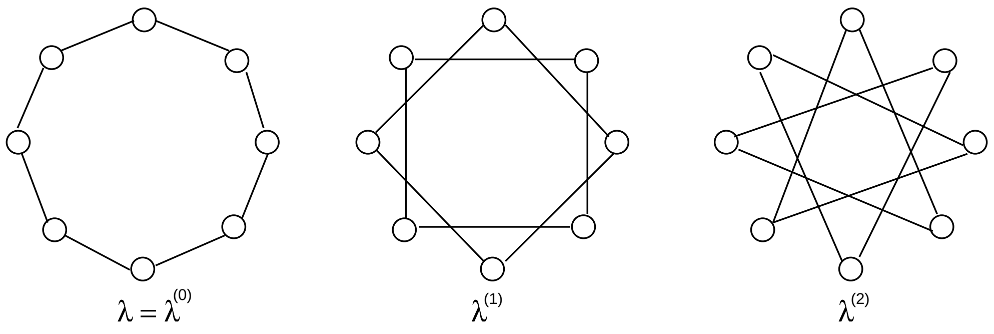


Figure 4.1: Edges grouped in sets according to their depth θ. The picture shows the sets with $\lambda^{(\theta)}$ with $\theta = 0, 1, 2$. Note that $\lambda(0)$ corresponds to the set of edges forming the a priori tour.

of the objective function after a fixed depth has been reached. We define

$$E^\theta[L(\lambda)] = \sum_{r=0}^{\theta} \left[\sum_{j=1}^{n} d_{\lambda(j),\lambda(j+r+1)} \cdot p_{\lambda(i)} p_{\lambda(i+r+1)} \prod_{k=1}^{r} (1 - p_{\lambda(i+k)}) \right], \tag{4.3}$$

and for the homogeneous PTSP this simplifies to

$$E^\theta[L(\lambda)] = \sum_{r=0}^{\theta} L(\lambda^{(r)}) p^2 (1-p)^r. \tag{4.4}$$

The time complexity of the depth-approximation is $O(\theta n)$, therefore, the smallest the θ, the fastest is the computation of the depth-approximation respect to the $O(n^2)$ exact objective function. Nevertheless, the smallest the θ, the bigger is the difference, or error, between approximated and exact computation. The main question here is how the quality of the depth-approximation degrades by decreasing the depth of the computation, and for different types of PTSP instances. This is discussed in section 4.3.

4.2 Sampling-approximation

Here, we propose an approximation to the PTSP objective function which is based on the observation that the objective function computes an expected value of a random quantity, and therefore it may be estimated by a sampling technique. More precisely, given an a priori tour λ, the objective function may be written as (see also section 1)

$$E[L(\lambda)] = \sum_{S \subseteq V} p(S) L(\lambda_{|S}). \tag{4.5}$$

In the above expression, S is a subset of the set of customers V, $L(\lambda_{|S})$ is the length of the tour λ, pruned in such a way as to only visit the customers in S, skipping the others, and $p(S)$ is the probability for the subset of customers S to require a visit:

$$p(S) = \prod_{i \in S} p_i \prod_{i \in V-S} (1 - p_i). \tag{4.6}$$

The objective function, as expressed by equation 4.5, computes the expected value of the tour λ, over all possible random subsets of customers.

The idea, in sampling-approximation, is to *estimate* the exact expected value 4.5 through sampling, in the following way. The length $L(\lambda)$ is a discrete random variable, taking the value $L(\lambda_{|S})$ with probability $p(S)$. Let S_i, $i \in 1, 3, ..., N$ be subsets of the original n customers sampled with probability $p(S_i)$. If each subset S_i is sampled independently of the others, the Central Limit Theorem implies that

$$\frac{1}{N} \sum_{i=1}^{N} L(\lambda_{|S_i}) \approx E[L(\lambda)], \tag{4.7}$$

for N big enough.

The time complexity of the sampling-approximation is $O(Nn)$, therefore, the smallest the sample size N, the fastest is the computation of the sampling-approximation respect to the $O(n^2)$ exact objective function. Nevertheless, the smallest the N, the bigger is the difference, or error, between approximated and exact computation. The main question here is how the quality of the sampling-approximation degrades by decreasing the number of samples, and for different types of PTSP instances.

There are two ways of evaluating the quality of sampling-approximation. One is theoretical, by exploiting the theorems of statistics about the precision and confidence intervals of an estimator. The problem, with this, is that it may give pessimistic indications of the estimation accuracy. In fact, in order to exploit theorems of statistics, one should be able to approximate the probability distribution of the length $L(\lambda_{|S})$ by a normal distribution. This is true, in general, for a number of sample N at least equal to 30 (this is an empirical rule which usually applies to distributions that are not too asymmetric). In our case, we may be lucky, and have a good approximation already for smaller N, but by using theoretical investigations we would not know this.

The other way for evaluating the quality of sampling-approximation is experimental, like in the case of depth-approximation. We can use this option because we may know the exact value of the objective function. This situation is different from that of the natural sciences where the exact value being estimated, in general, is not known. The experimental analysis of sampling-approximation accuracy is performed in section 4.3.

4.3 Analysis of depth- and sampling-approximation techniques

In the literature the use of the depth-approximation has been first suggested by Jaillet [24], without performing experiments. The only application of depth-approximation we are aware of is a recent paper [10], where the approximation is used in the Ant Colony Optimization framework with promising results. The idea of sampling as been applied in [2] for computing a lower bound on the optimal PTSP value, but it has never been used for approximating the objective function.

In the cited literature there is no analysis of the tradeoff between depth and quality of the depth-approximation. We feel that, before using any approximation of the objective function inside an optimization algorithm, it is useful to know as much as possible about the quality of the approximation.

A number of questions may be asked about the quality of the approximation. The main questions we would like to answer are the following:

1. suppose that, in an optimization algorithm, we want to have an objective function approximated by Δ percent error; how big should be then the depth $\theta(\Delta)$ (or number of samples $N(\Delta)$) of our depth-approximation (or sampling-approximation)?
2. is $\theta(\Delta)$ (or $N(\Delta)$) also dependent on the particular PTSP instance and a priori tour?

We try to answer these questions by experimental investigation. PTSP instances were generated starting from TSP instances (from the TSPLIB benchmark [32]) and assigning to each customer i a probability p_i of requiring a visit. We have tested both homogeneous and heterogeneous PTSP instances. In homogeneous PTSP instances, we set $p_i = p$ for all customers, with p varying from 0.1 to 0.9. We realized heterogeneous PTSP instances by generating customers probabilities randomly, according

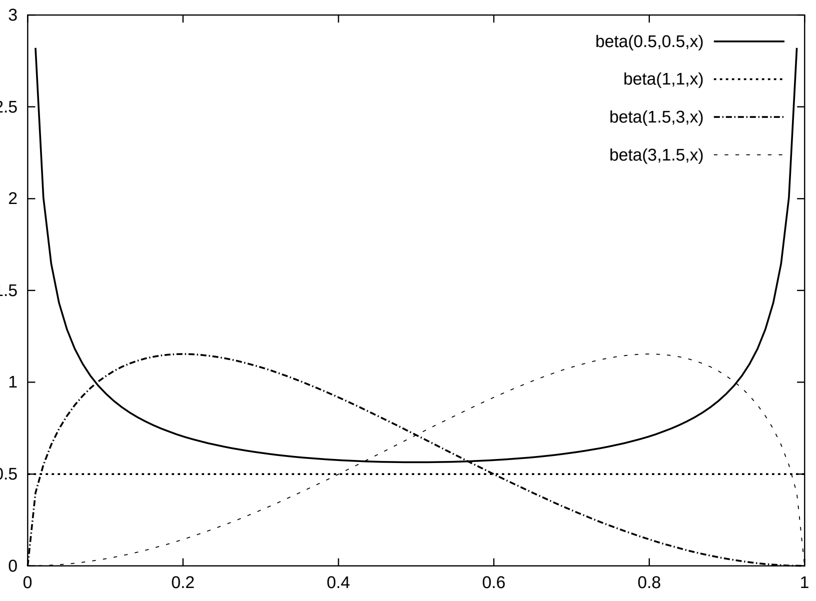


Figure 4.2: Probability distributions used for generating random customers probabilities.

to the probability distribution $\beta(a;b)$, which is defined as follows

$$\beta(a;b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} p^{a-1}(1-p)^{b-1}, \tag{4.8}$$

with $p \in [0,1]$. By choosing the parameters a and b of the β distribution, one can set the average customer probability $\overline{p}$ and the variance σ_p around the average value:

$$\overline{p} = \frac{a}{a+b} \tag{4.9}$$

$$\sigma_p = \frac{ab}{(a+b)^2(a+b+1)} \tag{4.10}$$

In order to model different configurations of customers probabilities, we generated customers probabilities according to the distributions $\beta(0.5;0.5)$, $\beta(1;1)$, $\beta(1.5;3)$, and $\beta(3;1.5)$, whose shapes are as shown in Figure 4.2, and whose average and deviation, computed according to 4.9 and 4.10, are as in Table 4.1.

Type of convergence Figure 4.3 shows how depth- and sampling-approximation converge to the exact objective value, by increasing, respectively, the depth and

	$\overline{p}$	σ_p
$\beta(0.5; 0.5)$	0.5	0.13
$\beta(1; 1)$	0.5	0.08
$\beta(1.5; 3)$	0.33	0.04
$\beta(3; 1.5)$	0.67	0.04

Table 4.1: Average customers probability $\overline{p}$ and standard deviation of customers probability σ_p implied by different choices of the β distribution parameters.

the number of samples in the computation. Figure 4.3 is relative to one particular instance of the homogeneous PTSP, but the following observations are valid in general.

Both depth- and sampling-approximation may be good already for small values of the depth and number of samples, but depth-approximation converges earlier than sampling-approximation. Another difference between the two approximation techniques is that sampling-approximation may overestimate the exact objective value, while depth-approximation is always less than or equal to the exact objective value. Moreover, the quality of sampling-approximation is a random quantity, and it may happen that a bigger number of samples does not imply a better approximation (even if, asymptotically, convergence to the exact objective value would be reached).

Dependence of approximation quality upon customers probabilities Given a certain set of customers (that is, a TSP instance) we are interested to see if the quality of approximation is affected by different configurations of customers probabilities. Due to the nature of the approximations, it is reasonable to expect that depth-approximation is quite sensitive to the probabilities of customers requiring a visit, while sampling-approximation may be less affected. Such hypothesis are confirmed in the case of homogeneous PTSP, where the depth-approximation quality degrades for instances with small customers probability, while sampling-approximation maintain a good quality over all range of probabilities. This is shown in Figure 4.4. In case of heterogeneous PTSP instances, both depth- and sampling-approximation are sensitive to the customers probabilities, as shown in Figure 4.5. In particular, the probability configuration which always has the worst approximation quality is the one which follows the distribution $\beta(1.5; 3)$. Note that $\beta(1.5; 3)$ corresponds to a probability configuration where a high proportion of customers have a low probability of requiring a visit (see average customers probability and standard deviation

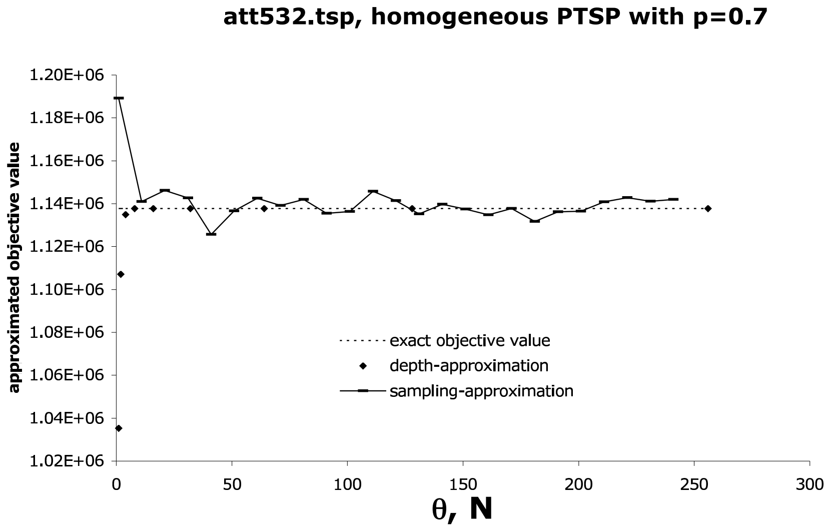


Figure 4.3: Depth- and sampling-approximation of the objective function as a function of, respectively, depth and number of samples. The figure refers to a homogeneous PTSP instance with probability 0.7 and 532 customers, whose coordinates are from the att532.tsp instance of the TSPLIB. The objective function has been computed on a randomly generated a priori solution. For a depth and number of samples equal to 10, which is very small, as compared to the number of customers, the approximation, in this case, is already very good.

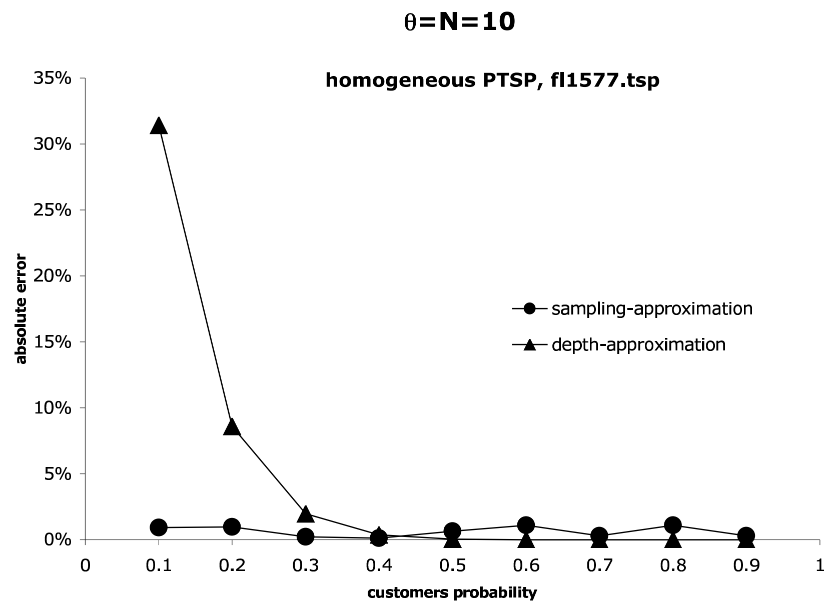


Figure 4.4: Absolute value of the error performed by the depth- and sampling-approximation as a function of the customers probability, in the homogeneous PTSP, and for a 1577-customers TSP instance of the TSPLIB. The objective function has been computed on a randomly generated a priori solution. The error is expressed as a percentage respect to the exact objective value.

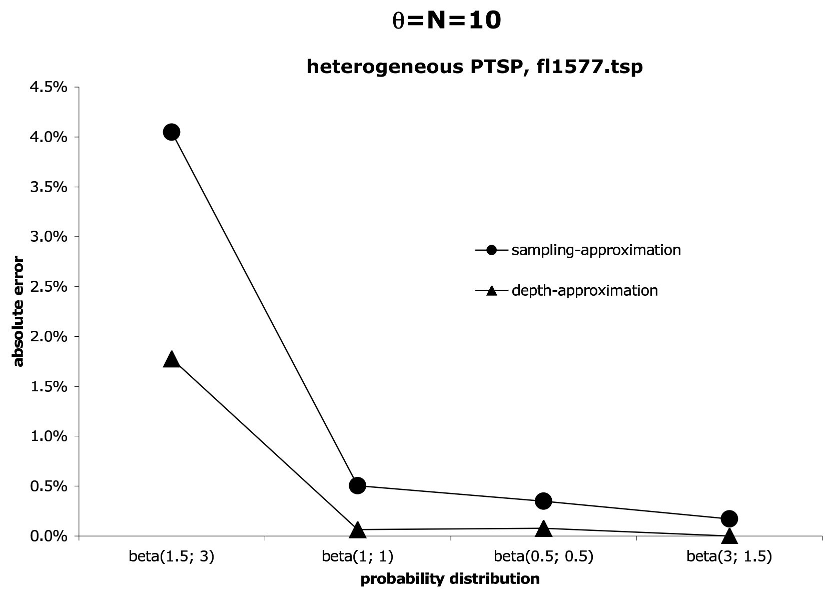


Figure 4.5: Absolute value of the error performed by the depth- and sampling-approximation for different customers demands probability configurations (heterogeneous PTSP) for a 1577-customers TSP instance of the TSPLIB. The error is expressed as a percentage respect to the exact objective value.

of $\beta(1.5; 3)$ respect to other distributions in Table 4.1). Considering both the homogeneous and the heterogeneous PTSP, we may argue that, for depth-approximation, a high proportion of customers with low probability of requiring a visit decreases the approximation quality (for a fixed depth).

Dependence of approximation quality upon TSP instances Given a certain depth or number of samples for the approximation, and given a certain customers probability configuration, we are interested to see if the quality of approximation is affected by different sets of customers (that is, TSP instances). The quality of depth-approximation is not much sensitive to the the number of customers, as it appears from Figures 4.6, which compares depth-approximation quality for ten instances derived from the TSPLIB, with a number of customers ranging from 51 to 1577. Among the instances of Figure 4.6, the error of depth-approximation is quite small, always under 2.5%.

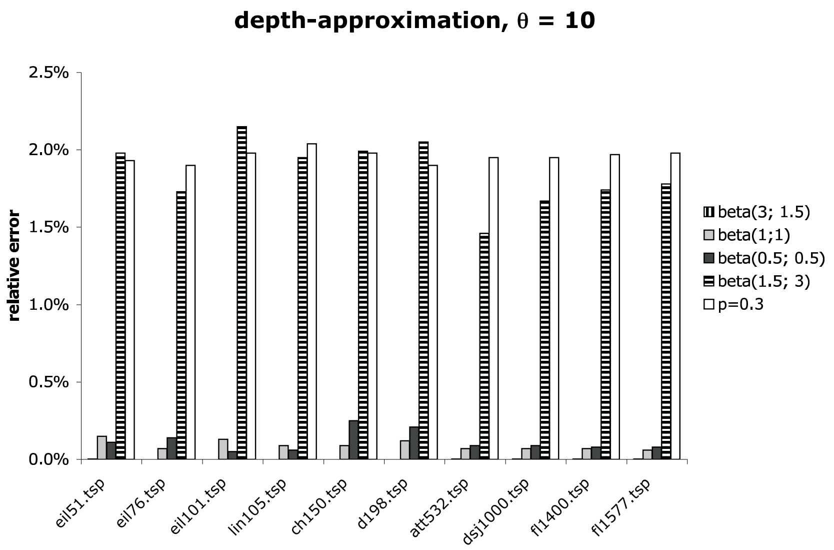


Figure 4.6: Error performed by the depth-approximation for different instances from the TSPLIB, with customers probabilities generated according to β probability laws (heterogeneous PTSP) and for the homogeneous PTSP with customers probability $p = 0.3$.

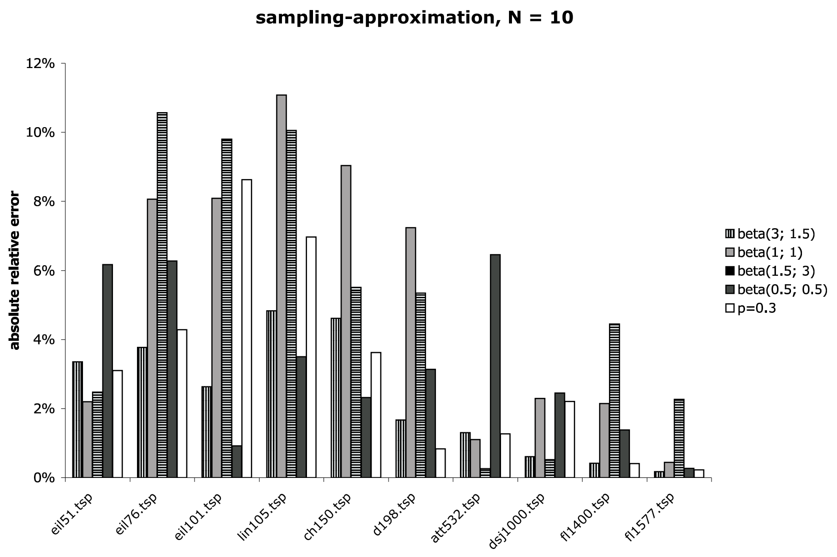


Figure 4.7: Error performed by the sampling-approximation for different instances from the TSPLIB, with customers probabilities generated according to β probability laws (heterogeneous PTSP), and for the homogeneous PTSP with customers probability $p = 0.3$.

The situation, with sampling-approximation, is different, as shown in Figure 4.7. One counterintuitive feature is that, in many cases, an instance with more customers is better approximated than an instance with less customers. For example, compare eil76.tsp with fl1577.tsp in Figures 4.7. For heterogeneous PTSPs, differently from depth-approximation, we cannot say whether there is a β distribution which is worse or better than others, and the worst error, for 10 samples, may range from around 12% to around 2%. In the homogeneous PTSP instances (with $p = 0.3$), the error range is also quite wide, from around 10% to less than 1%.

4.3.1 Conclusions and guidelines

Given an instance with n customers, it may be convenient to use an objective function approximation if this is both fast, and it gives an acceptable error. Let us assume that an approximation is fast enough if it runs in $O(10 \cdot n)$ time. This is to say that we consider a depth and a number of samples equal to 10 for, respectively, depth- and sampling-approximation. Will than the error be acceptable, on the base of the above analysis? Let us assume, here, that an error is acceptable if it is not bigger than 3%. With these assumptions, we can say that depth-approximation will almost always produce an acceptable error. The only situation where sampling-approximation may be better, is in homogeneous PTSP instances with low customers probability (say, a probability smaller than 0.3), but this is not valid for all TSP instances, therefore the quality of sampling-approximation should be tested before relying on it.

It would be interesting to further investigate the relation between the depth or number of samples and the quality of the approximation. In particular, it would be useful to find a *functional* relation between the depth or number of samples and the approximation quality. This would let us know the smallest depth or number of samples necessary to achieve a certain chosen precision.

Outlook and future work

In this thesis we studied the design of a good heuristic for the PTSP, which composed by two modules: a solution construction and a solution improvement (or local search) algorithm. We studied separately two modules that, once integrated and iteratively alternated, would constitute the final heuristic algorithm.

As far as the solution construction algorithm is concerned, we investigated the potentialities of pACS, a particular ant colony optimization algorithm, for the homogeneous PTSP. We showed that the pACS algorithm is a promising solution construction heuristic for the PTSP. We compared pACS with other tour construction heuristics and we provided an estimation of the absolute error with respect to the optimal PTSP solution for some instances. We also compared pACS to ACS (an ant colony optimization algorithm for the TSP), and we showed that for customers probability close to 1, the ACS heuristic is a better alternative than pACS.

The study of an efficient local search for the PTSP, should greatly improve the solution quality of pACS and of any tour construction heuristic in general, and it is an important direction of research. We have shown that the current algorithms in the literature, 2-p-opt and 1-shift, are inadequate. This results poses the question whether a new fast local search operator for the PTSP can be derived. One possibility to design fast local search is to use an approximation of the objective function, which is faster to be computed than the exact one. Two types of approximations are proposed and analyzed in this thesis.

The next step of this research will be to integrate the solution construction pACS algorithm with the local search algorithms, with the aim of obtaining state-of-the art results for the PTSP. If the TSP inspired 2-p-opt and 1-shift local search are used, an approximated objective function evaluation should be employed for computing the cost of a local search move, in order to speed up the computation. An alternative would be to design a different local search operator which allow for an exact and fast computation of the move cost, but the design of such a local search is a completely open question.

Bibliography

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

[2] D. J. Bertsimas, P. Chervi, and M. Peterson. Computational approaches to stochastic vehicle routing problems. *Transportation Sciences*, 29(4):342–352, 1995.

[3] D. J. Bertsimas and L. Howell. Further results on the probabilistic traveling salesman problem. *European Journal of Operational Research*, 65:68–95, 1993.

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

[5] D. J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44(2):216–304, 1996.

[6] L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 2002.

[7] L. Bianchi, L. M. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 176–187. Springer, Berlin, Germany, 2002.

[8] L. Bianchi and J. Knowles. Local search for the probabilistic traveling salesman problem: a proof of the incorrectness of bertsimas' proposed 2-p-opt and 1-shift

algorithms. Technical Report IDSIA-21-02, IDSIA, Lugano, Switzerland, 2002. Submitted to European Journal of Operational Research.

[9] J. Bramel and D. Simchi-Levi. *The Logic of Logistics*. Springer, Berlin, Germany, 1997.

[10] J. Branke and M. Guntsch. New ideas for applying ant colony optimization to the probabilistic traveling salesman problem. In *Proceedings of EvoCOP 2003 – Third European Workshop on Evolutionary Computation in Combinatorial Optimization*, Lecture Notes in Computer Science, pages –, 2003.

[11] T. G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97:409–438, 1997.

[12] 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.

[13] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, DEI, Politecnico di Milano, Italy, 1992. in Italian.

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

[15] 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*. McGraw-Hill, 1999.

[16] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

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

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

[19] M. Fisher. In O. M. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, chapter Vehicle Routing, pages 1–33. Elsevier, Amsterdam, Holland, 1995.

[20] L. M. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*, pages 622–627. IEEE Press, Piscataway, NJ, 1996.

[21] B. L. Golden and A. A. Assad, editors. *Vehicle Routing: Methods and Studies.* Elsevier, Amsterdam, Holland, 1988.

[22] IDSIA - Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Manno, Switzerland. http://www.idsia.ch.

[23] IRIDIA - Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Bruxelles, Belgium. http://iridia.ulb.ac.be.

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

[25] 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.

[26] A. Jézéquel. *Probabilistic Vehicle Routing Problems.* Master's thesis, MIT, Cambridge, MA, 1985.

[27] G. Laporte, F. Louveaux, and H. Mercure. An exact solution for the a priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.

[28] S. Lin. Computer solutions of the traveling salesman problem. *Bell Systems Journal*, 44:2245–2269, 1965.

[29] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*, chapter 19 Local Search. Dover Publications, 1982.

[30] F. A. Rossi and I. Gavioli. *Aspects of Heuristic Methods in the Probabilistic Traveling Salesman Problem*, pages 214–227. World Scientific, Singapore, 1987.

[31] P. Toth and D. Vigo, editors. *The vehicle routing problem.* SIAM Monographs on Discrete Mathematics, 2002.

[32] http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95/.

Appendix A

Proof that 2-p-opt as proposed in [1] is incorrect

A.1 Expressions proposed in [1] for the cost of a 2-p-opt move

In Bertsimas' PhD thesis [1], the delta evaluation expression for the 2-p-opt local seach is reported in a slightly different form than that in [3] (equation (3.7)), all the rest (A, B, and $\Delta E_{i,i+1}$) being the same. In the following we reproduce verbatim the recursive equation given in [1] for the computation of $\Delta E_{i,j}$. For $j = i + k$ and $k \geq 2$:

$$\begin{aligned}\Delta E_{i,j} = \Delta E_{i+1,j-1} \quad & +p^2[(q^{-k}-1)A_{i,k+1} + (q^k - 1)(B_{i,1} - B_{i,n-k}) \\ & +(q^k-1)(A_{j,1} - A_{j,n-k}) + (q^{-k}-1)B_{j,k+1} \\ & +(q^{n-k}-1)(A_{i,1} - A_{i,k}) + (q^{k-n}-1)B_{i,n-k+1} \\ & +(q^{k-n}-1)A_{j,n-k+1} + (q^{n-k}-1)(B_{j,1} - B_{j,k})].\end{aligned} \tag{A.1}$$

Equation (A.1) differs from (3.7) by the first term on the right side: here this term is $\Delta E_{i+1,j-1}$, while in (3.7) it is $\Delta E_{i+1,j}$. We observe that the recursive equation (A.1) is correct for $k = 2$, and that in this case the first term on the right side is zero, since $\Delta E_{i+1,i+k-1} = \Delta E_{i+1,i+1}$. Unfortunately, equation (A.1) is not correct for $k \geq 3$, as shown in the following theorem.

A.2 Incorrectness of the proposed 2-p-opt recursive equation from [1]

Theorem 3 *Given an instance of the homogeneous PTSP, an a priori tour* $\tau = (1, 2, ..., i, i+1, ..., i+k, ..., n)$*, and an a priori tour* $\tilde{\tau} = (1, 2, ..., i-1, i+k, i+k-1, ..., i, i+k+1, ..., n)$*, then* $E[L(\tilde{\tau})] - E[L(\tau)] \neq \Delta E_{i,i+k}$ *for* $k \geq 3$*.*

The proof is again by example. Let $k = 3$, then, according to (A.1)

$$\begin{aligned}\Delta E_{i,i+3} = \Delta E_{i+1,i+2} \; &+p^2[(q^{-3}-1)A_{i,4} + (q^3-1)(B_{i,1} - B_{i,n-3}) \\ &+(q^3-1)(A_{i+3,1} - A_{i+3,n-3}) + (q^{-3}-1)B_{i+3,4} \\ &+(q^{n-3}-1)(A_{i,1} - A_{i,3}) + (q^{3-n}-1)B_{i,n-2} \\ &+(q^{3-n}-1)A_{i+3,n-2} + (q^{n-3}-1)(B_{i+3,1} - B_{i+3,3})].\end{aligned} \tag{A.2}$$

We show that in this expression there are arcs with an incorrect probability coefficient. Consider for instance arc $(i, i+1)$ and arc $(i+1, i)$. We examine the probability coefficients of these arcs, as computed by the recursive equation (A.2). Arcs $(i, i+1)$ and $(i+1, i)$ are computed both by the term $\Delta E_{i+1,i+2}$ and by the terms inside the square brackets of equation (A.2). Let us first focus on the term $\Delta E_{i+1,i+2}$, from (3.6) we have:

$$\Delta E_{i+1,i+2} = p^3[q^{-1}A_{i+1,2} - (B_{i+1,1} - B_{i+1,n-1}) - (A_{i+2,1} - A_{i+2,n-1}) + q^{-1}B_{i+2,2}]. \tag{A.3}$$

In the above expression, arc $(i+1, i)$ is computed by the term corresponding to $r = n-1$ of $A_{i+1,2} = \sum_{r=2}^{n-1}(1-p)^{r-1}d(i+1, i+1+r)$, and arc $(i, i+1)$ is computed by the term corresponding to $r = 1$ of $B_{i+1,1} = \sum_{r=1}^{n-1}(1-p)^{r-1}d(i+1-r, i+1)$. From this observation we see that arc $(i+1, i)$ is computed as follows by the term $\Delta E_{i+1,i+2}$ of (A.2):

$$\frac{p^3}{(1-p)}(1-p)^{n-2}d(i+1, i) = p^3(1-p)^{n-3}d(i+1, i), \tag{A.4}$$

and arc $(i, i+1)$ is computed as follows by the term $\Delta E_{i+1,i+2}$ of (A.2):

$$-p^3 d(i, i+1). \tag{A.5}$$

Let us now focus on the terms inside the square brackets of equation (A.2), which also compute arcs $(i, i+1)$ and $(i+1, i)$. Arc $(i, i+1)$ is computed by the term corresponding to $r = 1$ of $A_{i,1}$, while arc $(i+1, i)$ is computed by the term corresponding to $r = n-1$ of $B_{i,n-2}$. From this observation we see that arc $(i, i+1)$ is computed as follows (by the terms inside the square brackets of equation (A.2)):

$$p^2[(1-p)^{n-3} - 1]d(i, i+1), \tag{A.6}$$

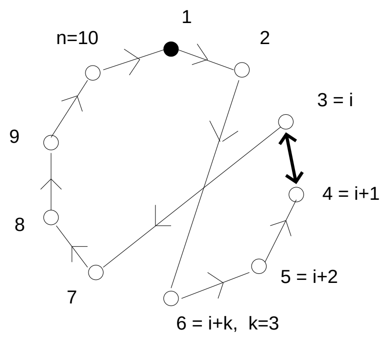


Figure A.1: Tour $\tilde{\tau}$, with $\tilde{\tau}(i) = i+3$, $\tilde{\tau}(i+1) = i+2$, $\tilde{\tau}(i+2) = i+1$, and $\tilde{\tau}(i+3) = i$. The bold line represents arcs $(i, i+1)$ and $(i+1, i)$.

and arc $(i+1, i)$ is computed as follows (by the terms inside the square brackets of equation (A.2)):

$$p^2(1-p)^{n-2}[(1-p)^{3-n} - 1]d(i+1, i). \tag{A.7}$$

Now, by summing (A.5) with (A.6) we get Bertsimas' equations coefficient for arc $(i, i+1)$:

$$p^2[(1-p)^{n-3} - p - 1]d(i, i+1), \tag{A.8}$$

while, by summing (A.4) with (A.7) we get Bertsimas' coefficient for arc $(i+1, i)$:

$$p^2[(1-p)^{n-3} - p + 1]d(i+1, i). \tag{A.9}$$

Now we consider the objective function (3.3), and we write the probability coefficients that arcs $(i, i+1)$ and $(i+1, i)$ must have in $E[L(\tau)]$ and in $E[L(\tilde{\tau})]$. When the a priori tour is τ, arc $(i, i+1)$ has a probability coefficient p^2. When the a priori tour is $\tilde{\tau}$, arc $(i, i+1)$ has a probability coefficient $p^2(1-p)^{n-2}$. Figure A.1 may be of help in visualizing this. Similarly, arc $(i+1, i)$ has probability coefficient $p^2(1-p)^{n-2}$ in τ and a probability coefficient p^2 in $\tilde{\tau}$. Therefore arc $(i, i+1)$ is computed as follows in $E[L(\tilde{\tau})] - E[L(\tau)]$:

$$p^2[(1-p)^{n-2} - 1]d(i, i+1), \tag{A.10}$$

in contrast with (A.8), and the arc $(i+1, i)$ is computed as follows in $E[L(\tilde{\tau})] - E[L(\tau)]$:

$$p^2[1 - (1-p)^{n-2}]d(i+1, i), \tag{A.11}$$

in contrast with (A.9). ■

Appendix B

Example of a 2-p-opt move evaluation

We place the nodes of the graph (customers) on the vertices of a regular hexagon in the Euclidean plane of side 1 so that all distances between pairs of nodes are from the set, $\{1, 2, \sqrt{3}\}$. We set $p = 0.5$.

Let us calculate the difference in expected length of the tours $\tau = 1, 2, 3, 4, 5, 6$ and $\tilde{\tau} = 1, 5, 4, 3, 2, 6$. We shall first do this by calculating, by equation (3.3), the expected length of each and subtracting one from the other. Then we shall compare this result with the calculation using Bertsimas' recursive equation (3.7).

The expected length of the tour $\tau = 1, 2, 3, 4, 5, 6$ is given by:

$$\begin{aligned}
E[L(\tau)] &= (0.25 \times 0.5^{(1-1)} \times d(1,2) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(1,3) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(1,4) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(1,5) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(1,6) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(2,3) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(2,4) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(2,5) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(2,6) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(2,1) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(3,4) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(3,5) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(3,6) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(3,1) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(3,2) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(4,5) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(4,6) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(4,1) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(4,2) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(4,3) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(5,6) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(5,1) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(5,2) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(5,3) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(5,4) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(6,1) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(6,2) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(6,3) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(6,4) = 0.031250 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(6,5) = 0.015625 \times 1 = 0.015625) + \\
&= 3.967548. \qquad \text{(B.1)}
\end{aligned}$$

The expected length of the tour $\tilde{\tau} = 1, 5, 4, 3, 2, 6$ is given by:

$$\begin{aligned}
E[L(\tilde{\tau})] &= (0.25 \times 0.5^{(1-1)} \times d(1,5) = 0.25 \times \sqrt{3} = 0.433013) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(1,4) = 0.125 \times 2 = 0.25) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(1,3) = 0.0625 \times \sqrt{3} = 0.108253) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(1,2) = 0.03125 \times 1 = 0.03125) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(1,6) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(5,4) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(5,3) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(5,2) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(5,6) = 0.03125 \times 1 = 0.031250) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(5,1) = 0.015625 \times \sqrt{3} = 0.027063) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(4,3) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(4,2) = 0.125 \times \sqrt{3} = 0.216506) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(4,6) = 0.0625 \times \sqrt{3} = 0.108253) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(4,1) = 0.03125 \times 2 = 0.0625) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(4,5) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(3,2) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(3,6) = 0.125 \times 2 = 0.25) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(3,1) = 0.0625 \times \sqrt{3} = 0.108253) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(3,5) = 0.03125 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(3,4) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(2,6) = 0.25 \times \sqrt{3} = 0.433013) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(2,1) = 0.125 \times 1 = 0.125) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(2,5) = 0.0625 \times 2 = 0.125) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(2,4) = 0.03125 \times \sqrt{3} = 0.054127) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(2,3) = 0.015625 \times 1 = 0.015625) + \\
&\quad (0.25 \times 0.5^{(1-1)} \times d(6,1) = 0.25 \times 1 = 0.25) + \\
&\quad (0.25 \times 0.5^{(2-1)} \times d(6,5) = 0.125 \times 1 = 0.125) + \\
&\quad (0.25 \times 0.5^{(3-1)} \times d(6,4) = 0.0625 \times \sqrt{3} = 0.108253) + \\
&\quad (0.25 \times 0.5^{(4-1)} \times d(6,3) = 0.03125 \times 2 = 0.0625) + \\
&\quad (0.25 \times 0.5^{(5-1)} \times d(6,2) = 0.015625 \times \sqrt{3} = 0.027063) \\
&= 4.144431. \qquad \text{(B.2)}
\end{aligned}$$

Thus, the difference between the expected lengths of the tours is:

$$E[L(\tilde{\tau})] - E[L(\tau)] = 3.967548 - 4.144431 = 0.176883. \tag{B.3}$$

In the following we calculate $\Delta E_{2,5}$ using Bertsimas' recursive equation (3.7), and we show that $\Delta E_{2,5}$ is not equal to (B.3). By symmetry we have that $\forall k \in \{1, ..., n-1\}, A_{i,k} = A_{j,k}, i,\ j \in \{1, ..., n\}$. Furthermore, $\forall k \in \{1, ..., n-1\},\ i \in \{1, ..., n\}$, $B_{i,k} = A_{i,k}$. Thus, $\forall i \in \{1, ..., n\}$, we have:

$$\begin{aligned} A_{i,1} = B_{i,1} &= 1 \times 1 + 1/2 \times \sqrt{3} + 1/4 \times 2 + 1/8 \times \sqrt{3} + 1/16 \times 1 \\ &= 5/8 \times \sqrt{3} + 25/16 \\ &= 1/16(10 \times \sqrt{3} + 25) \\ &= 2.645031755 \end{aligned} \tag{B.4}$$

$$\begin{aligned} A_{i,2} = B_{i,2} &= 1/2 \times \sqrt{3} + 1/4 \times 2 + 1/8 \times \sqrt{3} + 1/16 \times 1 \\ &= 5/8 \times \sqrt{3} + 9/16 \\ &= 1/16(10 \times \sqrt{3} + 9) \\ &= 1.645031755 \end{aligned} \tag{B.5}$$

$$\begin{aligned} A_{i,3} = B_{i,3} &= 1/4 \times 2 + 1/8 \times \sqrt{3} + 1/16 \times 1 \\ &= 1/8 \times \sqrt{3} + 9/16 \\ &= 1/16(2 \times \sqrt{3} + 9) \\ &= 0.77900635 \end{aligned} \tag{B.6}$$

$$\begin{aligned} A_{i,4} = B_{i,4} &= 1/8 \times \sqrt{3} + 1/16 \times 1 \\ &= 1/8 \times \sqrt{3} + 1/16 \\ &= 1/16(2 \times \sqrt{3} + 1) \\ &= 0.27900635 \end{aligned} \tag{B.7}$$

$$\begin{aligned} A_{i,5} = B_{i,5} &= 1/16 \times 1 \\ &= 0.0625. \end{aligned} \tag{B.8}$$

Using these and Bertsimas' recursive equation (3.7), we will now compute $\Delta E_{2,5}$ on the tour $\tau = 1, 2, 3, 4, 5, 6$. That is, the difference in expected length when we move from the tour $\tau = 1, 2, 3, 4, 5, 6$ to the tour $\tilde{\tau} = 1, 5, 4, 3, 2, 6$.

First, we have: $i = 2$, $j = 5$, $k = 3$, $p = 0.5$, and $q = 0.5$. This gives us

$$\begin{aligned}
\Delta E_{2,5} &= \Delta E_{3,5} + (p^2 = 0.25) \times [\\
&\quad (q^{-3} - 1 = 7.0) \times (A_{2,4} = 0.279006) +\\
&\quad (q^3 - 1 = -0.875) \times ((B_{2,1} = 2.645032) - (B_{2,3} = 0.779006)) +\\
&\quad (q^3 - 1 = -0.875) \times ((A_{5,1} = 2.645032) - (A_{5,3} = 0.779006)) +\\
&\quad (q^{-3} - 1 = 7.0) \times (B_{5,4} = 0.279006) +\\
&\quad (q^3 - 1 = -0.875) \times ((A_{2,1} = 2.645032) - (A_{2,3} = 0.779006)) +\\
&\quad (q^{-3} - 1 = 7.0) \times (B_{2,4} = 0.279006) +\\
&\quad (q^{-3} - 1 = 7.0) \times (A_{5,4} = 0.279006) +\\
&\quad (q^3 - 1 = -0.875) \times ((B_{5,1} = 2.645032) - (B_{5,3} = 0.779006))]\\
&= \Delta E_{3,5} + 0.25 \times [\\
&\quad 1.953044 +\\
&\quad -1.632772 +\\
&\quad -1.632772 +\\
&\quad 1.953044 +\\
&\quad -1.632772 +\\
&\quad 1.953044 +\\
&\quad 1.953044 +\\
&\quad -1.632772]\\
&= \Delta E_{3,5} + 0.320272. \qquad \text{(B.9)}
\end{aligned}$$

And we can calculate $\Delta E_{3,5}$ in a similar way. We have $i = 3$, $j = 5$, $k = 2$, $p = 0.5$, and $q = 0.5$. This gives us:

$$\begin{aligned}
\Delta E_{3,5} &= \Delta E_{4,5} + (p^2 = 0.25) \times [\\
&\quad (q^{-2} - 1 = 3) \times (A_{3,3} = 0.779006) + \\
&\quad (q^2 - 1 = -0.75) \times ((B_{3,1} = 2.645032) - (B_{3,4} = 0.279006)) + \\
&\quad (q^2 - 1 = -0.75) \times ((A_{5,1} = 2.645032) - (A_{5,4} = 0.279006)) + \\
&\quad (q^{-2} - 1 = 3) \times (B_{5,3} = 0.779006) + \\
&\quad (q^4 - 1 = -0.9375) \times ((A_{3,1} = 2.645032) - (A_{3,2} = 1.645032)) + \\
&\quad (q^{-4} - 1 = 15) \times (B_{3,5} = 0.0625) + \\
&\quad (q^{-4} - 1 = 15) \times (A_{5,5} = 0.0625) + \\
&\quad (q^4 - 1 = -0.9375) \times ((B_{5,1} = 2.645032) - (B_{5,2} = 1.645032))] \\
&= \Delta E_{4,5} + 0.25 \times [\\
&\quad 2.337019 + \\
&\quad -1.774519 + \\
&\quad -1.774519 + \\
&\quad 2.337019 + \\
&\quad -0.9375 + \\
&\quad 0.9375 + \\
&\quad 0.9375 + \\
&\quad -0.9375] \\
&= \Delta E_{4,5} + 0.28125. \qquad \text{(B.10)}
\end{aligned}$$

And finally, we can calculate $\Delta E_{4,5}$ using equation (3.6) for switching two adjacent nodes with $i = 4$, $j = 5$, $k = 1$, $p = 0.5$, and $q = 0.5$. This gives us:

$$\begin{aligned}
\Delta E_{4,5} &= (p^3 = 0.125) \times [\\
&\quad (A_{4,2} = 1.645032)/(1 - p = 0.5) - \\
&\quad ((B_{4,1} = 2.645032) - (B_{4,5} = 0.0625)) - \\
&\quad (A_{5,1} = 2.645032) - (A_{5,5} = 0.0625)) + \\
&\quad (B_{5,2} = 1.645032)/(1 - p = 0.5)] \\
&= 0.125 \times 1.415064 \\
&= 0.176883. \qquad \text{(B.11)}
\end{aligned}$$

Thus, combining (B.9),(B.10), and (B.11) we have:

$$\begin{aligned}\Delta E_{2,5} &= 0.320272 + 0.28125 + 0.176883 \\ &= 0.778405. \qquad \text{(B.12)}\end{aligned}$$

Thus comparing, (B.12) with (B.3) we have demonstrated the error in Bertsimas' equation (3.7).

Appendix C

Example of a 1-shift move evaluation

As in appendix B, we place the nodes of the graph (customers) on the vertices of a regular hexagon in the Euclidean plane of side 1 so that all distances between pairs of nodes are from the set, $\{1, 2, \sqrt{3}\}$. We set $p = 0.5$.

Let us calculate the difference in expected length of the tours $\tau = 1, 2, 3, 4, 5, 6$ and $\tilde{\tau} = 1, 3, 4, 5, 2, 6$. We shall first do this by calculating, by equation (3.3), the expected length of each and subtracting one from the other. Then we shall compare this result with the calculation using the Bertsimas' recursive equation (3.14). The length of the tour $\tau = 1, 2, 3, 4, 5, 6$ is $E[L(\tau)] = 3.967548$, as calculated by (B.1) in

Appendix B. The length of the tour $\tilde{\tau} = 1, 3, 4, 5, 2, 6$ is given by:

$$
\begin{aligned}
E[L(\tilde{\tau})] \;=\;& (0.25 \times 0.5^{(1-1)} \times d(1,3) = 0.25 \times \sqrt{3} = 0.433013) + \\
& (0.25 \times 0.5^{(2-1)} \times d(1,4) = 0.125 \times 2 = 0.25) + \\
& (0.25 \times 0.5^{(3-1)} \times d(1,5) = 0.0625 \times \sqrt{3} = 0.108253) + \\
& (0.25 \times 0.5^{(4-1)} \times d(1,2) = 0.03125 \times 1 = 0.03125) + \\
& (0.25 \times 0.5^{(5-1)} \times d(1,6) = 0.015625 \times 1 = 0.015625) + \\
& (0.25 \times 0.5^{(1-1)} \times d(3,4) = 0.25 \times 1 = 0.250) + \\
& (0.25 \times 0.5^{(2-1)} \times d(3,5) = 0.125 \times \sqrt{3} = 0.216506) + \\
& (0.25 \times 0.5^{(3-1)} \times d(3,2) = 0.0625 \times 1 = 0.0625) + \\
& (0.25 \times 0.5^{(4-1)} \times d(3,6) = 0.03125 \times 2 = 0.0625) + \\
& (0.25 \times 0.5^{(5-1)} \times d(3,1) = 0.015625 \times \sqrt{3} = 0.027063) + \\
& (0.25 \times 0.5^{(1-1)} \times d(4,5) = 0.25 \times 1 = 0.25) + \\
& (0.25 \times 0.5^{(2-1)} \times d(4,2) = 0.125 \times \sqrt{3} = 0.216506) + \\
& (0.25 \times 0.5^{(3-1)} \times d(4,6) = 0.0625 \times \sqrt{3} = 0.108253) + \\
& (0.25 \times 0.5^{(4-1)} \times d(4,1) = 0.03125 \times 2 = 0.0625) + \\
& (0.25 \times 0.5^{(5-1)} \times d(4,3) = 0.015625 \times 1 = 0.015625) + \\
& (0.25 \times 0.5^{(1-1)} \times d(5,2) = 0.25 \times 2 = 0.5) + \\
& (0.25 \times 0.5^{(2-1)} \times d(5,6) = 0.125 \times 1 = 0.125) + \\
& (0.25 \times 0.5^{(3-1)} \times d(5,1) = 0.0625 \times \sqrt{3} = 0.108253) + \\
& (0.25 \times 0.5^{(4-1)} \times d(5,3) = 0.03125 \times \sqrt{3} = 0.054127) + \\
& (0.25 \times 0.5^{(5-1)} \times d(5,4) = 0.015625 \times 1 = 0.015625) + \\
& (0.25 \times 0.5^{(1-1)} \times d(2,6) = 0.25 \times \sqrt{3} = 0.433013) + \\
& (0.25 \times 0.5^{(2-1)} \times d(2,1) = 0.125 \times 1 = 0.125) + \\
& (0.25 \times 0.5^{(3-1)} \times d(2,3) = 0.0625 \times 1 = 0.0625) + \\
& (0.25 \times 0.5^{(4-1)} \times d(2,4) = 0.03125 \times \sqrt{3} = 0.054127) + \\
& (0.25 \times 0.5^{(5-1)} \times d(2,5) = 0.015625 \times 2 = 0.03125) + \\
& (0.25 \times 0.5^{(1-1)} \times d(6,1) = 0.25 \times 1 = 0.25) + \\
& (0.25 \times 0.5^{(2-1)} \times d(6,3) = 0.125 \times 2 = 0.25) + \\
& (0.25 \times 0.5^{(3-1)} \times d(6,4) = 0.0625 \times \sqrt{3} = 0.108253) + \\
& (0.25 \times 0.5^{(4-1)} \times d(6,5) = 0.03125 \times 1 = 0.03125) + \\
& (0.25 \times 0.5^{(5-1)} \times d(6,2) = 0.015625 \times \sqrt{3} = 0.027063) + \\
=\;& 4.285056. \qquad \text{(C.1)}
\end{aligned}
$$

Thus, the difference between the expected lengths of the tours is:

$$E[L(\tilde{\tau})] - E[L(\tau)] = 4.285056 - 3.967548 = 0.317508. \tag{C.2}$$

In the following we calculate $\Delta E'_{2,5}$ using Bertsimas' recursive equation (3.14). Using equations (B.5)...(B.8) for the values of A and B and Bertsimas' recursive equation (3.14), we will now compute $\Delta E'_{2,5}$ on the tour $\tau = 1, 2, 3, 4, 5, 6$. That is, the difference in expected length when we move from the tour $\tau = 1, 2, 3, 4, 5, 6$ to the tour $\tilde{\tau} = 1, 3, 4, 5, 2, 6$.

First, we have: $i = 2$, $j = 5$, $k = 3$, $p = 0.5$, and $q = 0.5$. This gives us

$$\begin{aligned}
\Delta E'_{2,5} &= \Delta E'_{2,4} + (p^2 = 0.25) \times [\\
&\quad (q^{-3} - q^{-2} = 4) \times (A_{2,4} = 0.279006) +\\
&\quad (q^3 - q^2 = -0.125) \times ((B_{2,1} = 2.645032) - (B_{2,3} = 0.779006)) +\\
&\quad (q - 1 = -0.5) \times ((A_{5,1} = 2.645032) - (A_{5,3} = 0.779006)) +\\
&\quad (q^{-1} - 1 = 1) \times (B_{5,4} = 0.279006) +\\
&\quad (q^3 - q^4 = 0.0625) \times ((A_{2,1} = 2.645032) - (A_{2,3} = 0.779006)) +\\
&\quad (q^{-3} - q^{-4} = -8) \times (B_{2,4} = 0.279006) +\\
&\quad (1 - q^{-1} = -1) \times (A_{5,4} = 0.279006) +\\
&\quad (1 - q = 0.5) \times ((B_{5,1} = 2.645032) - (B_{5,3} = 0.779006))]\\
&= \Delta E'_{2,4} + 0.25 \times [\\
&\quad 1.116025 +\\
&\quad -0.233253 +\\
&\quad -0.933013 +\\
&\quad 0.279006 +\\
&\quad 0.116627 +\\
&\quad -2.232051 +\\
&\quad -0.279006 +\\
&\quad 0.933013]\\
&= \Delta E'_{2,4} - 0.308163. \qquad (C.3)
\end{aligned}$$

And we can calculate $\Delta E'_{2,4}$ in a similar way. We have $i = 2$, $j = 4$, $k = 2$, $p = 0.5$, and $q = 0.5$. This gives us:

$$\begin{aligned}
\Delta E'_{2,4} &= \Delta E'_{2,3} + (p^2 = 0.25) \times [\\
&\quad (q^{-2} - q^{-1} = 2) \times (A_{2,3} = 0.779006) +\\
&\quad (q^2 - q = -0.25) \times ((B_{2,1} = 2.645032) - (B_{2,4} = 0.279006)) +\\
&\quad (q - 1 = -0.5) \times ((A_{4,1} = 2.645032) - (A_{4,4} = 0.279006)) +\\
&\quad (q^{-1} - 1 = 1) \times (B_{4,3} = 0.779006) +\\
&\quad (q^4 - q^5 = 0.03125) \times ((A_{2,1} = 2.645032) - (A_{2,2} = 1.645032)) +\\
&\quad (q^{-4} - q^{-5} = -16) \times (B_{2,5} = 0.0625) +\\
&\quad (1 - q^{-1} = -1) \times (A_{4,5} = 0.0625) +\\
&\quad (1 - q = 0.5) \times ((B_{4,1} = 2.645032) - (B_{4,2} = 1.645032))]\\
&= \Delta E'_{2,3} + 0.25 \times [\\
&\quad 1.558013 +\\
&\quad -0.591506 +\\
&\quad -1.183013 +\\
&\quad 0.779006 +\\
&\quad 0.03125 +\\
&\quad -1 +\\
&\quad -0.0625 +\\
&\quad 0.5]\\
&= \Delta E'_{2,3} + 0.007812. \qquad \text{(C.4)}
\end{aligned}$$

And finally, we can calculate $\Delta E'_{2,3}$ using equation (3.6) for switching two adjacent nodes with $i = 2$, $j = 3$, $k = 1$, $p = 0.5$, and $q = 0.5$. This gives us:

$$\begin{aligned}
\Delta E'_{2,3} &= (p^3 = 0.125) \times [\\
&\quad (A_{2,2} = 1.645032)/(1 - p = 0.5) -\\
&\quad ((B_{2,1} = 2.645032) - (B_{2,5} = 0.0625)) -\\
&\quad (A_{3,1} = 2.645032) - (A_{3,5} = 0.0625)) +\\
&\quad (B_{3,2} = 1.645032)/(1 - p = 0.5)]\\
&= 0.125 \times 1.415064\\
&= 0.176883. \qquad \text{(C.5)}
\end{aligned}$$

Thus, combining (C.3), (C.4), and (C.5) we have:

$$\begin{aligned}\Delta E'_{2,5} &= -0.308163 + 0.007812 + 0.176883 \\ &= -0.123468. \qquad \text{(C.6)}\end{aligned}$$

Thus comparing, (C.6) with (C.2) we have demonstrated the error in Bertsimas' equation (3.14).