Analysis of the Dynamic Travelling Salesman Problem with Different Policies

Santiago Ravassi

A Thesis in

The Department of Mathematics

and Statistics

Presented in Partial Fulfillment of the Requirements for the Degree of Master of Science (Mathematics) at Concordia University

Montreal, Quebec, Canada

December 2011

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Santiago Ravassi

Entitled: Analysis of the Dynamic Travelling Salesman Problem with Different Policies

and submitted in partial fulfilment of the requirements for the degree of

Master of Science (Mathematics)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair

Dr. Yogendra P. Chaubey

_____ Examiner

Dr. Galia Dafni

_____ Examiner

Dr. José Garrido

_____ Supervisor

Dr. Lea Popovic

Approved by _____

Chair of Department or Graduate Program Director

_____

Dean of Faculty

Date _____

Abstract

Analysis of the Dynamic Travelling Salesman Problem with Different Policies

Santiago Ravassi

We propose and analyze new policies for the travelling salesman problem in a dynamic and stochastic environment (DTSP). The DTSP is defined as follows: demands for service arrive in time according to a Poisson process, are independent and uniformly distributed in a Euclidean region of bounded area, and the time service is zero; the objective is to reduce the time the server takes to visit to all the present demands for the first time. We start by analyzing the nearest neighbour (NN) policy since it has the best performance for the dynamic vehicle routing problem (DTRP), a closely related problem to the DTSP. We next introduce the random start policy whose efficiency is similar to that of the NN, and we observe that when the random start policy is delayed, it behaves like the DTRP with the NN policy. Finally, we introduce the partitioning policy, and show that, relative to other policies, it reduces the expected time that demands are swept from the region for the first time.

Acknowledgements

I would like to express my gratitude first and foremost to my supervisor, Professor Lea Popovic. She has been an extremely patient and enthusiastic supervisor who has helped me academically and personally during my master studies. I feel very fortunate and honoured to have her as my supervisor.

My special thanks to Professor Galia Dafni for her generosity and support in difficult moments.

I would also like to thank the staff of the Department of Mathematics and Statistics for the great help they provided to see this thesis done.

My thanks to my friends Nadia, Laura, Federico, Emmanuelle, Darío, Debora, Felipe, Mohamad, and Laura (F).

I want to thank my mother and father for supporting me in all my pursuits , and my sister for her great friendship. No matter where I live, they are always present.

Finally, I would like to thank Lívia for her love, energy, and companionship. She is my light, and to her I dedicate this thesis.

To Lívia,

Contents

List of Figures

List of Tables

Introduction

Given a collection of demands and the cost of travel between each pair of them, the objective of the of the traveling salesman problem (TSP) is to find the cheapest way of visiting all the demands. The DTSP is the stochastic version of the TSP. It was introduced by Psaraftis in 1988 [32] in a non spatial setting, and focuses in finding a policy that allows a single server to visit demands whose positions are known and independently generated. The service request is randomly assigned according to a Poisson process and uniformly assigned across demands, and the service time of demands are randomly and independently assigned according to some distribution with known mean and finite variance. In 1991 Bertsimas and van Ryzin [3] introduced the DTRP, a related problem to the DTSP where demands are uniformly located in a region of finite area and studied different policies that can be used depending on both the rate at which demands are created and the mean service time of the demands.

We have slightly departed from the definition of the DTSP given by Psaraftis since demands are generated according to a Poisson process but independently and uniformly placed over a bounded region in the Euclidean plane. We assume that once the server visits a demand it is immediately freed to visit a new one, that is, the service time is assumed to be zero for all the demands. In our study, when we talk about the DTSP, we will refer to our definition of the problem; otherwise, it will be stated.

In Chapter 1, we discuss randomized algorithms and probabilistic algorithms and

why it is sometimes convenient to use them instead of deterministic algorithms.

In Chapter 2, we propose some probabilistic concepts that might be useful in the development of the DTSP. We start by defining random variables and stochastic processes, then introduce Markov chains. The definition of discrete, continuous time and inhomogeneous Markov chains, as well as piecewise deterministic continuous-time Markov chains will be introduced along with their most relevant properties.

In Chapter 3, we examine the results obtained by Bertsimas and Van Ryzin in 1991 on the DTRP under different policies and with different arrival rates and service time of demands. We later discuss some common points between the DTSP and the DTRP, and why some of their results can be used in our problem. Then, we estimate an upper bound for the expected distance between demands in the DTSP with the NN policy. Finally, we observed that there exist similarities in the upper bound assumed for the mean distance between demands in the DTRP and the upper bound assumed for the mean distance between demands in our problem when both problems are under the NN policy.

In Chapter 4, we analyze the NN policy on the DTSP since the NN was considered to be optimal in the DTRP under heavy traffic. We estimate the mean number of unattended demands when the system stabilizes and the mean time at which this happens using different algorithms. Based on the NN policy, we introduce the random start policy, which has the same performance as the NN. A variation of the random start policy is presented; under certain conditions, its performance is equivalent to that of the DTRP. Finally, we introduce the partitioning policy which is shown to reduce the expected time during which the region has no demands for the first time, with respect to the NN policy.

Chapter 1

Randomized Algorithms and Probabilistic Analysis of Algorithms

Randomized algorithms make random choices during their execution. In practice, a randomized program would use values generated by a random number generator to guide its execution, in the hope of achieving good average performance over all the possible random choices. There are two principal advantages to randomized algorithms. First, in many situations randomized algorithms run faster than the best known deterministic algorithms. Second, randomized algorithms are simpler to describe and implement than deterministic algorithms of comparable performance. However, these gains come at a price; the answer may have some probability of being incorrect, or efficiency is guaranteed only with some probability. Though it may seem unusual to design an algorithm that may produce incorrect results or run inefficiently, if the probability of such consequence is sufficiently small, the improvement in speed or memory requirements may well be worthwhile.

An example of a randomized algorithm is the randomized quicksort algorithm,

one of the fastest sorting algorithms in practice. Quicksort is a divide and conquer algorithm to sorting items on a list; it splits the list into two new lists and then recursively call the quicksort procedure to sort them. If we want to have an effective strategy, the split phase must ensure that one of the new sub list is neither larger nor smaller than the other by a proportion of $\frac{3}{4}$, see [27], so a random choice of the split point will effectively divide the partition half the time. The expected running time for the randomized quicksort is of $\mathcal{O}(n \log n)$; moreover, with high probability the quicksort algorithm runs in time $\mathcal{O}(n \log n)$, see [27]. Two of the most well known deterministic sorting algorithms are bubble sort and heapsort. Bubble sort is known to be simple to implement; however, it has poor performance since worst and average case performance of the algorithm is $\mathcal{O}(n^2)$, see [18]. On the other hand, heapsort has worst and average case performance of $\mathcal{O}(n \log n)$, but the approximate average expected running time of heapsort is between 2 to 4/3 times larger than quicksort, depending of the size of the list to be sorted, see [18].

The performance of randomized algorithms depends on the random decisions and not on the assumptions about the inputs. On the other hand, in the probabilistic analysis of the algorithm, the distribution of the input is assumed to be random, and the algorithm -that might be deterministic- is analyzed.

Complexity theory tries to classify computation problems according to their computational complexity, in particular distinguishing between easy and hard problems. A method to estimate the computational complexity of an algorithm is the probabilistic analysis of algorithms, which is used to study how algorithms perform when the input is taken from a well-defined probabilistic space. For the classical worst-case complexity theory the travelling salesman problem is NP-hard though they are often easy to solve in practice. Probabilistic analysis of algorithms gives a theoretical explanation for this phenomenon. NP-hard problems might have algorithms that are extremely efficient on almost all inputs, see [26]; that is these problems are hard to

solve when the input is some pathological set, but in real-life situations the problems are not hard to solve. In other words, if the input is randomly generated according to some probability distribution on the collection of all possible inputs, it is very likely that the problem instance is easy to solve, whereas instances that are hard to solve appear with relatively small frequency. We first perform a probabilistic analysis for the different policies used in the DTSP.

Chapter 2

Probabilistic Tools

We review the theory of Markov processes and martingales, but first we need to explain what a stochastic process is. Most of the theory in this chapter was taken from [10, 11, 21, 22, 26, 29, 37].

2.1 Introduction

Any experiment that involves randomness can be modeled by a probability space. Such space comprises of a set of outcomes Ω, a collection of events $\mathscr{F}$, and a probability measure P.

Definition 2.1.1 *The complete list of all the possible outcomes of an experiment is called the sample space and is denoted by* Ω.

An event is a subset (collection) of Ω, but not all the subsets of Ω are events. Thus, we define the collection of events $\mathscr{F}$ from the collection of subsets of Ω. Since we are interested in combining events, we want our collection of events to include these combination of events. Thus, we need to ensure certain properties to $\mathscr{F}$.

Definition 2.1.2 *The collection* $\mathscr{F}$ *of subsets of* Ω *is called a* σ*-field if it satisfies the following conditions:*

1. *if* $A_1, A_2, \ldots \in \mathscr{F}$ *then* $\cup_{i=1}^{\infty} A_i \in \mathscr{F}$

2. *if* $A \in \mathscr{F}$ *then* $A^c \in \mathscr{F}$.

With any experiment we may associate a pair $(\Omega, \mathscr{F})$, where Ω is the set of possible outcomes and $\mathscr{F}$ is a σ-field of subsets of Ω which contains all the events whose occurrence we may be interested in; thus, to call a set A an event is equivalent to say that A belongs to the σ-field in question.

Example 2.1.1 *A coin is tossed twice, then* $\Omega = \{\{HH\}, \{TT\}, \{HT\}, \{TH\}\}$. *Then, two* $\sigma-$*fields of* Ω *are the collection of sets*

$$\mathscr{F}_1 = \{\varnothing, \Omega, \{HH, TT\}, \{HT, TH\}\} \text{ and } \mathscr{F}_2 = \{\varnothing, \Omega\}.$$

We are also interested in assigning likelihoods to the events in $\mathscr{F}$. The probability function P that assigns likelihoods to the members of $\mathscr{F}$ is called a probability measure.

Definition 2.1.3 *A probability measure* P *on* $(\Omega, \mathscr{F})$ *is a function* $P : \mathscr{F} \to [0, 1]$ *satisfying*

1. $P(\varnothing) = 0$, $P(\Omega) = 1$

2. *if* $A_1, A_2, \ldots$ *is a collection of disjoints members of* $\mathscr{F}$, *in that* $A_i \cap A_j = \emptyset$ *for all pairs* $i \neq j$, *then*
$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

The triple $(\Omega, \mathscr{F}, P)$, *comprising a set* Ω, *a* σ*-field* $\mathscr{F}$ *of subsets of* Ω, *and a probability measure* P *on* $(\Omega, \mathscr{F})$, *is called a probability space.*

The probability function tells us how the probability is distributed over the set of events $\mathscr{F}$. A probability measure is a special case of a measure on the pair $(\Omega, \mathscr{F})$.

A measure is a function $\mu : \mathscr{F} \to [0, \infty)$ satisfying $\mu(\varnothing) = 0$ together with Definition 2.1.3-2. A measure μ is a probability measure if $\mu(\Omega) = 1$, that is, a probability measure must assign 1 to the entire probability space.

Definition 2.1.4 *A random variable is a function $X : \Omega \to \mathbb{R}$ with the property that $\{\omega \in \Omega : X(\omega) \leq x\} \in \mathscr{F}$ for each $x \in \mathbb{R}$. Such a function is said to be $\mathscr{F}$-measurable.*

Definition 2.1.5 *A stochastic process $X = \{X_t : t \in T\}$ is a collection of random variables that map the sample space Ω into some set S. The index t often represents time, and in that case the process X models the value of a random variable X that changes over time. We call X_t the state of the process at time t.*

The mathematical analysis of random processes varies greatly depending on weather S and T are finite, countable or uncountable.

2.2 Markov Chains

The Markov process is a special stochastic process that retains no memory of where it has been in the past. This means that only the current state of the process can influence where it goes next. The set of possible values the process can take will be denoted by M. The set M might be finite or countable, and it is called the state space. For any states $i, j \in M$, if $X_t = i$, then the process is said to be in state i at time t. We suppose that whenever the process is in state i, there is a fixed probability $P_{i,j}$ that it will next be in state j.

Definition 2.2.1 *A discrete-time stochastic process $X = \{X_0, X_1, X_2, \ldots\}$ is a Markov chain if*

$$P(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}, \ldots, X_0 = i_0) = P(X_t = i_t | X_{t-1} = i_{t-1})$$

for all states i_t with $t \geq 0$.

This definition expresses that the state X_t depends on the previous state X_{t-1} but is independent of how the process arrived to state X_{t-1}. This is called the Markov Property or memoryless property, and it is what we mean when we say that a chain is Markovian. It is important to note that the Markov property does not imply that X_t is independent of the random variables $X_0, X_1, \ldots, X_{t-2}$; it just implies that any dependency of X_t on the past is captured in the value X_{t-1}.

If for all t X_t assumes values in a finite set, then we say that X is a finite state space process or finite Markov chain, and if X_t assumes values from countable infinite set, then X is a discrete state process or Markov chain. If T is countable infinite set, we say that X is a discrete-time process. A Markov chain is said to be time homogeneous, if $P(X_t = i|X_{t-1} = j)$ is independent of t.

We will first consider discrete-time homogeneous Markov chains, and then we will introduce continuous-time and inhomogeneous continuous time Markov chains.

The Markov property implies that the Markov chain is uniquely defined by the one-step transition matrix,

$$\mathbf{P} = (P_{i,j}).$$

That is, the entry in the ith row and jth column is the transition probability $P_{i,j}$. It follows that, for all i,

$$\sum_{j \geq 0} P_{i,j} = 1.$$

This transition matrix representation of a Markov chain is convenient for computing the distribution of future states process. Let $\underline{p}(t) = (p_0(t), p_1(t), p_2(t), \ldots)$ be the vector giving the distribution of the state of the chain at time t. Summing over all possible states at time $t-1$, we have

$$p_i(t) = \sum_{j \geq 0} p_j(t-1) P_{j,i}$$

or

$$\underline{p}(t) = \underline{p}(t-1)\mathbf{P}.$$

Thus, for $t, s \geq 0$,

$$\underline{p}(t+s) = \underline{p}(t)\mathbf{P}^s,$$

where $\mathbf{P}^m$ is the matrix whose entries are the m-step transition probabilities, so the probability that the chain moves from state i to state j in exactly m steps is $P^s_{i,j} = P(X_{t+s} = j | X_t = i)$.

When the representation of a Markov chain is through a directed weighted graph $D = (V, E, w)$, the set of the vertices V of the graph is the set of states of the chain. There is a directed edge $(i, j) \in E$ if and only if $P_{i,j} > 0$, in which case the weight $w(i, j)$, of the edge (i, j), is given by $w(i, j) = P_{i,j}$. Self-loops are allowed, and, for each $i \in V$, it is required that $\sum_{j:(i,j)\in E} w(i, j) = 1$.

Classification of States

A first step in analyzing the long-term behavior of a Markov chain is to classify its states. In the case of a finite Markov chain, this is equivalent to analyzing the connectivity structure of the directed graphs.

Definition 2.2.2 *A state i communicates with state j if*

$$P(X_t = j \text{ for some } t \geq 0 | X_0 = i) > 0 \text{ and } P(X_t = i \text{ for some } n \geq 0 | X_0 = j) > 0.$$

Definition 2.2.3 *A set of states C is a communicating class if every pair of states in C communicates with each other, and no state in C communicates with any state not in C.*

Definition 2.2.4 *A Markov chain is irreducible if, for every pair of states, there is a nonzero probability that the first state can reach the second.*

Lemma 2.2.1 *A finite Markov chain is irreducible if and only if its graph representation is a strongly connected graph.*

Definition 2.2.5 *A state i is recurrent if*

$$P(X_t = i \ \ \textit{for infinitely many} \ \ t \geq 1 | X_0 = i) = 1,$$

and it is transient if

$$P(X_t = i \ \ \textit{for infinitely many} \ \ t \geq 1 | X_0 = i) = 0.$$

Another important result that help to classify of the states is Proposition 2.2.1. Let $r_{i,j}^t$ denote the probability that, starting at state j, the first transition to state i occurs at time t; that is,

$$r_{j,i}^t = P(X_t = i, X_s \neq i \text{ for } 1 \leq s \leq t-1 | X_0 = j).$$

Proposition 2.2.1 *A state i is recurrent if*

$$\sum_{t \geq 1} r_{i,i}^t = 1,$$

and it is transient if

$$\sum_{t \geq 1} r_{i,i}^t < 1.$$

The total number of visits and the first passage time to a state also helps to classify a state. The random variable R_i denotes the total number of visits to state i,

$$R_i = \sum_{t=0}^{\infty} I\{X_t = i\},$$

and T_i defines the first passage time to state i,

$$T_i = \min\{t \geq 1 | X_t = i\},$$

with the convention that $T_i = \infty$ if this visit never occurs. Then, $r_{j,i}^t = P(T_i = t | X_0 = j)$, and a state i is recurrent if and only if $P(T_i < \infty) = 1$, that is if the visit to state i occurs with probability 1. On the other hand, a state i is transient if and only if $P(T_i = \infty) > 0$, that is if there is a chance the visit to state i never occurs.

Proposition 2.2.2 *The following three events are equivalent,*

1. $P(T_i < \infty) = 1$,
2. $P(R_i = \infty) = 1$,
3. $E(R_i) = \infty$,

and the following three events are also equivalent,

1. $P(T_i = \infty) > 0$,
2. $P(R_i < \infty) = 1$,
3. $E(R_i) < \infty$.

Proposition 2.2.2 relates the first passage time, the total number of visits, and Definition 2.2.5. A more general result is given by Proposition 2.2.3.

Proposition 2.2.3 *Given $X_0 = i$ state i,*

$$P(R_i = k) = P(T_i < \infty)^{k-1} P(T_i = \infty).$$

Proposition 2.2.4 *The states of an irreducible Markov chain are either all recurrent or all transient.*

The expected time to first reach state j when the chain starts at state i is given by

$$h_{i,j} = \sum_{t \geq 1} t \; r_{i,j}^t.$$

Definition 2.2.6 *A recurrent state i is positive recurrent if $h_{i,i} = E_i(T_i) < \infty$; otherwise, it is null recurrent.*

Lemma 2.2.2 *In a finite Markov chain:*

1. *at least one state is recurrent*

2. *all recurrent state are positive recurrent*

Thus, for a null recurrent state to exist the Markov chain must have an infinite number of states.

Proposition 2.2.5 *In an irreducible Markov chain, if $h_{i,i} < \infty$ for some $i \in M$, then $h_{i,j} < \infty$ for all $i, j \in M$.*

Hence, we can therefore classify an irreducible chain as positive recurrent if one state and hence all states are positive recurrent. From Propositions 2.2.4 and 2.2.5, we have that the states of an irreducible Markov chain are either all transient, all null recurrent, or all positive recurrent.

Proposition 2.2.6 *A recurrent state i is null recurrent if*

$$\lim_{t \to \infty} P_{i,i}^t = 0.$$

Otherwise, it is positive recurrent.

Definition 2.2.7 *A state j in a discrete-time Markov chain is periodic if there exists an integer $\Delta > 1$ such that $P(X_{t+s} = j | X_t = j) = 0$ unless s is divisible by Δ. A discrete-time Markov chain is periodic if any state in the chain is periodic. A chain that is not periodic is aperiodic.*

Definition 2.2.8 *An aperiodic and positive recurrent state is an ergodic state. A Markov chain is ergodic if all its states are ergodic.*

Corollary 2.2.1 *An aperiodic, finite, and irreducible Markov chain is an ergodic chain.*

Ergodic theorems concern the limiting behavior of averages over time, and, in the case of Markov chains, the long-run proportion of time spent in each state. For a Markov chain to be ergodic, two conditions are required in all the states: aperiodicity and positive recurrence. Aperiodicity ensures that the limiting probability that the chain is in any state is independent of the initial state. Positive recurrence, makes sure that the expected time any state waits to be visited is finite when the chain is irreducible, as stated Proposition 2.2.5; in addition, positive recurrence guarantees, together with aperiodicity, that $P_{i,j}^t$ converges to a positive limit.

Stationary Distributions

If $\mathbf{P}$ is the one-step transition probability matrix of a Markov chain and if $\underline{p}(t)$ is the probability distribution of the state of the chain at time t, then $\underline{p}(t+1) = \underline{p}(t)\mathbf{P}$. Of particular interest are state probability distributions that do not change after a transition.

Definition 2.2.9 *A stationary or invariant distribution of a Markov chain is a probability distribution $\underline{\pi}$ such that*

$$\underline{\pi} = \underline{\pi}\boldsymbol{P}.$$

If a chain ever reaches a stationary distribution then it maintains that distribution for all future time, and thus a stationary distribution represents a steady state or an equilibrium in the chain's behavior. The fundamental theorem of Markov chains characterizes chains that converge to stationary distributions.

Theorem 2.2.1 (Ergodic Theorem) *Any irreducible ergodic Markov chain has the following properties:*

1. *the chain has a unique stationary distribution* $\underline{\pi}$,
2. *for all* j *and* i, *the* $\lim_{t\to\infty} P^t_{j,i}$ *exists, and it is independent of* j,
3. $\pi_i = \lim_{t\to\infty} P^t_{j,i} = \dfrac{1}{h_{i,i}}$.

From Theorem 2.2.1, we can make some observations. If the time is sufficiently large, the probability that the chain is in state i is π_i and is independent of the initial state. If the average time to return to state i from i is $h_{i,i}$, then we expect to be in state i for $\frac{1}{h_{i,i}}$ of the time; thus, in the long-run, we must have $\pi_i = \frac{1}{h_{i,i}}$. Note, that a Markov chain does not have to be aperiodic to have a unique stationary distribution; if i is a state of a periodic Markov chain, then the stationary distribution π_i is not the limiting probability of the chain being in state i but the long term frequency of visiting state i.

2.3 Discrete-Time Martingales

A martingale is a stochastic process whose average value remains constant in a particular strong sense. We will define discrete time martingales as they are used in Section 2.6.1; continuous time martingales will be used in Section 2.8 and will be defined later in Section 2.7.

Suppose that $\Im = \{\mathscr{F}_t : t \geq 0\}$ is a sequence of sub-σ-fields of $\mathscr{F}$, then we say that $\Im$ is a filtration if $\mathscr{F}_t \subseteq \mathscr{F}_s$ for all $t \leq s$. We say a sequence $X = \{X_t : t \geq 0\}$ is adapted to a filtration $\Im$ if X_t is $\mathscr{F}_t$ measurable for all t. In other words, if we know $\mathscr{F}_t$, we can discern the value of X_t and, more generally, the values of X_s for all $s \leq t$.

Definition 2.3.1 *Let $\Im$ be a filtration of the probability space $(\Omega, \mathscr{F}, P)$, and let $\{X_0, X_1, \ldots\}$ be a sequence of random variables which is adapted to $\Im$. We call the pair $(X, \mathscr{F}) = \{(X_t, \mathscr{F}_t) : t \geq 0\}$ a discrete-time martingale if, for all $t \geq 0$,*

1. $E|X_t| < \infty$,

2. $E(X_{t+1}|\mathscr{F}_t) = X_t$.

From this definition, we can think of $\mathscr{F}_t$ as the state of knowledge or history of the process X up to time t, or more precisely as a σ-field with respect to which each of the variables $X_0, X_1, \ldots, X_t$ is measurable.

Definition 2.3.2 *If conditions 1 and 2 from Definition 2.3.1 are replaced by,*

1. $E(X_t^+) < \infty$,

2. $E(X_{t+1}|\mathscr{F}_t) \geq X_t$.

then the pair $(X, \mathscr{F})$ is called a discrete-time submartingale. If they are replaced by,

1. $E(X_t^-) < \infty$

2. $E(X_{t+1}|\mathscr{F}_t) \leq X_t$.

then then the pair $(X, \mathscr{F})$ is called a discrete-time supermartingale.

Consider the notation $a^+ = \max\{a, 0\}$ and $a^- = min\{a, 0\}$. Since $a = a^+ - a^-$ and $|a| = a^+ + a^-$, the conditions in Definition 2.3.2 are weaker than in Definition 2.3.1. Note that a process is both a martingale and a submartinagale if and only if it is a martingale.

Definition 2.3.3 *A random variable $T : \Omega \to \{0, 1, \ldots\} \cup \{\infty\}$ is called a stopping time with respect to a filtration $\Im$ if $\{T = t\} \in \mathscr{F}_t$ for all $t \geq 0$.*

2.4 Discrete-Time Homogeneous Countable Markov Chains

Given a Markov chain, we would like to obtain information on its stationary distribution, if it exists. This is simple for finite-state Markov chains where the stationary distribution can be computed exactly if it exists. However, the problem is highly non trivial when the state space is countable where, in addition, countable state Markov chains require further analysis in the properties of the stationary distribution since one needs to establish its existence.

Discrete-time homogeneous countable Markov chains have already been defined in Definition 2.2.1. For a description of irreducibility, recurrence and transience, and positive recurrence and null recurrence of Markov chains see Definitions 2.2.4, 2.2.5, and 2.2.6 respectively. Note that Lemma 2.2.2 states that an irreducible Markov chain defined in a finite state space is always recurrent. However, in a countable space, it might be either positive recurrent, null recurrent, or transient.

Classification of chains

The following examples show two Markov chains defined in a countable state space. While the first example describes a recurrent Markov chain, the second one describes a transient Markov chain though both are irreducible.

Example 2.4.1 *Let $\{b_t\}_{t\geq 1}$ be an independent and identically distributed sequence of Bernoulli random variables: $P(b_t = 1) = P(b_t = -1) = \frac{1}{2}$ for all t and $h : \mathbb{Z} \to \mathbb{Z}$.*

The Markov chain

$$X_{t+1} = X_t + h(X_t) + b_{t+1} \quad \text{for } t = 0, 1, 2 \ldots$$

is recurrent if h satisfies:

1. *$|h(x)| < |x|$ for $x \neq 0$,*

2. $h(x) < 0$ *if* $x > 0$, *and*

3. $h(x) > 0$ *if* $x < 0$.

The function h defined in Example 2.4.1, ensures that the process X is pushed toward the state 0; thus, by intuition, we can expect X to be recurrent. The opposite occurs in Example 2.4.2 where h forces the process to "spread out".

Example 2.4.2 *Based on Example 2.4.1, if we redefine* h, *so that*

1. $h(x) > 0$ *if* $x > 0$, *and*

2. $h(x) < 0$ *if* $x < 0$,

the Markov chain is transient.

Proposition 2.4.1 *If the class* C *is recurrent, then for all* $i \in C$

$$\sum_{j \in C} p_{i,j} = 1.$$

Proposition 2.4.2 *Given a communicating class* C, *if for some* $i \in C$

$$\sum_{j \in C} p_{i,j} < 1.$$

then the class is transient.

Note that if the chain is defined on a countable state space, Proposition 2.4.2 is only a sufficient but not a necessary condition.

Given a Markov chain X, and a fixed state j, for each state $i \in M$, let

$$\alpha(i) = P(X_t = j \text{ for some } t \geq 0 | X_0 = i).$$

Proposition 2.4.3 *Suppose X is irreducible. If X is transient, then there is a unique solution to the equations:*

1. $0 \leq \alpha(i) \leq 1$,

2. $\alpha(j) = 1$, $\inf\{\alpha(i) : i \in M\} = 0$,

3. $\alpha(i) = \sum_{k \in M} p(i,k)\alpha(k), \quad i \neq j$,

that must correspond to the appropriate probability. Moreover, if X is recurrent there is no solution.

That is, an irreducible Markov chain is transient if and only if for any j we can find a function $\alpha(i)$ satisfying the equations in Proposition 2.4.3.

Given an irreducible and aperiodic Markov chain. If the state space is finite, the chain would be recurrent with a unique stationary distribution by Corollary 2.2.1 and Theorem 2.2.1. However, in a countable state space a recurrent Markov chain might be positive recurrent or null recurrent. Only when it is positive recurrent, the chain might have a unique stationary distribution.

Let f be a function that takes values on the elements of M. For $i \in M$,

$$Pf(i) = \sum_{j \in M} p_{j,i} f(i) = E_i\big[f(X_1)\big].$$

That is, if the current state is i, $Pf(i)$ gives the expected value of the function f at the next step.

The following lemma is used to prove the Foster-Lyapunov criterion [7, 25], which will be used to determine the recurrence (or transience) of the Markov chains in Examples 2.4.1 and 2.4.2.

Lemma 2.4.1 *Let X be a Markov chain on a countable state space M, and $f : M \to [0, \infty)$ satisfy $Pf(i) \leq f(i)$ for all $i \in M \setminus F$ where $F \subset M$. Then the stopped process*

$\left\{f(X_{t\wedge D})\right\}_{t\geq 0}$ *is a supermartingale. Similarly, if* $Pf(i) = f(i)$*, then the process is a martingale.*

A function $f : M \to [0, \infty)$ is compact if for each $c \in [0, \infty)$ the set $\{i \in M : f(i) \leq c\}$ is finite.

Theorem 2.4.1 (Foster-Lyapunov Criterion) *Let* X *be an irreducible Markov chain. Suppose there is a finite set* $F \subset M$ *and a compact function* f *such that*

1. $Pf(i) \leq f(i)$ *for all* $i \notin F$*,*
2. $\{i \in S : f(i) \leq M\}$ *is a finite set for each* $M > 0$*.*

Then X *is recurrent.*

Theorem 2.4.2 *Assume* X *is irreducible. Suppose there is a finite set* F *and a function* $g : M \to [0, \infty)$ *such that*

1. $Pg(i) \leq g(i)$ *for all* $i \in F$
2. $\inf\{g(i) : i \in M\} = 0$*.*

Then X *is transient.*

Using Theorem 2.4.1 and choosing $f(x) = |x|$, it can be shown that the Markov chain in Example 2.4.1 is recurrent. Similarly, by Theorem 2.4.2 and choosing $g(x) = \frac{1}{|x|}$, it can be shown that the Markov chain in Examples 2.4.2 is transient.

Stationary Distributions

Theorem 2.4.3 *If a Markov chain is irreducible, aperiodic and positive recurrent, then*

1. *It has a unique limiting distribution such that for all* i, j*,*

$$\lim_{t\to\infty} p^t(i, j) = \pi_j > 0.$$

2. *The limiting distribution* $\underline{\pi}$ *satisfies:* $\sum_{i \in M} \pi_i = 1$ *and* $\pi_j = \sum_{i \in M} \pi_i p(i,j)$.

Proposition 2.4.4 *Let* X *be an irreducible and aperiodic Markov chain and assume that* $X_0 = i$. *If* X *is positive recurrent, then*

$$h_{i,i} = \frac{1}{\pi_i}.$$

If X *is null recurrent or transient, then*

$$h_{i,i} = \infty.$$

Theorem 2.4.3 and Proposition 2.4.4 provide the same results as Theorem 2.2.1. However, the use of Theorem 2.4.3 requires to know that the Markov chain is positive recurrent, a property that might not be trivial to verify in the countable case.

2.5 The Poisson Process

A continuous-time stochastic process $\{N(t) : t \geq 0\}$ is said to be a a counting process if $N(t)$ represents the total number of 'arrivals' or 'events' that occur by time t. Each realization of the process N is a non-decreasing step function $N : t \to \mathbb{N}_0$. The Poisson process is a stochastic counting process that is related to both the uniform and the exponential distribution.

Definition 2.5.1 *A Poisson process with parameter* λ *is a stochastic counting process* $\{N(t), t \geq 0\}$ *such that the following statements hold.*

1. $N(0) = 0$

2. *The process has independent and stationary increments. That is, for any* $t, s > 0$, *the distribution of* $N(t+s) - N(s)$ *is identical to the distribution* $N(t)$, *and for*

any two disjoints intervals $[t_1, t_2]$ *and* $[t_3, t_4]$*, the distribution of* $N(t_2) - N(t_1)$ *is independent of the distribution* $N(t_4) - N(t_3)$*.*

3. $lim_{t\to 0}\frac{P(N(t)=1)}{t} = \lambda$*. That is, the probability of a single event in a short interval is* λt*.*

4. $lim_{t\to 0}\frac{P(N(t)\geq 2)}{t} = 0$*. That is, the probability of more than one event is a short interval* t *ends to zero.*

Theorem 2.5.1 *Let* $\{N(t) : t \geq 0\}$ *be a Poisson process with parameter* λ*. For any* $t, s \geq 0$ *and any integer* $n \geq 0$*,*

$$P_n(t) = P(N(t+s) - N(s) = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}.$$

The parameter λ is also called the rate of the Poisson process since the number of events during any period of length t is a Poisson random variable with expectation λt. The reverse is also true, that is, we could have defined the Poisson process as a process with Poisson arrivals, as follows.

Theorem 2.5.2 *Let* $\{N(t) : t \geq 0\}$ *be a stochastic process such that:*

1. $N(0) = 0$

2. *the process has independent increments. That is, the number of events in disjoint time intervals are independent from each other.*

3. *the number of events in an interval of length* t *has a Poisson distribution with mean* λt*.*

Then $\{N(t) : t \geq 0\}$ *is a Poisson process with rate* λ*.*

Theorem 2.5.3 *Given that* $N(t) = n$*, then the* n *arrival times have the same distribution as the order statistics of* n *independent random variables uniformly distributed over* $[0, t]$*.*

This result states that, under the condition that the n events have occurred in $[0, t]$, the times at which the events occur, considered as unordered random variables, are distributed independently and uniformly in the interval $[0, t]$.

2.6 Continuous-Time Homogeneous Markov Chains

In a countable space, the continuous-time homogeneous Markov chain is the continuous time analogue of the homogeneous Markov chain, were the process spent a random interval of time in a state before moving to the next state.

Definition 2.6.1 *A continuous-time random process $X = \{X_t : t \geq 0\}$ is a continuous-time homogeneous Markov chain if, for all $s, t \geq 0$,*

$$P(X_{s+t} = i \mid X_u, 0 \leq u \leq t) = P(X_{s+t} = i \mid X_t),$$

and this probability is independent of the time t.

As in the discrete case, this definition says that the distribution of the state of the system at time X_{s+t}, conditioned on the history up to time t, depends only on state X_t and is independent of the particular history that lead the process to state X_t.

A continuous-time homogeneous Markov chain can be expressed as a combination of two random processes as follows:

1. A transition matrix $\mathbf{P} = (p_{i,j})$ where $p_{i,j}$ is the probability that the next state is j, given that the current state is i.

2. A vector of parameters $\theta_1, \theta_2, \ldots$ such that the distribution of the time that the process spends in state i before moving to the next step is exponential with parameter θ_i. The distribution of time spent at a given state must be exponential in order to satisfy the memoryless requirement of the Markov process.

In other words, the continuous-time homogeneous Markov chain is a stochastic process that moves from state to state in accordance with a Markov chain, but is such that the amount of time it spends in each state, before proceeding to the next state, is exponentially distributed. Note that the Poisson process is a Markov process having states $0, 1, 2, \ldots$ that always proceeds from state k to state $k+1$, where $k \geq 0$ and the parameters $\theta_1, \theta_2, \ldots$ are all equal to 1.

Assuming a stationary distribution π exists, then the probability π that the HCTMC will be in state i infinitely far out in the future is

$$\pi_i \theta_i = \sum_k \pi_k \theta_k p_{k,i},$$

regardless of its initial state.

We will introduce the basic properties of Q-matrices and explain their connection with continuous-time Markov chains. This new approach provides a more direct mathematical description and makes possible a number of constructive realizations of a given Markov chain. Theorem 2.6.1 will provide an alternative definition of continuous-time Markov chains related to the one we just introduced.

Definition 2.6.2 *A Q-matrix on M is a matrix $Q = \{q_{i,j} \in M\}$ that satisfies the following properties*

1. $0 \leq -q_{i,i} < \infty$,
2. $q_{i,j} \geq 0$ *for all* $i \neq j$,
3. $\sum_{j \in m} q_{i,j} = 0$ *for all* i.

Some additional definitions are needed before for Theorem 2.6.1.

Definition 2.6.3 *A jump matrix $\Pi = (\pi_{i,j} : i, j \in M)$ of Q is defined by*

$$\pi_{i,j} = \begin{cases} \frac{q_{i,j}}{q_i} & \text{if } j \neq i \text{ and } q_i \neq 0 \\ 0 & \text{if } j \neq i \text{ and } q_i = 0 \end{cases}$$

$$\pi_{i,i} = \begin{cases} 0 & \text{if } q_i \neq 0 \\ 1 & \text{if } q_i = 0, \end{cases}$$

where $q_i = q(i) = -q_{i,i}$.

A jump process is a right-continuous stochastic process with piecewise constant sample paths.

Theorem 2.6.1 *Let X be a minimal jump process with values in M. Let Q be a Q-matrix on M with jump matrix Π and semigroup $(P(t) : t \geq 0)$. Then the following two conditions are equivalent:*

1. *The jump chain $(Y_n)_{n\geq 0}$ of $(X)_{t\geq 0}$ is discrete-time Markov with transition matrix Π and for each $n \geq 1$, conditional on $Y_0, \ldots, Y_{n-1}$, the holding times $S_1, \ldots, S_n$ are independent exponential random variables of parameters $q(Y_0), \ldots, q(Y_1), q(Y_n)$ respectively.*

2. *For all $n = 0, 1, 2, \ldots$, all times $0 \leq t_0 \leq t_1 \leq \ldots \leq t_{n+1}$ and all states $i_0, i_1, \ldots, i_{n+1}$*

$$P(X_{t_{n+1}} = i_{n+1} | X_{t_0} = i_0, \ldots, X_{t_n} = i_n) = p_{i_n, i_{n+1}}(t_{n+1} - t_n).$$

If $(X_t)_{t\geq 0}$ satisfies any of these conditions, then it is called a Markov chain with generator matrix Q.

We call $\tau_0, \tau_1, \ldots$ the jump times of $(X_t)_{t\geq 0}$, where

$$\tau_0 = 0 \quad \text{and} \quad \tau_{n+1} = \inf\{t \geq \tau_n : X_t \neq X_{\tau_n}\},$$

for $n = 0, 1, \ldots$, where $\inf \emptyset = \infty$. The first explosion time φ is defined by

$$\varphi = \lim_{n \to \infty} \tau_n.$$

It is possible that φ is finite, that is the chain undergoes a infinite number of jumps in a finite amount of time. We shall not consider what happens to a process after explosion, so it is convenient to adjoint to M a new state, ∞ say, and require that $X_t = \infty$ if $t \geq \varphi$. Any process satisfying this requirement is called minimal. Proposition 2.6.1 describes some conditions that ensures that a Markov chain is minimal.

Proposition 2.6.1 *Let X be a Markov chain generated by Q. Then X does not explode if any of the following conditions holds*

1. *M is finite,*
2. $\sup_{i \in M} q_i < \infty$,
3. *$X_0 = i$, and i is recurrent for the jump chain.*

By Theorem 2.6.1, the jump time has the probability distribution

$$P(\tau_{l+1} - \tau_l \in B | X_{\tau_0} = i_0, \ldots, X_{\tau_n} = i_n) = \int_B e^{-tq(i_n)} q(i_n) dt,$$

where B is a Borel subset of $[0, \infty)$, and the post jump location at the jump time τ_{l+1} is given by

$$P(X_{\tau_{l+1}} = j | X_{\tau_0} = i_0, \ldots, X_{\tau_n} = i_n) = \pi_{i,j}.$$

Theorem 2.6.2 *Let Q be a Q-matrix, then the backward equation*

$$P'(t) = QP(t) \ , \ P(0) = I$$

has a minimal nonnegative solution $(P(t) : t \geq 0)$. *The solution* $(P(t) : t \geq 0)$ *of the backward equation is also the minimal non-negative solution of the forward equation*

$$P'(t) = P(t)Q \ , \ P(0) = I.$$

This solution also forms a matrix semigroup

$$P(s)P(t) = P(s+t) \quad s,t \geq 0.$$

The definition of irreducible continuous-time Markov chains is the same as Definition 2.2.4 of irreducible discrete-time Markov chains. However, we can no longer use the discrete definition of recurrence for the continuous case since a infinite number of return visits does not necessary imply the time of these visits will occur *ad infinitum*. For example, a chain can visit i infinitely many times before it explodes starting from state i; however, i is certainly not a recurrent state.

Definition 2.6.4 *A state* i *is recurrent if*

$$P(\{X_t = i\} \quad \textit{is unbounded} \quad t \geq 0 | X_0 = i) = 1,$$

and it is transient if

$$P(\{X_t = i\} \quad \textit{is unbounded} \quad t \geq 0 | X_0 = i) = 0.$$

Definition 2.6.4 is stronger that Definition 2.2.5 as it can be used in the discrete case. Proposition 2.2.4 is valid in the continuous case. The continuous-time analogue of T_i and $r_{j,i}^t$ are

$$T_i = \inf\{t \geq \tau_1 : X_t = i\} \text{ and } r_{j,i}^t = P(X_t = i, X_s \neq i \text{ for } \tau_1 \leq s < t | X_0 = j).$$

As in the discrete case, if $h_{i,i} = E_i(T_i) < \infty$, the chain is positive recurrent, otherwise, is null recurrent as in the discrete case.

Theorem 2.6.3 *If $q_i = 0$ or $P_i(T_i < \infty) = 1$, then i is recurrent and $\int_0^\infty p_{i,i}(t)dt = \infty$.*

Theorem 2.6.4 *If $q_i > 0$ and $P_i(T_i < \infty) < 1$, then i is transient and $\int_0^\infty p_{i,i}(t)dt < \infty$.*

Theorem 2.6.5 *Let c positive and set $Z_n = X_{nc}$.*

1. *If i is recurrent for $(X_t)_{t\geq 0}$ then i is recurrent for $(Z_n)_{n\geq 0}$.*
2. *If i is transient for $(X_t)_{t\geq 0}$ then i is transient for $(Z_n)_{n\geq 0}$.*

In other words, recurrence and transience are determined by any discrete-time sampling of $(X_t)_{t\geq 0}$.

Stationary Distributions

The notion of stationary distribution also plays an important role in the study of continuous-time Markov chains. We say v is stationary if

$$vQ = 0.$$

We say a vector $b = (b_i : i \in M)$ is a measure on M if $0 \leq b_i < \infty$ for all $i \in M$.

Theorem 2.6.6 *Let Q be a Q-matrix with jump matrix Π and let v a measure. The following are equivalent*

1. *v is stationary,*
2. *$u\Pi = u$ where $u_i = v_i q_i$.*

The equation $u = u\Pi$ can be interpreted as follows. For a state i ,$v_i q_i$ is the rate at which transitions occur out of the state; expression on the right, $\sum_{j \in M} v_j q_j \Pi_{j,i}$, is the rate at which transitions occur into state i. If $q_i = q_j$ for all $i, j \in M$ that is the exponential distribution governing the time spent have the same parameter, then Theorem 2.6.6-2 is reduced to $v\Pi = v$. Thus, the stationary distribution of the continuous-time Markov chain is the same as the stationary distribution of the embedded Markov chain.

Theorem 2.6.7 *If Q is irreducible and recurrent. Then Q has a stationary measure v which is unique up to scalar multiples.*

Theorem 2.6.8 *Let Q be an irreducible Q-matrix on M. Then the following are equivalent:*

1. *some state in M is positive recurrent,*
2. *every state in M is positive recurrent,*
3. *Q is non-explosive and has a stationary distribution v.*

Moreover when these conditions hold, we have that $h_i = \frac{1}{v_i q_i}$.

The next result justifies calling measures v with $vQ = 0$ stationary.

Theorem 2.6.9 *Let Q be irreducible and recurrent, and let v be a measure. For any $s > 0$, the following are equivalent:*

1. *$vQ = 0$,*
2. *$vP(s) = v$.*

The complete description of limiting behavior for irreducible chains in continuous-time is provided by the following result.

Theorem 2.6.10 *Let Q be an irreducible generator matrix of X and ϱ a initial distribution of X_0. Then,*

$$P(X_t = j) = \frac{1}{q_j h_j} \quad as \quad t \to \infty \text{ for all } j \in M,$$

where $\frac{1}{q_j h_j} = v_j$.

Theorem 2.6.11 *Given a Q-matrix where $q_{i,i} < \infty$ for $i \in M$, the Q-process $P(t)$ is unique if and only if the equation*

$$(\lambda + q_{i,i})\mu_i = \sum_{j \neq i} q_{i,i}\mu_j, \ 0 \leq \mu_i \leq 1, \ \text{and for all } i \in M,$$

has only the trivial solution $u_i = 0$ for some (equivalent, for all) $\lambda > 0$.

Theorem 2.6.11 has many applications though it seems hard to apply in Example 2.6.1. We are going to introduce Theorem 2.6.12 that will let us easily show that matrix Q in Example 2.6.1 is positive recurrent [4].

Example 2.6.1 *This example is a simplified version of the Schlögl model since there is one vessel rather a finite number of them.*

$$q_{i,j} = \begin{cases} \lambda_1 \binom{i}{2} + \lambda_4 & if \ j = i + 1 \\ \lambda_2 \binom{i}{3} + \lambda_3 x & if \ j = i - 1 \\ 0 & otherwise, \end{cases}$$

where matrix $Q = \{(q_{i,j}) : i, j \in \mathbb{N}\}$ is homogeneous, i indicates the number of reactions in the vessel, and $\lambda_1, \ldots, \lambda_4$ are positive constants.

Theorem 2.6.12 *Given an irreducible Q-matrix in a countable state space M where $\sup_{i \in M} q_i < \infty$. If there exists a compact function h and a constant $k \geq 0$, $\eta > 0$ such that*

$$\sum_{j \in M} q_{i,j}(h_j - h_i) \leq K - \eta \ h_i, \quad \text{for all } i \in M,$$

then the Markov chain is positive recurrent and hence has a unique stationary distribution.

In Example 2.6.1 choose $h_i = i$ and $\eta < \lambda_3$. Then we can find a finite $K = \left\{ k : \lambda_1 \binom{i}{2} + \lambda_4 - \left(\lambda_2 \binom{i}{3} + \lambda_3 \, i \right) \leq k + i(\lambda_3 - \eta) \text{ for all } i \leq m \right\}$, were $m = \min\{i : 2 + 3\frac{3\lambda_1}{\lambda_2} + \lambda_4 \leq i\}$. Hence, by Theorem 2.6.12, it is ergodic.

Schlögl introduced the model in 1972 as a typical model of non equilibrium systems. It can be solved in similar fashion by choosing $h_i = \sum_{u \in S} x(u)$, where u is a vessel in a finite set S, and $x(u)$ is the number of reactions in vessel u.

Quasi-stationary Distributions

Quasi-stationary distributions are used for modelling the long-term behaviour of stochastic systems which, in some sense terminate, but appear to be stationary over any reasonable time scale. One might wish to determine the distribution of the residual lifetime of a system at some arbitrary time t, conditional on the system being functional.

The following definition of quasi-stationary distribution is taken from Pollett [31] and introduced by van Doorn [36]. It is assumed that the system can be modelled as a time homogeneous Markov chain X taking values in a countable state space M and generated by a non-explosive Q-matrix Q. Since we are concerned with chains that terminate, for simplicity, let us take 0 to be the sole absorbing state, that is, $q_0 = 0$, and suppose that $M = \{0\} \cup C$ where $C = \{1, 2, \ldots\}$ is an irreducible transient class. In order that there exists a positive probability of reaching 0, given that the chain starts in C, we shall suppose that $q_{i,0} > 0$ for at least one $i \in C$.

Definition 2.6.5 *Let $\pi = (\pi_j, j \in C)$ be a probability distribution over C and define $p(\cdot) = (p_j(\cdot), j \in M)$ by*

$$p_j(t) = \sum_{i \in C} \pi_i p_{ij}(t), \quad j \in M,\ t > 0.$$

Then, π is a quasi-stationary distribution if, for all $t > 0$ and $j \in C$,

$$\frac{p_j(t)}{\sum_{i \in C} p_i(t)} = \pi_j.$$

That is, if π is the initial distribution of the chain, then π is a quasi-stationary if the state probabilities at time t, conditional on the chain being in C at t, are the same for all t.

2.7 Continuous-Time Martingales

Continuous-time martingales are similar to discrete-time martingales, and we introduce them since, under certain conditions, a continuous-time Markov chain can be transformed into a continuous-time martingale.

Definition 2.7.1 *Let $\Im$ be a filtration of the probability space $(\Omega, \mathscr{F}, P)$, and let $\{X_t : t \geq 0\}$ be a sequence of random variables which is adapted to $\Im$. We call the pair $(X, \mathscr{F}) = \{(X_t, \mathscr{F}_t) : t \geq 0\}$ a continuous-time martingale if, for all $t \geq 0$,*

1. *$E|X_t| < \infty$,*
2. *$E(X_t|\mathscr{F}_s) = X_s$ for all $t \geq s$.*

Similar to discrete-time martingales, we can think of $\mathscr{F}_t$ as σ-field of $\{\sigma(s) : s \leq t\}$ if no filtration is specified. Submartingales and supermartingales are defined as in Definition 2.3.2.

2.8 Continuous-Time Inhomogeneous Markov Chains

The definition of continuous-time inhomogeneous Markov chains is similar to its homogeneous counterpart. Let $X = \{X_t : t \geq 0\}$ denote a jump process defined on $(\Omega, \mathscr{F}, P)$ taking values in a finite or countable set M. Using a filtration rather than a sequence of random variables, Definition 2.6.1 can be reformulated in Definition 2.8.1.

Definition 2.8.1 *A jump process X is a Markov chain if*

$$P(X_t = i|\mathscr{F}_s) = P(X_t = i|X_s),$$

where $(\mathscr{F}_t)_{t\geq 0}$ is a filtration of the sequence of random variables $(X_t)_{t\geq 0}$.

Note that since the process is inhomogeneous, $P(X_t = i|\mathscr{F}_s)$ may not be equal to $P(X_{t-s} = i|\mathscr{F}_0)$. If a jump process has interarrival times that are not exponentially distributed and not independent, then the process is not Markovian.

For all $i, j \in M$ and $t \geq s \geq 0$, let $p_{i,j}(t, s)$ denote the transition probability $P(X_t = j|X_s = i)$, and the transition matrix of the Markov chain

$$P(t, s) = (p_{i,j}(t, s)).$$

We assume that

$$\lim_{t\to s^+} p_{i,j}(t, s) = \delta_{i,j} = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases}.$$

It follows that for $0 \leq s \leq \zeta \leq t$,

- $p_{i,j}(t, s) \geq 0$, for $i, j \in M$,
- $\sum_{j\in M} p_{i,j}(t, s) = 1$, for $i \in M$,

- $p_{i,j}(t,s) = \sum_{k \in M} p_{i,k}(\zeta, s) p_{k,j}(t,\zeta) \;\; i,j \in M.$

Definition 2.8.2 *The matrix $Q(t) = (q_{i,j}(t))$, for $t \geq 0$ satisfies the $q-$Property if*

1. *$q_{i,j}(t)$ is Borel measurable for all $i,j \in M$ and $t \geq 0$,*

2. *there exists a constant K such that $|q_{i,j(t)}| \leq K$,*

3. *$p_{i,j}(t,s) = \sum_{k \in M} p_{i,k}(\zeta, s) p_{k,j}(t,\zeta)$, for $i,j \in M$.*

For any real-valued function f and $i \in M$,

$$Q(t)f(\cdot)(i) = \sum_{j \in M} q_{i,j}(t) f(j) = \sum_{j \neq i} q_{i,j}(t)(f(j) - f(i)),$$

where the second equality follows from the definition.

Definition 2.8.3 *A matrix $Q(t)$, for $t \geq 0$ is a generator of X if it satisfies the q-Property, and for all real bounded functions f defined on M*

$$f(X_t) - \int_0^t Q(\zeta) f(\cdot)(x_\zeta) d\zeta$$

is a martingale.

We will see that for any given $Q(t)$ satisfying the q-Property, there exists a Markov chain X generated by $Q(t)$. For convenience, we will call any matrix $Q(t)$ that posses the q-Property a generator.

Let $0 = \tau_0 < \tau_1 < \ldots < \tau_l < \ldots$ denote a sequence of jump times of X such that the random variables $\tau_1, \tau_2 - \tau_1, \ldots, \tau_{k+1} - \tau_k, \ldots$ are independent Let $X_0 = i$ and $i \in M$, then $X_t = i$ on the interval $[0, \tau_1)$, and, in general, $X_t = X_{\tau_k}$ for $t \in [X_{\tau_k}, X_{\tau_{k+1}})$

The first jump has the probability distribution

$$P(\tau_1 \in B) = \int_B e^{\int_0^t q_{i,i}(s) ds} (-q_{i,i}(t)) dt,$$

where B is a Borel subset of $[0, \infty)$.

The post-jump location $X_t = j$, $X_0 = i$ and $j \neq i$, is given by

$$P(X_{\tau_1} = j|\tau_1) = \frac{q_{i,j}(\tau_1)}{-q_{i,i}(\tau_1)}.$$

If $q_{i,i} = 0$, then define $P(X_{\tau_1} = j|\tau_1) = 0$. If we let $B_i = \{t : q_{i,i}(t) = 0\}$, then

$$P(q_{i,i}(\tau_1) = 0) = P(\tau_1 \in B_i) = \int_{B_i} e^{\int_0^t q_{i,i}(s)ds}(-q_{i,i}(t))dt = 0.$$

The jump time τ_{l+1} has the conditional probability distribution

$P(\tau_{l+1} - \tau_l \in B_l|\tau_1, \ldots, \tau_l, x_{\tau_1}, \ldots, x_{\tau_l}) =$

$$\int_{B_l} e^{\int_{\tau_l}^{\tau_{l+1}} q_{X_{\tau_l}, X_{\tau_l}}(s)ds}(-q_{X_{\tau_l}, X_{\tau_l}}(\tau_1 + t))dt.$$

The post-jump location of $X_t = j$, for $j \neq X_{\tau_l}$ is given by

$$P(X_{\tau_l+1} = j|\tau_1, \ldots, \tau_l, \tau_{l+1}, X_{\tau_l}, \ldots, X_{\tau_l})) = \frac{q_{X_\tau, j}(\tau_{l+1})}{-q_{X_{\tau_l}, X_{\tau_l}}(\tau_{l+1})}.$$

Theorem 2.8.1 *If the $Q(t)$ matrix satisfies the $q-$Property for $t \geq 0$. Then,*

1. *The process X constructed above is a Markov chain.*

2. *The process*

$$f(X_t) - \int_0^t Q(\zeta)f(\cdot)(X_\zeta)d\zeta$$

is a martingale for any uniformly bounded function f on M. Thus, $Q(t)$ is a generator of X_t.

3. *The transition matrix $P(t, s)$ satisfies the forward differential equation*

$$\frac{dP(t,s)}{dt} = \lim_{h\to 0} \frac{P(t+h, s) - P(t, s)}{h} = P(t, s)Q(t), \ t \geq s,$$

$$P(s, s) = I,$$

where I is the identity matrix.

4. *If $Q(t)$ is continuous in t, then $P(t,s)$ satisfies the backward differential equation*

$$\frac{dP(t,s)}{ds} = \lim_{h\to 0} \frac{P(t,s) - P(t,s+h)}{-h} = Q(s)P(t,s), \text{ for } t \geq s,$$

$$P(s,s) = I.$$

Corollary 2.8.1 *Let X be a Markov process, $Q(t)$ a matrix satisfying the q-Property for $t \geq 0$, and f uniformly bounded real function on M, then*

$$Q(t)f(\cdot)(i) = \lim_{h\to 0} \frac{E(f(X_{t+h}) - f(i)|X_t = i)}{h}.$$

We can see the expression $Q(t)f(\cdot)(i)$ as the limiting mean rate of change of f.

Definition 2.8.4 *A generator $Q(t)$ is said to be weakly irreducible if, for each fixed $t \geq 0$, the system of equations*

$$\begin{aligned} v(t)Q(t) &= 0, \\ \sum_{i=1}^{m} v_i(t) &= 1. \end{aligned} \tag{2.8.1}$$

Note that it has a unique solution $v(t)$ and $v(t) \geq 0$.

A generator $Q(t)$ is said to be (strongly) irreducible if, for each fixed $t \geq 0$, equations (2.8.1) have a unique solution $v(t)$ and $v(t) > 0$.

The expression $v(t) \geq 0$ means that for each $i \in M$, $v_i(t) \geq 0$; a similar interpretation holds for $v_t > 0$. From the definition above, irreducible implies weak irreducible, but the converse does not hold. For example, the generator

$$Q(t) = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix},$$

is weakly irreducible since $v = (0, 1)$ is the solution for equation (2.8.1), but it is not irreducible. Once the chain reaches state 1, it never leaves it.

If a weakly irreducible Markov chain contains only one communicating class of recurrent states, and if there are no transient states, then the Markov chains is irreducible. That is, if a state i is not transient, then at every time t, there exists a state x_t such that $q_{x_t,i} > 0$, and since a (weakly) irreducible generator implies that $v(t)Q(t) = 0$, then $v_{x_t}(t)$ has to be positive.

Definition 2.8.5 *For $t \geq 0$, $v(t)$ is a quasi-stationary distribution if it is the solution of the equations in (2.8.1) and satisfies $v(t) \geq 0$.*

Definition 2.8.6 *For $t \geq 0$, $v(t)$ is a stationary distribution if it is the solution of the equations in (2.8.1) and satisfies $v(t) > 0$.*

Example 2.8.1 *Given the generator for a two-state inhomogeneous Markov chain*

$$Q(t) = \begin{pmatrix} -\lambda(t) & \lambda(t) \\ \mu(t) & -\mu(t) \end{pmatrix},$$

the generator $Q(t)$ is irreducible if both $\lambda(t) > 0$ and $\mu(t) > 0$ and it is weakly irreducible if $\lambda(t) + \mu(t) > 0$. Then $v(t) = \left(\frac{\mu(t)}{\mu(t)+\lambda(t)}, \frac{\lambda(t)}{\mu(t)+\lambda(t)} \right)$ is the corresponding stationary or quasi-stationary distribution, respectively. An equivalent description of the chain is to say that if the chain is in state $1 (or\ 2)$, then it stays in this state with a length of time exponentially distributed with parameter $\lambda(t)$ or $(\mu(t))$.

2.9 Piecewise Deterministic Continuous-Time Markov Chains

In 1980, Davis [6] introduced piecewise deterministic Markov processes (PDMPs) as a general class of continuous-time Markov processes which includes both discrete and continuous processes, except diffusions. PDMPs are suitable for formulating optimization problems in many other areas of operational research.

Starting from x an element from the state space $E \subset \mathbb{R}$, the process follows a deterministic trajectory [1] until the first jump time T_1, which occurs either spontaneously in a random manner, or when the trajectory hits the boundary of E. In both cases, a new point is selected by a random operator, and the process restarts from this new point. Consequently, if the parameters of the process under consideration are described by the state x of a piecewise deterministic process, between two jumps the system follows a deterministic trajectory.

As mentioned before in the case of events, there exist two types of jump:

1. The first one is deterministic. From the mathematical point of view, it is given by the fact that the trajectory hits the boundary of E. From the physical point of view, it can be seen as a modification of the mode of operation when a physical parameter reaches the critical value.

2. The second one is stochastic. It models the random nature of some failures or inputs modifying the mode of operation of the system, see [38].

The mathematical model related to the PDMP is as follows. Let d be a mapping from a countable set K to $\mathbb{N}$, representing the possible states of operation of the process in question. Let $(E_v^0)_{v\in K}$ be a family of open subsets of $\mathbb{R}^{d(v)}$, and, for $v \in K$, ∂E_v^0 denotes the boundary of the interior E_v^0. A piecewise deterministic

[1] For example, by the solution of an ordinary differential equation.

Markov process is determined by its local characteristics $(\Im_v, \lambda_v, Q_v)_{v\in K}$, where $\Im_v$ is a Lipschitz continuous vector field in E_v^0 determining a flow $\phi_v(x,t)$. The set

$$\Gamma^+ = \{x \in \partial E_v^0 : x = \phi(y,t), y \in E_v^0, t > 0\}$$

is the boundary point at which the flow $\phi(x,t)$ exits from E_0^v, and the set

$$\Gamma^- = \{x \in \partial E_v^0 : x = \phi(y,-t), y \in E_v^0, t > 0\}$$

is characterized by the fact the flow stating from a point in E_v will not leave E_v immediately. Thus, we can define the state space by

$$E = \{(v,x) : v \in K, x \in E_v^0 \cup \Gamma_v^- \setminus (\Gamma_v^- \cap \Gamma_v^+)\}.$$

The boundary of the state space is given by the jump rate of the process $\lambda_v : E \to \mathbb{R}^+$; the value of the PDMP right after the jump is generated by $Q_v : E \cup \Gamma^+ \times E \to [0,1]$ being the transition measure of the PDMP state after the jump, given that v is the state of the PDMP immediately before the jump. It satisfies the following property

$$[\forall (v,x) \in K \times E \cup \Gamma^+], Q_v[x, E \setminus \{(v,x)\}] = 1,$$

that is the transition measure ensures that the jump has to be to a different state.

Suppose the PDMP starts with $v_0 \in K$ and $x_0 \in E$, the evolution of the PDMP $X_t = (m_t, x_t)$. The first jump T_1 can be defined as follows

$$P_{X_0}(T_1 > t) = I_{[t<t^*_{v_0}(x_0)]} \cdot \exp\left[-\int_0^t \lambda_{v_0}[\phi(x_0,s)]ds \right],$$

where $t^*_{v_0}(x_0) = \inf\{t : t > \phi(t) \in \partial E_{v_0}\}$. The trajectory of X_t for $t \in [0, T_1]$ is given

by

$$\begin{cases} x_t = & \phi_{v_0}(x_0, t) \\ m_t = & v_0 \end{cases},$$

thus, the state space of this process is defined by the product of a Euclidean space and a discrete set.

At time T_1 the process jumps to a new location and to a new regime defined by the random variable $X_1 = (v_1, x_1)$ with probability distribution $Q_{v_0}[\phi(x_0, t), \cdot]$. Starting from X_1, the next inter-jump time $T_2 - T_1$ and post-jump location $X_2 = (v_2, x_2)$ are selected in similar way. Under some technical hypotheses, the process defined is Markovian with piecewise deterministic continuous trajectories and jump times $T_1, T_2, \ldots$ and post-jump locations $X_1, X_2, \ldots$.

Chapter 3

The Dynamic Traveling Salesman Problem

If a salesman, starting from his home city, is to visit exactly once each city on a given list and then return home, it is possible for him to select the order on which he visits the cities so that the total of the distance travelled in his tour is minimal. If he has the distance to tour every pair of cities, he has all the data necessary to find the minimum, but it is by no means obvious how to use these data in order to get the answer. This problem is called the travelling salesman problem (TSP).

There are three aspects of the history of any mathematical problem: how it arose, how research on it influenced other developments in mathematics, and how the problem was finally solved. If, as in the TSP, the problem is to develop an algorithm that satisfies formal or informal standards of efficiency, the TSP has not yet been solved. This modest-sounding exercise is in fact one of the most intensively investigated problems in computational mathematics, the first problem in the book Computers and Intractability [8], and the most common conversational comparator ('Why. It's as hard as the traveling salesman problem!') [20]. The origin of the TSP along with its name is unclear. There is a brief reference to the problem in the German handbook

printed in 1832 *Der Handlungsreisende wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glcklichen Erfolgs in seinen Geschften gewiss zu sein Von einem alten Commis-Voyageur ('The traveling salesman problem, how he should be and what he should do to get Commissions and to be Successful in his Business. By a veteran Traveling Salesman').*

According to Applegate et al. [1] mathematical problems related to the traveling salesman problem were treated in the 1800s by Sir William Rowan Hamilton and by Thomas Penyngton Kirkman. The general form of the TSP appears to be first studied by mathematicians such Karl Menger in the 1930s and later promoted by Hassler Whitney and Merrill Flood. Two of the earliest papers containing mathematical results concerning the TSP are by Marks [24] and Ghosh [9], appearing in the late 1940s in which they show that the expected length of an optimal tour on n vertices randomly allocated on a unit square is at least $(\sqrt{n} - \frac{1}{\sqrt{n}})/\sqrt{2}$ and more than $1.27\sqrt{n}$ respectively. Their work lead to a famous result of Beardwoood et al. [2] published in 1959 whose result states that with probability 1, as n approaches infinity the optimal tour length divided by $\sqrt{n}$ will approach some constant value β [1].

By the end of the 1960s, it was well appreciated that there appears to be a significant difference between hard problems such as the TSP, and easy problems. The problems for which there exists good algorithms are known as the P class, for polynomial time. A possibly more general class is known as NP, for non deterministic polynomial time. The NP class consists of those problems that are verifiable in polynomial time. That is, if we are given a potential solution then we could check if the given solution is correct in polynomial time. A problem is called $NP - complete$ if every problem in NP is polynomial reducible [2] to it. The problems for which the existence of a polynomial-time algorithm implies that every NP problem has

[1] We will later review this asymptotic property in Section 3.1

[2] Let A be an algorithm for the solution of problem B. A problem C is polynomially reducible to problem B if it can be solved in polynomial time by an algorithm that uses A as a subroutine provided that each subroutine call of A counts as one step.

a polynomial-time algorithm, are called $NP - hard$ problems. In 1972, Karp [14] showed that the TSP is $NP - hard$. The algorithm developed by Held and Karp in 1972 still carries the best known guarantee on the running time of a general solution method for the TSP with a $O(n^2 2^n)$ bound. Since deterministic TSP solutions are hard to solve, heuristic TSP methods started being developed.

Heuristic methods are used to speed up the process of finding a satisfactory solution, where an exhaustive search is impractical. If a heuristic algorithm generates solutions that are reasonably close to the optimal in polynomial time, then the heuristic algorithm is called an approximation algorithm. A well known approximation algorithm for the TSP is the Christofides method (1976), which guarantees a solution within a length at most of 1.5 times the optimum. Measuring the performance of a heuristic algorithm requires knowing the optimal TSP tour. A common way of measuring the performance of TSP heuristics is to compare its results to the Held-Karp lower bound. This measure, which is relatively quick and easy to compute, is useful when evaluating the quality of near optimal solutions for large problems where the true optima are not known. The Held-Karp lower bound can be found in polynomial time by using the simplex method and a polynomial constraint separation algorithm [35]. For example, NN averages less than 24% above the Held-Karp lower bound on random Euclidean instances with N ranging from 10,000 to 1,000,000, while for a selection of 15 of the largest 2-dimensional instances from Version 1.2 of TSPLIB [3](including all 11 with $N > 3000$), NN averaged roughly 26% above [13]. Christofides algorithm normally keeps within 15% to 20% of the Held-Karp lower bound with a complexity $O(n^3)$ [28].

The dynamic traveling salesman problem (DTSP) is a combinatorial optimization problem where the objective is to minimize the Euclidean distance that takes to visit all the demands in a dynamically changing environment. In the classic TSP,

[3]TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types.

one tries to minimize the time in a static environment that is known before starting the travel, while in the DTSP new demands appear randomly at a Poisson rate. The distribution of the demands in the Euclidean plane is uniform and independent. According to Regan et al. [33], Psaraftis first introduced the DTSP in 1988 [32].

The traditional TSP can be said to be static as well as deterministic since TSP deals with demands whose location are known in advance to the planning process; this provides a perfect set-up for applying advanced mathematical based optimization methods such as partitioning [23]. The traditional travelling salesman problem could be formulated as:

1. All information relevant to the planning of the routes is assumed to be known by the planner before the routing process begins.

2. Information relevant to the routing does not change after the routes have been constructed.

whereas in the dynamic counterpart of the traveling salesman problem considers a TSP in which a subset of new demands arrives after demands are being served; it can be summarized as:

1. Not all information relevant to the planning of the routes is known by the planner when the routing process begins.

2. Information can change after the initial routes have been constructed.

A related problem to the DTSP is the dynamic traveling repairman problem DTRP. We will talk about some results belonging to the DTRP that will be helpful in the analysis of the DTSP.

3.1 A Stochastic and Dynamic Vehicle Routing Problem in The Euclidean Plane

In the DTRP the vehicle serves demands, in a dynamic and stochastic environment, with the goal of minimizing the total waiting time of demands rather than the total travel distance in the system. As in the DTSP demands arrive at a Poisson rate and are uniformly and independently distributed in a Euclidean service region.

Our study of the DTSP is based on the work of Bertsimas et al. [3] on DTRP, whose work was initially motivated by Psafarftis's definition of the DTSP. The DTRP closely resembles the DTSP, and, as is the case of the TSP, the TRP is NP-complete [34]. Their work analyses the performance of the DTRP under different policies and traffic intensities and is briefly explained in this section.

Tools

Before talking about their results, we need to introduce some mathematic tools used in their work.

Queues Queuing theory can be described as follows: consider a server and a population of demands, which at some times enter the server in order to be serviced. It is often the case that the server can only serve a limited number of demands. If a new demand arrives and the server is exhausted, it enters a waiting line and waits until the server becomes available. So we can identify three main elements in a queue: the arrival of demands, the server and the waiting line.

The notation GI/G/1 represents a single server that has unlimited queue capacity and infinite calling population, demands are independent and follow a general distribution (that might not be exponential), and the distribution of the service time may follow any general statistical distribution.

It is known [15] that the expected waiting time W in a GI/G/1 queue for demands is

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)}, \tag{3.1.1}$$

where $\frac{1}{\lambda}$ is the expected interarrival time, $\overline{s}$ is the expected service time, $\rho = \lambda\overline{s}$ is the traffic intensity, and σ_a^2 and σ_s^2 are the variances of the interarrival and service time distribution, respectively. If $\rho \to 1$, the upper bound is asymptotically exact.

M/G/1 queue represents a single server that has unlimited queue capacity and infinite calling population. The arrival of demands is a Poisson process and the distribution of the service time may follow any general statistical distribution. It is known [16] that the expected waiting time W is

$$W = \frac{\lambda\overline{s^2}}{2(1-\rho)}, \tag{3.1.2}$$

where $\overline{s^2} = \sigma_s^2 + \overline{s}^2$ is the second moment of the service time and $\rho = \lambda\overline{s}$.

If we consider a queueing system that contains k queues $Q_1, Q_2, \ldots, Q_k$ each with finite capacity. Customers arrive according to a Poisson process with rate $\frac{\lambda}{k}$. The queues are served by a single server that serves each queue until is empty before proceeding to the next one in a fixed cyclic order. The travel time around the circles is a constant d. The service time at every queue are independent and identically distributed random variable with mean $\overline{s}$ and second moment $\overline{s^2}$. The expected waiting time for this system is

$$W = \frac{\lambda\overline{s^2}}{2(1-\rho)} + \frac{1-\frac{\rho}{k}}{2(1-\rho)}d, \tag{3.1.3}$$

where $\rho = \lambda\overline{s}$.

Geometric Probability Through the analysis of the different policies the expected distance the server needs to travel plays an important role. Given X_1 and X_2 two uniformly and independently distributed random variables in a square of area A, then from [19]

$$E\|X_1 - X_2\| = c_1\sqrt{A} \text{ and } E\|X_1 - X_2\|^2 = c_2 A,$$

where $c_1 \approx 0.52$ and $c_2 \approx \frac{1}{3}$. If x^* is the center of a square of area A, then

$$E|X_1 - x^*| = c_3\sqrt{A} \text{ and } E|X_1 - x^*|^2 = c_4\sqrt{A},$$

where $c_3 \approx 0.838$ and $c_4 = \frac{1}{6}$.

Asymptotic Properties of the TSP in the Euclidean Plane Let $X_1, X_2, \ldots, X_n$ be independently and uniformly distributed demands in a square of area A and L_n denotes the length of the optimal tour through the points. Then there exist a constant β_{TSP}, such that

$$\lim_{n\to\infty} \frac{L_n}{\sqrt{n}} = \beta_{TSP},$$

with probability 1 [2]. The estimated value of β_{TSP} is $\beta_{TSP} \approx 0.72$ [12]. It is also known [20] that

$$\lim_{n\to\infty} \frac{V(L_n)}{n} = 0.$$

Space Filling Curves An N-dimensional space-filling curve is a continuous, surjective function from the unit interval $[0, 1]$ to the hypercube $[0, 1]^N$. Let $\mathcal{C}$ be the unit circle and $\mathcal{S}$ be the unit square. Let ψ be a 2-dimensional space filling curve from $\mathcal{C}$ onto $\mathcal{S}$. The following properties results were obtained from Platzman et al.[30]. If $\theta, \theta' \in \mathcal{C}$, then

$$\|\psi(\theta) - \psi(\theta')\| \leq 2|\theta - \theta'|.$$

If $X_1, \ldots, X_n$ are any n points in $\mathcal{S}$ and L_n is the length of a tour of these n points

formed by visiting them in increasing order of their preimages in $\mathcal{C}$, then

$$L_n \leq 2\sqrt{n}.$$

If the points $X_1, \ldots, X_n$ are independently and uniformly distributed in $\mathcal{S}$, then there exists a constant β_{SFC}, such that

$$\lim_{n\to\infty} \sup \frac{L_n}{\sqrt{n}} = \beta_{SFC} \approx 0.956,$$

with probability one.

Problem Definition and Notation

The problem is defined in a convex bounded region $\mathcal{A}$ of area A that contains a server that travels at a constant unit velocity between demands. Demands for service arrive according to a Poisson process with rate λ, and their location are independent and uniformly distributed in $\mathcal{A}$. Each demand i requires an independent and identically distributed amount of on-site service s_i with mean $\overline{s} > 0$ and second moment $\overline{s^2}$. It is assumed, for simplicity, that $\mathcal{A}$ is a square area.

The traffic intensity is given by $\rho = \lambda\overline{s}$. The elapsed time between a demand i arrives and its service is completed is denoted by T_i. The waiting time of a demand i, W_i is defined by $W_i = T_i - s_i$. The steady-state system time T is defined by $T = \lim_{i\to\infty} E(T_i)$ and $W = T - \overline{s}$. Since on-site service times are randomly assigned, the goal is to find a policy that minimizes T, and this optimal system time is denoted by T^*.

The M/G/1 model represents a single server visiting demands that arrive at Poisson rate, where each demand requires an identical and independent general distribution to be served. Note that we cannot treat the DTRP as a M/G/1 queue since in the total service time of the DTRP we have to consider the travel time and the on-site

time. Although the on-site service times are independent, the travel times generally are not. Hence, the total service time are not identically distributed random variables, and therefore the methodology of M/G/1 queues is not applicable.

Lower Bound on the Optimal Policy

The performance of the proposed policies used in the DTRP will be evaluated with respect to two lower bounds. When $\rho \to 0$ that is when the arrival rate of new demands is significantly smaller than the expected service time, the following light traffic lower bound is used:

$$T^* \geq \frac{E(\|X - x^*\|)}{1-\rho} + \frac{\lambda \overline{s^2}}{2(1-\rho)} + \overline{s},$$

where x^* is the median of the region $\mathcal{A}$. If $\mathcal{A}$ is a square then $E(\|X - x^*\|) = 0.383\sqrt{A}$.

When $\rho \to 1$ that is when the arrival rate of new demands is approximately the same as the expected service time, the following heavy traffic lower bound is used:

$$T^* \geq \gamma^2 \frac{\lambda A}{(1-\rho)^2} - \frac{1-2\rho}{2\lambda}, \tag{3.1.4}$$

where $\gamma \approx 0.266$. If it is assumed that locations of demands at service completion epochs are approximately uniform then the value of $\gamma = \frac{1}{2}$. The larger value of γ is used to benchmark the different policies.

Note that in the lower bound of the heavy traffic intensity the waiting time grows at least as fast as $\frac{1}{(1-\rho)^2}$ rather than $\frac{1}{(1-\rho)}$, as is the case in the classical queuing system. Moreover, it is a function of the first moment of the on-site service time, another important difference from classical queuing system.

Proposed Policies

Several policies were proposed for the DTRP. The first-come, first-served policy (FCFS) and the stochastic queue median policy (SQM) were evaluated under light traffic.

FCFS If demands are present at the end of a service, next demand is served according to FCFS policy; however, when there is not any unattended demand after a service completion, the server waits until a new demand arrives before moving. Because demands locations are independent of the order of arrival and the number of demands in queue, the system behaves like a M/G/1 queue. Note that the travel times are not strictly independent [4], but they are identically distributed as it is the distance between two independent uniformly distributed locations in $\mathcal{A}$. Thus, formula 3.1.2 can be used to find the average system time of the FCFS policy:

$$T_{FCFS} = \frac{\lambda(\overline{s^2} + 2c_1\sqrt{A}\overline{s} + c_2 A)}{2(1 - \lambda c_1\sqrt{A} - \rho)} + \overline{s} + c_1\sqrt{A},$$

where $c_1 \approx 0.52$. This policy is stable when $\lambda c_1\sqrt{A} + \rho < 1$; thus, it is unstable when $\rho \to 1$. For the light traffic case,

$$\frac{T_{FCFS}}{T^*} \leq \frac{+ c_1\sqrt{A}}{\overline{s} + c_3\sqrt{A}} \text{ as } \lambda \to 0,$$

where $c_1 \approx 0.52$, $c_2 \approx 1/3$, and $c_3 \approx 0.383$. The worst case scenario for this policy is when $\overline{s} \to 0$ then

$$\frac{T_{FCFS}}{T^*} \leq \frac{c_1}{c_2} \approx 1.36.$$

[4] Consider the case the last traveled distance was $\sqrt{2A}$ that is the server is currently in one corner.

SQM The FCFS policy can be modified to achieve asymptotic optimal performance in light traffic. Consider the policy of locating the server at the median of $\mathcal{A}$ and following a FCFS policy, where the server travels directly to the service site from the median, service the demand, returns to the median after the service is completed, and waits there if no new demands are present. This policy is called the stochastic queue median policy (SQM). Similarly as in the FCFS policy, the system behaves as an M/G/1 queue. However, SQM varies from a system viewpoint since each service time includes on-site service plus the round-trip from the median and the demand but, from an individual demand viewpoint, includes the wait in queue, one-way travel time to the service location, and on-site service time. The average system time under this policy using equation 3.1.2:

$$T_{SQM} = \frac{\lambda(\overline{s^2} + 4c_3\sqrt{A}\overline{s} + 4c_4A)}{2(1 - 2\lambda c_3\sqrt{A} - \rho)} + \overline{s} + c_3\sqrt{A},$$

where $c_3 \approx 0.383$, $c_4 \approx 1/6$, and the stability of the policy when $2\lambda c_3\sqrt{A} + \rho < 1$. Then,

$$\frac{T_{SQM}}{T^*} = 1 \text{ as } \lambda \to 0.$$

Thus, the SQM policy is asymptotically optimal as λ approaches zero.

The FCFS and SQM policies become unstable for $\rho \to 1$ since the average distance traveled per service $\overline{d}$ remains constant, so $\overline{d}$ must decrease as λ increases. A policy that is stable for all values of ρ needs to increasingly restrict the distance the server can travel to service demands as ρ grows. In the case of heavy traffic, the following policies were analyzed: partitioning (PART), traveling salesman (TSP), space filling curves (SFC), and nearest neighbor (NN).

PART The PART policy restrict the distance the server can travel through a partition of the (square) service region $\mathcal{A}$ into m^2 equal subregions where m is even, so the

server can perform a closed tour. The value of m increases with ρ as the size of the partitions restrict the distance the server can travel. The server services the demands of a subregion following a FCFS policy, and when no more demands are presents, the server travels in a straight line to the next adjacent subregion and services until no demands are left. This pattern is continuously repeated. For simplicity, it was considered that the last location of a given subregion is projected onto the next subregion to determine the server's new starting location though in practice the server can start in the first demand of the new subregion. Each subregion behaves as an M/G/1 queue, and the policy as a whole behaves as a cyclic queue with m^2 queues, where the optimal m is choose according $\lambda, \overline{S},$ and A, then

$$T_{PART} \approx 2c_1 \frac{2\lambda c_1 \sqrt{A}}{(1-\rho)^2} + \frac{\lambda \overline{s^2}}{1-\rho}.$$

When $\rho \to 1$,

$$\frac{T_{PART}}{T^*} \leq 4.2 \text{ as } \rho \to 1,$$

when γ takes the conjectured value 1/2. Thus, when $\rho < 1$ there exists an optimal policy.

TSP The TSP is based on collections of demands into sets that can then be served using an optimal TSP tour. Let N_k be the k^{th} set and n a parameterizing constant that indicates the number of demands in each set. The first n demands are assigned to N_1, the following $n+1$ to $2n$ demands to N_2, etc. When all demands in N_1 have arrived, a TSP tour starting and ending in the server's depot (randomly located) will visit the n demands from set N_1, and if all demands in N_2 have arrived when the tour on N_1 is completed, they are served using a TSP tour; otherwise, the server waits until demands in N_2 have arrived before serving it. Thus, sets are queued and are serviced in a FCFS order. Since the iterarrival time, the time for n new demands to arrive,

and service time, n on-site services plus the travel time of the tour, are identically distributed, the service of sets forms a GI/G/1 queue, where the interarrival time follows a gamma distribution with shape n and parameter $\frac{1}{\lambda}$. After using equation 3.1.1 for the mean waiting time of GI/G/1 queues with the asymptotic properties of the TSP, and finding an optimal value of m, the average system time for this policy is

$$T_{TSP} \leq \beta_{TSP}^2 \frac{\lambda A}{(1-\rho)^2} + \frac{\beta_{TSP}\lambda\sqrt{A(\frac{1}{\lambda^2}+\sigma_s^2)}}{(1-\rho)^{\frac{3}{2}}} + \frac{\beta_{TSP}^2 \lambda A}{1-\rho} \ \rho \to 1,$$

and by using the heavy traffic lower bound,

$$\frac{T_{TSP}}{T^*} \approx 2 \text{ as } \rho \to 1.$$

As in practice the TSP policy is heuristic rather than optimal, the ratio can be slightly larger than 2.

SFC Let ψ, $\mathcal{C}$, and $\mathcal{S}$ be defined as in the Tools Section. Then the SFC policy is to service demands as they are encountered in repeated clockwise sweeps of the circle $\mathcal{S}$, where the depot could be treated as a permanent demand and be visited once per sweep. Let W_0 denote the waiting time of a tagged demand, $\mathcal{N}_0$ denote the set of locations of the N_0 demands served prior to to W_0, and L denote the length of the path from the server's location through $\mathcal{N}_0$ to W_0 induced by the SFC rule. Let s_i be the on-site service time of demands $i \in \mathcal{N}_0$, and R be the residual service time of the demand under service. Then

$$W_0 = \sum_{i \in \mathcal{N}_0} s_i + L + R,$$

then

$$W = E(N_0)\overline{s} + E(L) + \frac{\lambda \overline{s^2}}{2}. \tag{3.1.5}$$

In steady state, the expected number of demands served during a wait is equal to the demands that arrive, thus $E(N_0) = N = \lambda W$. Since L is the length of a path through $N_0 + 2$ points in the square $\mathcal{A}$, from the Space Filling Curve Subsection from Tools, $L \leq 2\sqrt{(N_0+2)A}$. Then, by using Jensen's inequality in the third statement,

$$E(L) \leq 2E\sqrt{(N_0+2)A} \leq 2\sqrt{(N+2)A} \leq 2\sqrt{\lambda W A} + 2\sqrt{2A}.$$

Plugging these results into 3.1.5 and solving W where $T = W + \overline{s}$, then

$$T_{SFC} \leq \gamma_{SFC}^2 \frac{\lambda A}{(1-\rho)^2} + o\left(\frac{1}{(1-\rho)^2}\right), \tag{3.1.6}$$

where $\gamma_{SFC}^2 \leq 2$. The value of γ_{SFC} is based on the worst case tour and is probably too large. If it is assumed that the clockwise interval between the preimages of the server and the tagged demand is a uniform $[0, 1]$ random variable and the $\mathcal{N}_0$ points are uniformly distributed on this interval, then $\gamma_{SFC} \approx 0.64$ -the simulated value of $\gamma_{SFC} \approx 0.66$-. Thus, the system for this policy is therefore about 15% lower than that of the TSP policy. Equation 3.1.6 shows that SFC policy grows within constant factor of optimal.

NN Finally, the NN policy was considered for two reasons: 1) NN was used in the heavy traffic lower bound of the equation 3.1.4, and 2) the shortest processing time rule is known to be optimal for the classic M/G/1 queue [5]. Let d_i be the travel distance to the demand i from the location of previous demand served. Because of the dependencies among the travel distance d_i, there was no rigorous analytical result produced for the NN policy, but if it is assumed there exists a constant γ_{NN} such that

$$E(d_i|N_T) \leq \gamma_{NN}\sqrt{\frac{A}{N_T}}, \tag{3.1.7}$$

where N_T is the number of demands in the system at a completion epoch, then it is possible to show that [17]

$$T_{NN} \leq \gamma_{NN}^2 \frac{\lambda A}{(1-\rho)^2} \text{ as } \rho \to 1.$$

The authors performed simulation experiments identical to those on the SFC policy to verify the asymptotic behavior of T_{NN} and estimate γ_{NN}. The value of $\gamma_{NN} \approx 0.64$ that is NN is 10% faster than T_{SFC}. The simulations showed that the system time follows the $\frac{\lambda A}{(1-\rho)^2}$ growth predicted by the lower bound in equation 3.1.4.

Conclusion

The DTSP and the DTRP consider that the region where demands arrive is a unit square in the Euclidean space, and both severs travel at a constant unit velocity; however, both problems differ in the conditions and objectives. In the DTRP, the mean service time is positive, and their objective is to reduce the mean waiting time of the demands, while, in the DTSP, the service time of each demand is zero, and the objective is to establish the mean time the system has no demands to serve for the first time. However, DTRP can give us some insight of what should be a good policy.

We have seen that the best DTRP policy in light traffic is SQM. When $\rho \to 0$, the server, in the SQM policy, tends to be free after a demand is served, and, by positioning in the center of the region when it is free, the server reduces the expected travel time to the new demand -and so the demand's mean waiting time-. On the other hand, the rest of the policies (FCFS, TSFC, and NN) which do not try to locate the server in a position that would leave it close to the new demand to come have poorer performance. However, as the traffic ρ increases, all these policies outperform

SQM. These policies put less emphasis on prioritizing the order demands arrive and more emphasis on serving as many demands as possible in the short term. The policies that best perform are the NN and SFC which completely ignore the order of arrival of demands, whereas the TSP policy, which serves blocks organized by the order of arrival, and the PART policy, which serves demands in each partition according to the FCFS policy, keep some consideration in the order demands arrived and are less efficient than NN and SFC.

In other words, policies that under heavy traffic are able to reduce the total waiting time of demands by reducing the coefficient between the length of the path and the number of demands served perform better. Moreover, since the service time is out of control in any policy, good heavy traffic policies in the DTRP that focus in reducing the travel distance from one demand to the other will perform better. Finally, since in the DTSP we seek to reduce the travel distance from one demand to the other, good DTRP heavy traffic policies should be efficient when used in the DTSP.

3.2 The DTSP with the NN Policy as a Discrete Markov Chain

Let ξ_t be the set of unattended demands, including the new demands created, at the moment the t^{th} demand $x_{\sigma(t)}$ is served. Let $\theta_{(t)}$ be the set of new demands uniformly distributed in the unit square generated by a Poisson process with rate λ on time interval of size d_t. The time $d_t = \| x_{\sigma(t-1)} - x_{\sigma(t)} \|_2$ is the shortest distance from demand $x_{\sigma(t-1)}$ to the rest of the unattended demands ξ_{t-1}, where $x_{\sigma(t)} \in \xi_{t-1}$.

If we define the triple

$$X_t = \{\xi_t \ , \ x_{\sigma(t)} \ , \ x_{\sigma(t+1)}\} \text{ for } t = 1, 2, \ldots \tag{3.2.1}$$

where the first element $\xi_t = \xi_{t-1} \setminus x_{\sigma(t)} \cup \theta_{(t)}$ is the set of unattended demands that evolves subtracting one served demand and adding the new generated demands. The second element $x_{\sigma(t)}$ is the t^{th} visited demand and the reference point used to find the closest unattended demand $x_{\sigma(t+1)}$ from ξ_t. The system starts with $X_1 = \{\{x_2\}\ ,\ x_1\ ,\ x_2\}$, where $x_{\sigma(1)} = x_1$ and $x_{\sigma(2)} = x_2$, and it grows according to equation 3.2.1. The process X is a discrete Markov chain since the state X_t depends on the previous state X_{t-1}, and it is independent of how the process arrived to state X_{t-1} since the order in which demands arrive is irrelevant.

The DTSP with NN policy (DTSPNN) consists in generating a path $L_t = L(x_{\sigma(1)}, x_{\sigma(2)}, \ldots)$ that connects with constant unit velocity demands that are created according to a Poisson random variable $Z_\lambda(t)$ with mean λt and uniformly distributed in a unit square. Given a time limit $T_\lambda > 0$, the process stops when either there are no more points to visit or $|L_t| \geq T_\lambda$. Demands are chosen according to the closest distance to the server.

Let $x_{\sigma(t)}$ be the t^{th} demand served at time $\sum_{k=2}^{t} d_k$ and the closest unattended demand created up to time $\sum_{k=2}^{t-1} d_k$ from $x_{\sigma(t-1)}$, where $d_t = \| x_{\sigma(t-1)} - x_{\sigma(t)} \|_2$. The set of demands created on a time interval d_t by $Z_\lambda(d_t)$ is denoted by $\theta_{(d_t)}$. We can summarize the algorithm in the following steps:

1. Start with $t = 2$ and two random demands x_1 and x_2 where x_1 is the starting position, so $L_t = L(x_{\sigma(1)}, x_{\sigma(2)}) = L(x_1, x_2)$.

2. Generate $Z_\lambda(d_t)$ new demands where $d_t = \| x_{\sigma(t-1)} - x_{\sigma(t)} \|_2$.

3. Visit the closest demand from $x_{\sigma(t)}$ to $\bigcup_{k=2}^{t} \theta_{(d_k)} \setminus \{x_{\sigma(1)}, \ldots, x_{\sigma(t)}\}$ denoted by it $x_{\sigma(t+1)}$.

4. If either
$$\bigcup_{k=2}^{t} \theta_{(d_k)} \setminus \{x_{\sigma(1)}, \ldots, x_{\sigma(t+1)}\} \neq \emptyset$$

or

$$|L_t| = \sum_{k=2}^{t} d_k < T_\lambda,$$

set $t = t + 1$ and go back to step 2. Otherwise, stop.

We will refer to one execution of the algorithm as an "iteration" and the whole collection of different iterations as a "simulation".

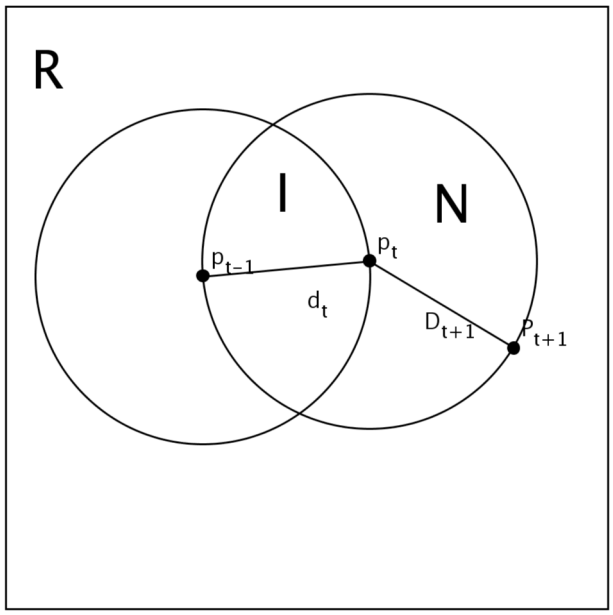
The DTSPNN can be modelled by the process $Y_t = f(X_t)$ where the function $f : X_t \to \mathbb{N} \cup \{0\}$ returns the number of unattended demands of the process X at time t, and $D_{t+1} = g(X_t)$ where the function $g : X_t \to \mathbb{R}^+$ returns the distance to the closest unattended demand from the last served demand $x_{\sigma(t)}$ at time t.

The number of unattended demands in the present does not provide enough information for the calculation of $P(Y_{t+1} = n|Y_t = m)$. The future number of unattended demands is influenced by the number of unattended demands generated by a Poisson process with rate λ in a time interval d_t; the problem lies in the fact that we cannot estimate d_t if the locations of the unattended demands are unknown.

If we knew the distribution of the unattended demands at a time t, we would be able to calculate the distribution of the number of unattended demands at a time $t + 1$ since the distance d_{t+1} can be obtained from x_t. Then, $[Y_{t+1}|X_t = x_t] = Y_t - 1 + Z_\lambda(d_{t+1})$, and if we were to have $E(Y_{t+1}|X_t) = Y_t$ then $E\big(Z_\lambda(d_{t+1})|X_t\big) = 1$ and $E(d_{t+1}|X_t) = \frac{1}{\lambda}$.

We can also obtain some information about the distribution of D_{t+1} given $X_t = x_t$. Assuming the algorithm visited demand p_{t-1} and then p_t, let D_{t+1} be the distance between last visited demand p_t and the next demand P_{t+1}; I be the area of the intersection of the two circumferences $C_1(p_{t-1}, d_t)$ and $C_2(p_t, D_{t+1})$; R be the area of the unit square not covered by either C_1 or C_2; and N be the area of C_2 that does not intersect with C_1 as shown in Figure 3.1,

Figure 3.1: Distribution of the distance to the closest unattended demand



Then,

$$P(D_{t+1} < l | X_t = x_t) = P\big(\{\exists \text{ new demands in } I \cup N\} \bigcup \{\exists \text{ old demands in } N\}\big)$$
$$= P\big(\{\exists \text{ new demands in } I \cup N\} \bigcup \{\text{all old demands are in } R\}^c\big),$$

since there are no old demands in C_1.

Since the locations of new demands are independent of the locations of old demands,

$$\begin{aligned} P(D_{t+1} < l | X_t = x_t) &= P(\exists \text{ new demands in } I \cup N) \ + \ P(\{\text{all old demands are in R}\}^c) \\ &\quad - P(\exists \text{ new demands in } I \cup N) \ \cdot \ P(\{ \text{ all old demands are in R}\}^c) \\ &= (1 - e^{-\lambda |I \cup N|}) \ + \ (\ 1 - |R|^{y_t - 1}) \ - \ (1 - e^{-\lambda |I \cup N|}) \cdot (1 - |R|^{y_t - 1}) \\ &= 1 - e^{-\lambda |I \cup N|} |R|^{y_t - 1}. \end{aligned}$$

Since the exact probability is hard to obtain, we will find a lower and upper bound. When R has the smallest area $|R|_s$ and $I \cup N$ the largest area $|I \cup N|_l$, then

$P(D_{t+1} < l | X_t = x_t) \leq 1 - e^{-\lambda |I \cup N|_l} |R|_s^{y_t - 1}$ for $0 < l < \sqrt{2}$, where

$$|R|_s = 1 - (\pi l^2 + \pi d_t^2 - |I|) \tag{3.2.2}$$

$$|I| = l^2 \cos^{-1}(\frac{l}{2d_t}) + d_t^2 \cos^{-1}(1 + \frac{l^2}{2d_t^2}) - \frac{1}{2}\sqrt{l^2(2d_t - l)(2d_t + l)} \tag{3.2.3}$$

$$|I \cup N|_l = \pi l^2. \tag{3.2.4}$$

The area $|R|_s$ is the smallest when the two circles are located such that their intersection with the unit square is the largest, and the area $|I \cup N|_l$ is the largest when the intersection between the circle with radius l is located such that its intersection with the unit square is the largest.

On the other hand, a lower bound can be calculated when it is considered that R is the largest area $|R|_l$ and $I \cup N$ the smallest $|I \cup N|_s$, then

$P(D_{t+1} < l | X_t = x_t) \geq 1 - e^{-\lambda |I \cup N|_s} |R|_l^{y_t - 1}$ for $0 < l < \sqrt{2}$, where

$$|R|_l = 1 - (\frac{\pi d_t^2}{4} + \frac{\pi l^2}{2} - \frac{1}{2}|I|) \qquad \text{if } 2l > d_t > l \tag{3.2.5}$$

$$|R|_l = 1 - \frac{\pi d_t^2}{4} \qquad \text{if } d_t > 2l \tag{3.2.6}$$

$$|R|_l = 1 - (\frac{\pi l^2}{4} + \frac{\pi d_t^2}{2} - \frac{1}{2}|I|) \qquad \text{if } 2d_t > l > d_t \tag{3.2.7}$$

$$|R|_l = 1 - \frac{\pi l^2}{4} \qquad \text{if } l > 2d_t \tag{3.2.8}$$

$$|I \cup N|_s = \frac{\pi l^2}{4} \tag{3.2.9}$$

The area $|I|$ was previously defined. Equation 3.2.5 occurs when the center of the circle with largest radius d_t is in a vertex of the unit square, and the center of the circle with smallest radius l is on a side of the unit square. When d_t is at least twice as large as l, equation 3.2.5 becomes 3.2.6. When $l > d_t$, equations 3.2.5 and 3.2.6 are equivalent to 3.2.7 and 3.2.8 respectively. Equation 3.2.9 is the area of the circle with radius l and center in a vertex of the unit square.

An upper bound of the expected distance to the next demand can be obtained, $E(D_{t+1}|X_t = x_t) = \int_0^{\sqrt{2}} P(D_{t+1} > l|X_t = x_t)\mathrm{d}l$

$$
\begin{aligned}
&\leq \int_0^{\sqrt{2}} e^{-\lambda|I\cup N|_s}|R|_l^{y_t-1}\mathrm{d}l \\
&= \int_0^{\frac{d_t}{2}} e^{-\lambda\frac{\pi l^2}{4}}(1-\frac{\pi d_t^2}{4})^{y_t-1}\mathrm{d}l + \int_{\frac{d_t}{2}}^{d_t} e^{-\lambda\frac{\pi l^2}{4}}\big(1-(\frac{\pi d_t^2}{4}+\frac{\pi l^2}{2}-\frac{1}{2}|I|)\big)^{y_t-1}\mathrm{d}l \\
&+ \int_{d_t}^{2d_t} e^{-\lambda\frac{\pi l^2}{4}}\big(1-(\frac{\pi l^2}{4}+\frac{\pi d_t^2}{2}-\frac{1}{2}|I|)\big)^{y_t-1}\mathrm{d}l + \int_{2d_t}^{\sqrt{2}} e^{-\lambda\frac{\pi l^2}{4}}(1-\frac{\pi l^2}{4})^{y_t-1}\mathrm{d}l.
\end{aligned}
$$

Equation 3.2.2 shows that when the number of unattended demands Y_t is large, the distance between old demands is small, so, with respect to d_t, D_{t+1} should be small, whereas when Y_t is small, there are not many old demands so D_{t+1} should be larger than d_t. The rate λ also plays a role in the distribution of D_{t+1} since when λ increases, the distance D_{t+1} should be smaller as the region tends to have more demands. Thus, we can assume that after some time -given that the server has not yet swept all the demands- the system will stabilize and there exists a quasi-stationary distribution for the number of unattended demands.

Let L_t be the length of the tour of the server after it visits the t^{th} demand, $\overline{u}_*$ be the mean number of unattended demands of the quasi-stationary distribution, and $\overline{t}_*$ be the mean time that takes the system to arrive to the quasi-stationary distribution.

After the number of untended demands of an iteration arrives to $\overline{u}_*$, the mean number of unattended demands of the iteration will remain close to $\overline{u}_*$ until the iteration vanishes. In order to remain around this value, the number of demands visited has to be approximately equal to the number of demands created. In other words, the mean distance between visits has to be $\frac{1}{\lambda}$,

$$E_*(d_t|X_t) = \frac{1}{\lambda} \text{ for } L_t \geq \overline{t}_*, \tag{3.2.10}$$

where E_* denotes expectation under the quasi-stationary distribution.

In Section 3.1 we have seen that if demands are independently and uniformly distributed in the unit square and served according to SFC policy, then with probability 1.

$$\lim_{n\to\infty} \sup \frac{L_n}{\sqrt{n}} \approx 0.956,$$

on the other hand, when L_n is the length of the optimal TSP tour,

$$\lim_{n\to\infty} \frac{L_n}{\sqrt{n}} \approx 0.72.$$

Motivated by these two lower bound we are going to assume that there is a positive constant c such that

$$E_*(d_t|X_t) \leq \frac{c}{\sqrt{\overline{u}_*}},$$

then

$$E_*(d_t|X_t) = \frac{1}{\lambda} \leq \frac{c}{\sqrt{\overline{u}_*}},$$

and,

$$\overline{u}_* \leq \lambda^2\ c^2.$$

In section 4.1.2 we will show an estimate that the expected number of unattended nodes $\overline{u}_*$ when the system stabilizes is associated with the rate λ,

$$\overline{u}_* = 0.468\lambda^{1.932},$$

so $c = \sqrt{0.468} \approx 0.68$. This constant is close to the one on equation 3.1.7 whose value is $\gamma_{NN} \approx 0.64$.

Chapter 4

The DTSP Simulations

Since it is believed that there is no time efficient algorithm to solve NP-hard problems, approximation algorithms are developed which generate near-optimal solutions. Probabilistic analysis of algorithms study the performance of a algorithm as a function of the input and are used to: predict the resources such as time and memory that the algorithm will consume, compare algorithm with competing alternatives, improve the algorithm by spotting the performance bottlenecks, or explain observed behavior. There are basically two categories of performance analysis, namely, combinatorial worst-case, and probabilistic average-case performance analysis. Essentially, a probabilistic analysis is based on certain assumptions on the probability distribution of instance I. Then we can find, for example, the expectation $E(A(I))$, the ratios $\frac{E(A(I))}{E(opt(I))}$ and $E(\frac{A(I)}{opt(I)})$, and the difference $E(A(I)) - E(opt(I))$, where A stands for an approximation algorithm solving a maximization problem, $A(I)$ and $opt(I)$ denote respectively the solution produced by algorithm A and the optimal solution for instance I. Or, one can show that algorithm A finds an optimal solution with high probability. The probabilistic analysis of algorithms is a refinement of worst-case analysis, which is often too pessimistic compared to the performance of algorithms in actual practice.

One common distinction is that probabilistic algorithms, unlike deterministic ones, make random choices when computing. They are commonly referred to as "coin-flipping algorithms." Such algorithms are likely to produce different results for the same problem when posed in different circumstances. On the other hand, the probabilistic analysis of an algorithm incorporates randomness into the data processed by an algorithm; that is, it considers the pair (algorithm, problem instance) and probabilistically explores the algorithm behavior over a large variety of problem instances. Typically, the analyst can make statements about the probability of selecting a particular instance, or focus attention on the distribution of suitable variables that describe the problem instance. The task is then to relate the algorithm performance to these variables.

The Monte Carlo simulation is a non deterministic method that relies on repeated random sampling to determine the properties of some phenomenon; it is used to approximate problems whose exact solution is complex and difficult to evaluate. The method can be generalized in the following steps:

1. Define a domain of possible inputs.
2. Generate random inputs from a probability distribution over the defined domain.
3. Compute the inputs.
4. Repeat steps 2 to 3 n times, where n is large.
5. Aggregate the results.

We will first use Monte Carlo to evaluate the NN policy for the DTSP with different arrival rates and then we will use it to evaluate the DTSP under modified NN policies. In the DTSP demands arrive at a Poisson rate and are both randomly and

uniformly distributed in the unit square region, but the NN will perform a deterministic computation on these inputs; that is, we will use the Monte Carlo method to perform a probabilistic analysis of a deterministic algorithm. A more detailed explanation of the implementation of the Monte Carlo simulation of the DTSP with NN policy follows.

4.1 The DTSP with Nearest Neighbour Policy

To explain the DTSPNN simulation, we will use the notation introduced in Section 3.2. We iterate the algorithm a number of times. If an iteration stops before T_λ, the exact time the iteration stopped and the number of points that visited is stored. For those iterations that did not vanish before specific moments $0 < T_{1,\lambda} < T_{2,\lambda} < \ldots < T_{N,\lambda} = T_\lambda$, we check the number of points both visited and unattended. We will use the subindices s and s' to denote iterations that did and did not stop respectively to express, using the collected information, the following results:

1. The proportion of iterations that did not stop at $T_{i,\lambda}$, denoted by $p_{s'}(T_{i,\lambda})$ [1].

2. Among the iterations that stopped before $T_{i,\lambda}$: Mean time spent $\overline{t}_s(T_{i,\lambda})$ before stopping and the mean number of served demands (visited) $\overline{v}_s(T_{i,\lambda})$.

3. Among the iterations that did not stop at time $T_{i,\lambda}$: Mean number of served demands $\overline{v}_{s'}(T_{i,\lambda})$ and the mean number of unattended demands $\overline{u}_{s'}(T_{i,\lambda})$.

4. Among all the iterations: Mean number of demands visited $\overline{v}(T_{i,\lambda})$ and mean number of demands unattended $\overline{u}(T_{i,\lambda})$.

We can establish some relations among these quantities.

- At any time $T_{i,\lambda}$:

[1] Though lightly cumbersome, the complete notation should be $p_\lambda^{s'}(T_{i,\lambda})$.

$\bar{t}_s(T_{i,\lambda}) \leq T_{i,\lambda}$; $\bar{t}_{s'}$ is not included since when $p_{s'}(T_{i,\lambda}) > 0$, $\bar{t}_{s'}(T_{i,\lambda}) = (T_{i,\lambda})$.

$\overline{u}(T_{i,\lambda}) = \overline{u}_{s'}(T_{i,\lambda})\ p_{s'}$.

$\overline{v}(T_{i,\lambda}) = \overline{v}_s(T_{i,\lambda})\ (1 - p_{s'}) + \overline{v}_{s'}(T_{i,\lambda})\ p_{s'}$.

- When $T_{i,\lambda}$ is sufficiently large:

  $\lambda\ T_{i,\lambda} = \overline{v}_{s'}(T_{i,\lambda}) + \overline{u}_{s'}(T_{i,\lambda})$ if $p_{s'}(T_{i,\lambda}) > 0$.

  $\overline{v}_s(T_{i,\lambda}) = \lambda \bar{t}_s(T_{i,\lambda})$.

Table 4.1 shows the state of the simulation at specific times T_λ so that $1 - p_{s'}(T_\lambda) \approx \{\frac{9}{10}, \frac{8}{10}, \ldots, \frac{1}{10}\}$. Each simulation of λ consists in 10,000 iterations.

Table 4.1: DTSPNN with $\lambda = 3, \ldots, 8$

λ	$T_{i,\lambda}$	$p_{s'}(T_{i,\lambda})$	$\bar{t}_s(T_{i,\lambda})$	$\overline{v}_s(T_{i,\lambda})$	$\overline{v}_{s'}(T_{i,\lambda})$	$\overline{u}_{s'}(T_{i,\lambda})$	$\overline{v}(T_{i,\lambda})$	$\overline{u}(T_{i,\lambda})$
3	0.79	0.6984	0.377586	1.16479	2.03207	2.52864	1.7705	1.766
	1.59	0.5996	0.561904	1.52448	4.07655	3.21748	3.0547	1.9292
	3.6	0.4999	0.942441	2.41812	9.9946	3.90178	6.2056	1.9505
	7.32	0.3998	1.67949	4.44302	21.8159	4.23512	11.3887	1.6932
	12.56	0.3	2.83165	7.87757	38.902	4.282	17.1849	1.2846
	20.25	0.2	4.49985	13.0779	63.866	4.37	23.2355	0.874
	33.39	0.1	6.91583	20.7657	106.84	4.29	29.3731	0.429
	150.83	0	11.4618	35.4313	-	-	35.4313	0
4	0.3	0.8976	0.171958	1.02246	1.11854	2.42747	1.1087	2.1789
	0.66	0.7993	0.309862	1.11809	1.77756	3.08557	1.6452	2.4663
	2.66	0.6998	0.637834	1.83744	7.95241	5.17219	6.1167	3.6195
	18.47	0.6	2.74702	9.09275	70.6742	6.775	46.0416	4.065
	41.51	0.5	8.12959	30.0558	164.322	6.8136	97.1888	3.4068
	68.62	0.4	15.8747	60.8272	274.623	6.74025	146.345	2.6961
	103.75	0.3	25.7425	100.469	417.607	6.75033	195.611	2.0251
	153.56	0.2	38.4372	151.708	620.784	6.8355	245.524	1.3671
	242.33	0.1	55.6494	221.695	980.424	6.866	297.568	0.6866
	1391.42	0	86.5959	347.283	-	-	347.283	0
5	0.4	0.8889	0.206478	1.05131	1.27517	3.25582	1.2503	2.8941
	3.8	0.7998	0.547544	1.74476	13.4546	7.87759	11.1103	6.3005
	192.8	0.6999	30.8405	149.996	957.235	10.1137	714.982	7.0786
	417.1	0.6	98.4164	486.333	2081.07	10.1862	1443.17	6.1117
	694.5	0.5	187.978	934.024	3470.82	10.18	2202.42	5.09
	1020.9	0.4	299.405	1491.8	5105.31	10.1928	2937.21	4.0771
	1456.8	0.3	431.785	2154.83	7287.75	10.1567	3694.71	3.047
	2041.1	0.2	593.714	2965.51	10213.7	10.046	4415.15	2.0092
	3015.8	0.1	802.158	4008.61	15094.5	10.238	5117.2	1.0238
	12593.6	0	1175.75	5879.56	-	-	5879.56	0
6	5	0.8581	0.419118	1.50035	20.8245	11.3462	18.0824	9.7362
	2230	0.8	316.611	1892.99	13367.2	14.4221	11072.4	11.5377
	6520	0.6998	1661.95	9958.41	39104.6	14.2844	30354.9	9.9962
	11335	0.6	3463.88	20767.1	67993.1	14.3383	49102.7	8.603
	17180	0.5	5598.16	33572.8	103055	14.3698	68314	7.1849
	23990	0.4	8062.77	48357.4	143907	14.4093	86577.2	5.7637
	33305	0.2999	10993.2	65937.3	199799	14.2768	106082	4.2816
	45490	0.2	14517.4	87079.9	272915	14.1995	124247	2.8399
	67845	0.1	19040.9	114221	407025	14.334	143501	1.4334
	446080	0	26920.7	161493	-	-	161493	0
7	100000	0.8026	22035.5	154271	700131	19.4047	592378	15.5742
	110000	0.7938	25582.8	179106	770143	19.2982	648271	15.3189
	230000	0.6962	72216.6	505612	1.61033e+6	19.2623	1.27472e+6	13.4104
	380000	0.5932	130982	917058	2.66058e+6	19.4093	1.95131e+6	11.5136
	550000	0.4946	195558	1.36922e+6	3.85083e+6	19.4127	2.59662e+6	9.6015
	750000	0.3981	268028	1.87662e+6	5.25113e+6	19.4479	3.22001e+6	7.7422
	1.02e+6	0.2983	354751	2.48381e+6	7.14150e+6	19.3027	3.87320e+6	5.758
	1.39e+6	0.1986	459414	3.21661e+6	9.73196e+6	19.1813	4.51056e+6	3.8094
	2.01e+6	0.0995	592565	4.14886e+6	1.40728e+7	19.4472	5.13630e+6	1.935
8	4e+6	0.883	997361	7.98010e+6	3.20068e+7	24.9785	2.91957e+7	22.056
	1e+7	0.766	4.50188e+6	3.60226e+7	9.60205e+7	25.5666	8.19810e+7	19.584
	2e+7	0.657	8.18151e+6	6.54663e+7	1.60034e+8	25.2938	1.27597e+8	16.618
	2e+7	0.569	1.14394e+7	9.15347e+7	2.24049e+8	24.7469	1.66935e+8	14.081
	4e+7	0.455	1.62610e+7	1.30116e+8	3.20069e+8	25.1648	2.16544e+8	11.45
	4e+7	0.378	1.97410e+7	1.57962e+8	3.84083e+8	24.4709	2.43436e+8	9.25
	6e+7	0.296	2.36231e+7	1.89025e+8	4.80103e+8	25.4561	2.75184e+8	7.535
	8e+7	0.199	2.92357e+7	2.33935e+8	6.40140e+8	25.1307	3.14770e+8	5.001
	1.16e+8	0.096	3.68867e+7	2.95156e+8	9.28203e+8	25.5	3.55929e+8	2.448

Conclusion

An important result shown in Table 4.1 is that all iterations eventually vanish; that is, $p_{s'}(T_{i,\lambda}) \to 0$ when $T_{i,\lambda} \to \infty$. For $\lambda \geq 3$ [2], $\overline{u}_{s'}$ stabilizes after a short time compared to the time taken to have all the iterations finished with no unattended demands. If we can estimate, among the iterations that did not finish, the mean number $\overline{u}_*$ of unattended demands the process stabilizes, then we can estimate, among the iterations that did not finish, the mean time $\overline{t}_*$ at which the number of untended demands stabilize. The estimation of $\overline{t}_*$ is important as it indicates when the simulation stabilizes, and once the simulation stabilizes, we can produce some predictions. Table 4.2 considers $\lambda = 5$ and 10,000 iteration, and it shows the evolution of the same simulation from Table 4.1 from a different point of view. The first time we observe the simulation is at time 50 since by this time, according to Table 4.3, $\overline{u}_{s'}$ is stabilized. After time 50, we continue observing the status of the simulation every 300 unit of times. In Table 4.1 times $T_{i,\lambda}$ are chosen so that $1-p_{s'}(T_\lambda) \approx \{\frac{9}{10}, \frac{8}{10}, \ldots, \frac{1}{10}\}$, whereas in Table 4.2 times $T_{i,\lambda}$ are equally spaced. In the third column of Table 4.2, we have added the ratio $\frac{p_{s'}(T_{i,5})}{p_{s'}(T_{i-1,5})}$ between the current and previous proportion of iterations that did not finish. This ratio remains between 0.79 and 0.83 when the number of iterations are significant and stabilized; that is, under certain conditions, the iterations in a simulation vanish following a geometric distribution whose parameter is $\frac{p_{s'}(T_{i,5})}{p_{s'}(T_{i-1,5})}$.

[2] We avoided the case when $\lambda \leq 3$ since iterations are short-lived, and so they don't get to stabilize.

Table 4.2: Detailed DTSPNN with $\lambda = 5$

$T_{i,5}$	$p_{s'}(T_{i,5})$	$\frac{p_{s'}(T_{i,5})}{p_{s'}(T_{i-1,5})}$	$\overline{t}_s(T_{i,5})$	$\overline{v}_s(T_{i,5})$	$\overline{v}_{s'}(T_{i,5})$	$\overline{u}_{s'}(T_{i,5})$	$\overline{v}(T_{i,5})$	$\overline{u}(T_{i,5})$
50	0.7713	0.7713	3.41331	14.8229	242.351	10.1923	190.315	7.8613
350	0.6256	0.811098	78.9938	389.606	1744.42	10.1122	1237.18	6.3262
650	0.5131	0.820173	174.928	868.838	3247.66	10.2083	2089.41	5.2379
950	0.4217	0.821867	273.634	1363.08	4749.96	10.156	2791.33	4.2828
1250	0.3423	0.811714	372.679	1858.9	6252.28	10.2068	3362.75	3.4938
1550	0.2812	0.821502	459.904	2295.67	7754.15	10.1245	3830.59	2.847
1850	0.2268	0.806543	546.916	2731.06	9255.81	9.94709	4210.88	2.256
2150	0.1855	0.817901	620.453	3099.36	10759.2	10.3353	4520.27	1.9172
2450	0.15	0.808625	689.996	3447.39	12260.7	9.99667	4769.39	1.4995
2750	0.1209	0.806	752.83	3761.68	13764.5	10.1572	4971.02	1.228
3050	0.0967	0.799835	810.32	4049.27	15267.8	10.1655	5134.1	0.983
3350	0.079	0.81696	856.058	4278.72	16765.8	9.9962	5265.2	0.7897
3650	0.0648	0.820253	896.297	4480.21	18267.9	10.1296	5373.66	0.6564
3950	0.0535	0.825617	930.815	4652.97	19770.1	10.0187	5461.74	0.536
4250	0.044	0.82243	962.163	4809.79	21273.9	10.4636	5534.21	0.4604
4550	0.0376	0.854545	985.088	4924.53	22778.2	10.2766	5595.82	0.3864
4850	0.0317	0.843085	1007.67	5037.61	24280.4	10.429	5647.6	0.3306
5150	0.0249	0.785489	1035.54	5177.19	25783.4	10.0763	5690.29	0.2509
5450	0.0219	0.879518	1048.61	5242.74	27284	10.2648	5725.45	0.2248
5750	0.0172	0.785388	1070.34	5351.67	28782.8	10.0581	5754.69	0.173
6050	0.0137	0.796512	1087.51	5437.67	30282.9	10.4526	5778.05	0.1432
6350	0.0119	0.868613	1096.84	5484.39	31785.3	10.3025	5797.37	0.1226
6650	0.0099	0.831933	1107.8	5539.28	33286.9	10.4848	5813.99	0.1038
6950	0.0078	0.787879	1119.83	5599.53	34784	10.1026	5827.17	0.0788
7250	0.0062	0.794872	1129.53	5648.14	36281.9	9.96774	5838.07	0.0618
7550	0.0053	0.854839	1135.22	5676.68	37773.8	10.4151	5846.79	0.0552
7850	0.0042	0.792453	1142.45	5712.92	39263.3	11.0476	5853.83	0.0464
8150	0.0034	0.809524	1147.98	5740.55	40784.6	11.2647	5859.7	0.0383
8450	0.0029	0.852941	1151.54	5758.34	42311	11.3793	5864.34	0.033
8750	0.0021	0.724138	1157.49	5788.18	43784.1	10.381	5867.97	0.0218
9050	0.0019	0.904762	1159.04	5795.98	45258.8	9.57895	5870.96	0.0182
9350	0.0015	0.789474	1162.26	5812.21	46678.7	10.2667	5873.51	0.0154
9650	0.0011	0.733333	1165.59	5828.87	48157.5	11.1818	5875.43	0.0123
9950	0.0005	0.454545	1170.77	5854.68	49695.8	10.8	5876.6	0.0054
10550	0.0004	0.8	1171.68	5859.22	52722	12.75	5877.97	0.0051
10850	0.0003	0.75	1172.64	5864.01	54216.7	11	5878.51	0.0033
11150	0.0002	0.666667	1173.61	5868.89	55477	14	5878.81	0.0028
11450	0.0001	0.5	1174.61	5873.88	56893	11	5878.99	0.0011
12650	0	0	1175.75	5879.56	-	-	5879.56	0

Assuming that the rate iterations vanish is geometric between constant time intervals, we can predict, based on the present information, the time a simulation will first have a certain proportions of iterations that did not finish.

Suppose we choose a time interval $[t_l, t_r]$ where we know that $t_l > \overline{t}_*$ and $p_{s'}(t_l) \neq p_{s'}(t_r) > 0$; that is, we know that the time interval takes place after the number of unattended demands stabilizes, and the time interval is wide enough to guarantee that a reasonably number of iterations will vanish between t_l and time t_r. Then, we will be able to estimate the time at which the number of iterations that did not finish will arrive to a desired proportion p_f. Having chosen p_f -where $p_f < p_{s'}(t_r)$-, we can

estimate the time t_e at which only the proportion p_f of demands will be left in the system.

Since the proportion of iterations that did not finish follows a geometric progression, then for $n \in \mathbb{N}$ we have

$$p_f = p_{s'}(t_l)\left[\frac{p_{s'}(t_r)}{p_{s'}(t_l)}\right]^n \text{ ;thus, } \; n = \frac{\ln\left(\frac{p_f}{p_{s'}(t_l)}\right)}{\ln\left(\frac{p_{s'}(t_r)}{p_{s'}(t_l)}\right)}, \tag{4.1.1}$$

and the estimated time t_e at which the proportion p_f will occur is

$$t_e = t_l + n(t_r - t_l). \tag{4.1.2}$$

Consider Table 4.1 when $\lambda = 6$. Suppose we have run the simulation until time 6520 and chosen $t_l = 2230$ and $t_r = 6520$, and we want to estimate the time t_e that the proportion of iterations that did not stop is $p_f = 0.1$. Then, by equation 4.1.1 $n \approx 15.572$, and by equation 4.1.2 the time there will be 10% of the iteration running is $t_e \approx 69.037$, which is close to 67.845, the value from Table 4.1.

The precision of the prediction will depend on the fact that a significant number of iterations stopped between t_l and t_r; that is, by either choosing a large time interval or by increasing the number of iteration in a simulation. These considerations will let us have a clear snapshot of the progression of how iterations vanish. In our example we have not verified that $t_l > \bar{t}_*$ as 2230 seems a conservative election -enough time has passed to ensure that $\overline{u}_{s'}$ stabilizes-; however, if we could estimate $\bar{t}_*$, we would be able to make predictions with smaller values of t_l.

4.1.1 Simulated Annealing and First Local Maximum Estimation of $\overline{u}_*$ and $\bar{t}_*$

Table 4.1 shows that after some time the quantity $\overline{u}_{s'}$ stabilizes. For example, when $\lambda = 5$, $\overline{u}_{s'} \approx 10.15$ after time 192.8; however, it is not clear in what moment -between 3.8 and 192.8- this is likely to happen. Understanding how the number of unattended demands $u_{s'}$ evolves in every iteration and so understanding $\overline{u}_{s'}$ will let us know the time $\bar{t}_*$ that $\overline{u}_*$ occurs. We used two methods to estimate the values of $\overline{u}_*$ and $\bar{t}_*$: the first local maximum (FLM) and simulated annealing (SA).

We drafted FLM since, from Table 4.1, we observe that $\overline{u}_{s'}$ grows until it hits certain number of unattended demands, and it stays there until all the iterations finish. For every single iteration, FLM records the first time the number of unattended demands $u_{s'}$ is a local maximum u_*, and the time t_* that this happens. On the other hand, SA will decide with some probability if a local maximum of $u_{s'}$ will be used to estimate u_* and t_*. Both FLM and SA are heuristic and are used to estimate $\overline{u}_*$ and $\bar{t}_*$. However, FLM is deterministic, and SA is randomized.

Both algorithms are similar in the sense that in each iteration one look at the previous and current observation of $u_{s'}$ in order to decide whether the iteration has stabilized or not. After obtaining u_* and t_* for each iteration, they are averaged over the set of iterations to calculate $\overline{u}_*$ and $\bar{t}_*$. Note that if the time interval between the previous and the current observation of $u_{s'}$ is small, we might produce noisy observations with the risk of estimating not only a sub-local maximum u_* but also early t_*. On the other hand, if the algorithm uses large time intervals, it will estimate a proper u_* but with a larger than optimal t_*. Since it is not clear what would be a good choice of time the intervals, we will analyze all the iteration with respect to different fixed set of time intervals.

Let $c > 0$, $c_j = j\ c$, $\mathbf{T}_{j,i} = c_j\ i$, and $\mathcal{T}_j = \bigcup_{i=1}^{\infty} \mathbf{T}_{j,i}$ a set of increasing times where the observations of $u_{s'}$ and $t_{s'}$ take place. That is, the time interval of a set whose time

observations are $\mathbf{T}_{j,1}, \mathbf{T}_{j,2}, \ldots, \mathbf{T}_{j,i}, \ldots$ will be c_j. Let $\overline{u}_*^j$ be the mean estimated number of unattended demands the system stabilizes, and let $\overline{t}_*^j$ be the mean estimated time the system stabilizes when the set of time observations $\mathfrak{T}_j$ is used.

FLM starts estimating $\overline{u}_*^1$ and continue estimating $\overline{u}_*^2$, $\overline{u}_*^3 \ldots$ until $\overline{u}_*^j < \overline{u}_*^{j-1}$ for the first time at which point $\overline{u}_* = \overline{u}_*^{j-1}$ and $\overline{t}_* = \overline{t}_*^{j-1}$ [3]. The following steps show how FLM works.

1. Choose a positive constant c. Set $j = 1$ and $\overline{u}_*^0 = 0$.
2. For each iteration and at times $\mathfrak{T}_j$:
   (a) Find i such that $u_{s'}(\mathbf{T}_{j,i}) < u_{s'}(\mathbf{T}_{j,i-1})$ for the first time.
   (b) Store $u_{s'}(\mathbf{T}_{j,i})$ and $\mathbf{T}_{j,i}$.
3. From 2b, compute $\overline{u}_*^j$ and $\overline{t}_*^j$ by averaging $u_{s'}(\mathbf{T}_{j,i})$ and $\mathbf{T}_{j,i}$ over the number of iterations.
4. If $\overline{u}_*^j > \overline{u}_*^{j-1}$, set $j = j + 1$ and go to step 2. Otherwise, set both $\overline{u}_* = \overline{u}_*^{j-1}$ and $\overline{t}_* = \overline{t}_*^{j-1}$, and stop.

As FLM, SA estimates $\overline{u}_*^1, \overline{u}_*^2, \ldots$ until $\overline{u}_*^j < \overline{u}_*^{j-1}$ for the first time, at which point $\overline{u}_* = \overline{u}_*^{j-1}$ and $\overline{t}_* = \overline{t}_*^{j-1}$. However, assuming it is evaluating $\overline{u}_*^j$, in each iteration SA estimates u_* and t_* according to a probabilistic approach. In every time observation $u_{s'}(\mathbf{T}_{j,1}), u_{s'}(\mathbf{T}_{j,2}), \ldots, u_{s'}(\mathbf{T}_{j,i}), \ldots$ of an iteration, it performs one of the following actions using a random value $y_i \sim Unif(0, 1)$:

i If $y_i \leq e^{\left(u_{s'}(\mathbf{T}_{j,i}) - u_{s'}(\mathbf{T}_{j,i-1})\right)c_j}$ then continue evaluating the following time observation,

ii If $y_i > e^{\left(u_{s'}(\mathbf{T}_{j,i}) - u_{s'}(\mathbf{T}_{j,i-1})\right)c_j}$ then set $u_* = u_{s'}(\mathbf{T}_{j,i-1})$ and $t_* = \mathbf{T}_{j,i-1}$,

[3] We consider that $\overline{u}_*^0 = 0$

That is, the algorithm continues evaluating the number of unattended demands in the future until ii happens. In other words, the algorithm always stops in some local maximum since when $u_{s'}(\mathbf{T}_{j,i})$ is not a local maximum, then $u_j \leq 1 \leq e^{\left(u_{s'}(\mathbf{T}_{j,i}) - u_{s'}(\mathbf{T}_{j,i-1})\right)c_j}$ always. The following steps show the SA algorithm in more detail:

1. Choose a positive constant c. Set $j = 1$ and $\overline{u}_*^0 = 0$.

2. At times $\mathcal{T}_j$ and for each iteration:

   (a) Find i such that $y_i > e^{\left(u_{s'}(\mathbf{T}_{j,i}) - u_{s'}(\mathbf{T}_{j,i-1})\right)c_j}$ for the first time where $y_i \sim Unif(0,1)$.

   (b) Store $u_{s'}(\mathbf{T}_{j,i})$ and $\mathbf{T}_{j,i}$.

3. From 2b, compute $\overline{u}_*^j$ and $\overline{t}_*^j$ by averaging $u_{s'}(\mathbf{T}_{j,i})$ and $\mathbf{T}_{j,i}$ over the number of iterations.

4. If $\overline{u}_*^j > \overline{u}_*^{j-1}$, set $j = j + 1$ and go to step 2. Otherwise, set both $\overline{u}_* = \overline{u}_*^{j-1}$ and $\overline{t}_* = \overline{t}_*^{j-1}$, and stop.

If, in every iteration, SA stopped in the first local maximum, it would produce the same result as FLM. Table 4.3 shows the values and time the process stabilizes when using FLM and annealing with 10,000 iterations and $c = 0.5$.

Conclusion

Table 4.3 shows that the mean number of unattended demands the process stabilizes are similar in both methods, and they are slightly larger than the values inferred from Table 4.1. In those values of λ where both method stopped using the same set of time $\mathcal{T}_j$, SA has larger $\bar{t}_*$ than FLM since, in every iteration, the first time happens that $u_{s'}(\mathbf{T}_{j,i}) < u_{s'}(\mathbf{T}_{j,i-1})$, SA might continue seeking for further local maximum, but with no guarantees that the latter local maximum will be larger than the previous

Table 4.3: SA and FLM estimations of $\overline{u}_*$ and $\overline{t}_*$ for the DTSPNN

	FLM			SA		
λ	c_j	$\overline{t}_*$	$\overline{u}_*$	c_j	$\overline{t}_*$	$\overline{u}_*$
3	2	2.13864	3.2111	1.5	1.84435	3.05211
4	5.5	7.03477	6.12158	5.5	7.16386	6.01467
5	10.5	15.4948	10.4296	10.5	15.7191	10.3745
6	15.5	24.421	15.4029	15.5	24.7014	15.375
7	20	32.5016	21.0477	20.5	33.6766	21.0354
8	23.5	38.9475	27.4463	23.5	39.2969	27.4438
9	22	37.5826	34.5241	20.5	35.4853	34.4506
10	23.5	40.6322	42.3554	20	35.6598	42.1858
11	31	53.0776	51.2513	29	50.2628	51.2293
12	31	53.8058	60.7113	31	54.0755	60.7291
13	31.5	55.2856	70.9847	31.5	55.5496	71.0092
14	30.5	54.757	81.8908	30.5	54.9939	81.9166
15	36	63.7423	94.0354	36	63.9778	94.0566
16	38	67.7217	106.838	38	67.9873	106.867
17	38.5	69.2284	120.269	38.5	69.454	120.295
18	45	79.8462	135.057	45	80.0955	135.085
19	42.5	77.1894	149.925	42.5	77.4431	149.961
20	40.5	75.4114	165.457	40.5	75.6038	165.489
21	47.5	86.6533	182.749	47.5	86.8894	182.788
22	45	84.181	199.963	45	84.3741	200.001
23	44	84.2978	217.938	44	84.4928	217.976
24	45	87.0444	237.203	58	105.272	238.855
25	58	106.146	258.828	58	106.359	258.865

one. On the other hand, when λ is small FLM, gives slightly larger values of $\overline{u}_*$ than SA, but the opposite occurs when λ is large. Figure 4.1 shows the regression fit and the residual sum of squares (RSS) of the fit for the FLM using the values of Table 4.3. Similarly, Figure 4.2 shows the regression fit and the RSS of SA. The time the system stabilizes fits a linear function of λ, and the mean number of unattended demands when the system stabilizes fits an approximately quadratic function of λ.

The FLM regression fit: $\overline{u}_*(\lambda) = 0.3870\lambda^{2.0275}$ with $RSS = 135.82$ and $\overline{t}_*(\lambda) = 4.0306\lambda - 0.0031$ with $RSS = 705.72$.

The SA regression fit: $\overline{u}_*(\lambda) = 0.3734\lambda^{2.0404}$ with $RSS = 186.69$ and $\overline{t}_*(\lambda) = 4.2478\lambda - 2.4963$ with $RSS = 635.47$.

Figure 4.1: FLM regression fit for the DTSPNN

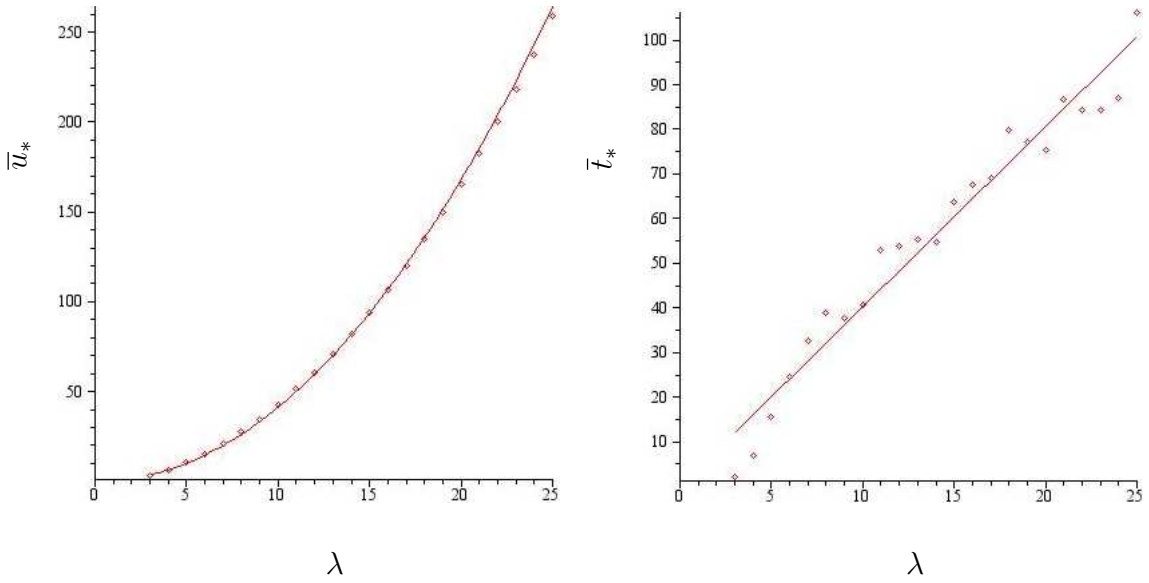
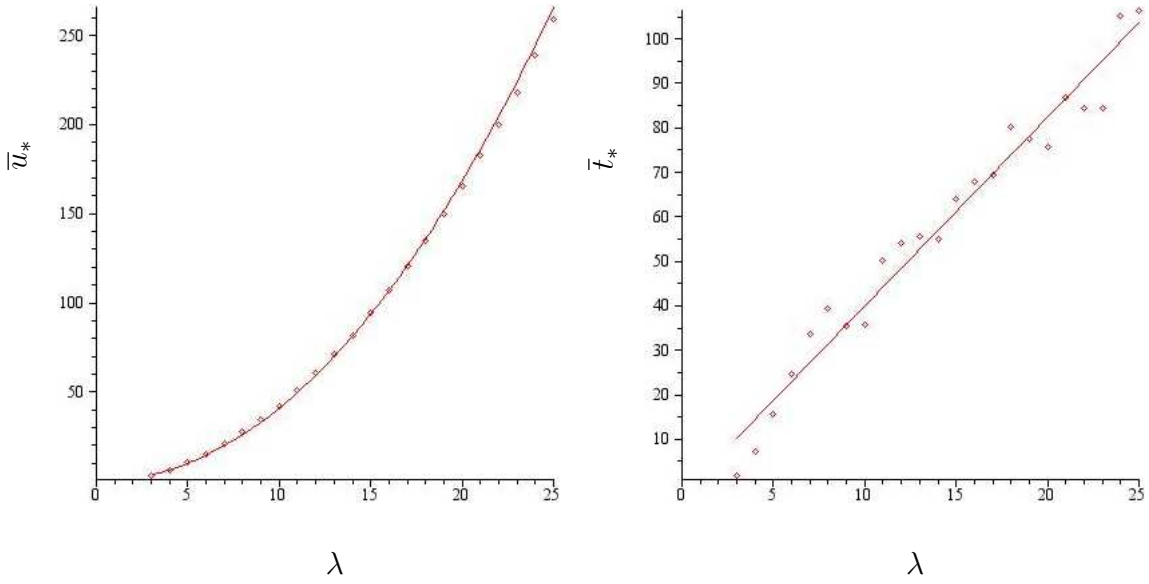


Figure 4.2: SA regression fit for the DTSPNN



Remarks We tried a variation of the previous algorithms. Instead of evaluating all the iterations with respect to a set of times $\mathcal{T}_j$, we decided to compare each iteration with respect to the sequence of time $\mathcal{T}_1, \mathcal{T}_2, \ldots$ to find a sequence of local maximum $u_*^1, u_*^2, \ldots$. That is, when in every iteration it is found for the first time that $u_*^j < u_*^{j-1}$ - or $y_i > e^{(u_*^j - u_*^{j-1})c_j}$ in the case of SA-, the values u_*^{j-1} and its time u_*^{j-1} are used to obtain the aggregated means $\overline{u}_*$ and $\overline{t}_*$. The steps for the variation of the FLM

algorithm are [4]:

1. Choose a positive constant c and set $k = 1$.

2. Consider the k^{th} iteration, and set $j = 0$ and $u_*^0 = 0$.

3. Set $j = j + 1$ and applying time interval $\mathcal{T}_j$:

   (a) Find i such that $u_{s'}(\mathbf{T}_{j,i}) < u_{s'}(\mathbf{T}_{j,i-1})$ for the first time.

   (b) Set $u_*^j = u_{s'}(\mathbf{T}_{j,i-1})$.

   (c) If $u_*^j > u_*^{j-1}$ jump to step 3.

4. Store u_*^{j-1} and t_*^{j-1}.

5. If there are iterations left, jump to step 2.

6. From step 4, calculate $\overline{u}_*$ and $\overline{t}_*$ by averaging u_*^{j-1} and t_*^{j-1} over the number of iterations.

Compared to Table 4.3, the results obtained by this algorithm underestimates the values of $\overline{u}_*$ for $\lambda = 3, \ldots, 25$. The number of unattended demands of a single iteration has too many fluctuations, so it will have local maximums with high frequency throughout its entire cycle. Thus, the algorithm will find local maximum in early stages of its cycle that will be suboptimal, and so, in most of the iterations, the algorithm will generate a suboptimal sequence $u_*^1, u_*^2, \ldots$ from which it will choose the local maximum of an iteration.

[4] Replacing step 3a by

Find i such that $y_i > e^{\left(u_{s'}(\mathbf{T}_{j,i}) - u_{s'}(\mathbf{T}_{j,i-1})\right)c_j}$ for the first time where $y_i \sim Unif(0,1)$.

would produce the SA version.

4.1.2 Ergodic Estimations of $\overline{u}_*$ and $\bar{t}_*$

After a short period of time, the iterations of the DTSPNN hit $\overline{u}_*$, and remain around this value for a long time until they vanish. That is, we are in presence of a quasi-stationary distribution since iterations appear to be stationary over a reasonable time scale before they vanish. Since the limiting state of the system is to "die-out", it does not provide information of what is the stationary state during the existence of the system. Thus, we cannot apply the ergodic theorem, but we can use a similar approach to estimate the quasi-stationary distribution. The approach is a conditional version of the ergodic theorem.

By sampling the number of unattended demands of each iteration that did not vanish at different time intervals, we can estimate $\overline{u}_*$. Let $\mathfrak{T}_j = \bigcup_{i=1}^{T_j} \mathbf{T}_{j,i}$ be the finite set of increasing times in which observations of the number of unattended demands take place for each iteration. The set $\mathfrak{T}_j$ is defined as in FLM and SA; however, its size is bounded by $T_j = \lfloor \frac{T_\lambda}{c_j} \rfloor$.

Let K be the number of iterations, and $u^k_{s'}(\mathbf{T}_{j,i})$ be the number of unattended demands of iteration k at time $\mathbf{T}_{j,i}$. Then, the mean number of unattended demands $\overline{u}^j_*$ of all the iterations that did not vanish at time T_j when observations take place according to $\mathfrak{T}_j$ is

$$\overline{u}^j_* = \frac{\displaystyle\sum_{k=1}^{K} \left(\left(\sum_{i=1}^{T_j} u^k_{s'}(\mathbf{T}_{j,i}) \right) \frac{\mathrm{I}_{\{u^k_{s'}(\mathbf{T}_{j,T_j})>0\}}}{T_j} \right)}{\displaystyle\sum_{k=1}^{K} \mathrm{I}_{\{u^k_{s'}(\mathbf{T}_{j,T_j})>0\}}} = \frac{\displaystyle\sum_{k=1}^{K} \overline{u}^k_j}{\displaystyle\sum_{k=1}^{K} \mathrm{I}_{\{u^k_{s'}(\mathbf{T}_{j,T_j})>0\}}}.$$

The expression $\overline{u}^k_j$ is the average number of unattended demands of an iteration k when observations are performed at times $\mathfrak{T}_j$. If at the time of the last observation - the T^{th}_j observation- an iteration has vanished then $\overline{u}^k_j$ is set to 0. Thus, the expression on the right is the average number of unattended demands of all the iterations that

did not vanish at time $\mathbf{T}_{j,T_j}$.

If λ is large, a single iteration lasts a long time before it vanishes, producing computationally expensive estimations of $\overline{u}_{s'}$, so we have chosen to limit the observation of the number of unattended demands until time T_λ. When λ is small, iterations vanish after a short period of time. Thus, T_λ should not be large, so we can avoid the situation of having a high proportion of rejected iterations or $u^k_{s'}(\mathbf{T}_{j,T_j}) = 0$. The set of increasing times $\mathcal{T}_j$ will be defined by c and T_λ and chosen according to λ. The larger the value of λ the larger the values of both c and T_λ. For a given λ, the following steps explain how $\overline{u}_*$ is obtained. Then, we will introduce the algorithm used to estimate $\bar{t}_*$.

1. Define c and T_λ. Set $j = 1$ and $\overline{u}^0_* = 0$.

2. Evaluate $\overline{u}^j_*$.

3. If $\overline{u}^j_* > \overline{u}^{j-1}_*$, set $j = j + 1$ and go to step 2. Otherwise, set $\overline{u}_* = \overline{u}^j_*$ and stop.

The estimations of $\overline{u}^j_*$ will grow along with j until a local maximum is found. The larger the value of j, the smaller the set of the maximum number of observations $\mathcal{T}_j$ per iteration. For small values of λ, the estimation of $\overline{u}_*$ should be more sensitive to the increments of j since iterations are short lived and early observations in the number of unattended demands -that are below the ideal $\overline{u}_*$- have more weight on the estimation of $\overline{u}^j_*$ than when iterations live longer. Thus, we do not expect that j will grow according to λ.

After the value of $\overline{u}_*$ is known, it is possible to estimate $\bar{t}_*$ using the algorithm below. For every iteration, we evaluate the mean time the number of unattended demands is grater or equal than $\dot{u}_*$, the closest natural number to $\overline{u}_*$. We have decided to use $\dot{u}_*$ rather than $\overline{u}_*$ since the number of unattended demands is a natural number. Note that $\overline{u}_*$ is equal to $\overline{u}^{j-1}_*$ rather than $\overline{u}^j_*$ (step 3), where $\overline{u}^j_*$ is slightly smaller that $\overline{u}^{j-1}_*$.

The value $\bar{t}_*$ is obtained by averaging each time t when $u_{s'}(t) \geq \dot{u}_*$ first happens (step 3a) in every iteration. Since most of the time there is a high chance that $u_{s'}(t) > \dot{u}_*$, we decided to alleviate the effect of the inequality by assigning the value of $\overline{u}_*^{j-1}$ over $\overline{u}_*^{j}$ to $\overline{u}_*$ and hence $\dot{u}_*$. However, since we are using $\dot{u}_*$ in the comparison, the election of either $\overline{u}_*^{j}$ and $\overline{u}_*^{j-1}$ might be irrelevant in the majority of the iterations. Finally, along with $\bar{t}_*$, we evaluated the mean number of unattended demands $\tilde{u}_*$ obtained at the time the number of unattended demands of each iteration surpasses $\dot{u}_*$ for the first time.

1. Set $\dot{u}_* = [\overline{u}_*]$.
2. Consider the set of observations $\mathcal{T}_1$ used on the estimation of $\overline{u}_*$.
3. For each iteration:
   (a) Find i such that $u_{s'}(\mathbf{T}_{1,i}) > \dot{u}_*$ for the first time.
   (b) If i is found, store $\mathbf{T}_{1,i}$ and $u_{s'}(\mathbf{T}_{1,i})$.
4. From 3b, compute $\bar{t}_*$ and $\tilde{u}_*$ by averaging $\mathbf{T}_{1,i}$ and $u_{s'}(\mathbf{T}_{1,i})$ over the iterations in which i was found.

Table 4.4 shows the result of the method. The number of iterations used in the simulation is denoted by K, and the number of iterations that were not rejected is K_*. The value of c is increased along with λ, so we can observe each iteration for longer time without increasing the number of observations.

Conclusion

Table 4.4 shows that for every λ, the ergodic approach has values of $\overline{u}_*$ that are both smaller than the values obtained with the FLM and SA methods and closer to the values of Table 4.1. Thus, the values of $\bar{t}_*$ are also smaller in Table 4.4 than in the previous methods. Note that, $\tilde{u}_*$ is significantly larger than $\overline{u}_*$, which means $\bar{t}_*$ is still

Table 4.4: Ergodic estimation of $\overline{u}_*$ and $\bar{t}_*$ for the DTSPNN

λ	K	K_*	N	c	T_λ	j	c_j	$\bar{t}_*$	$\tilde{u}_*$	$\overline{u}_*$	$\dot{u}_*$
3	40000	4684	1000	0.03	30	21	0.63	2.63272	6.46458	4.51929	5
4	10000	3734	1000	0.075	75	16	1.2	4.07845	8.55434	6.80603	7
5	10000	5461	1000	0.5	500	8	4	5.98595	11.6805	10.1792	10
6	10000	7960	1000	2	2000	6	12	9.73513	16.7084	14.3674	14
7	10000	8744	10000	2	20000	3	6	11.8124	22.0180	19.3308	19
8	10000	9176	10000	3	30000	3	9	15.7674	28.6725	25.0636	25
9	10000	9347	10000	3	30000	6	18	18.7231	35.9848	31.5635	32
10	10000	9463	10000	3	30000	5	15	20.2378	43.3329	38.8488	39
11	10000	9544	10000	3	30000	4	12	22.4479	51.6298	46.9094	47
12	10000	9617	10000	3	30000	4	12	24.7289	60.9746	55.7585	56
13	10000	9667	10000	3	30000	4	12	26.1611	70.3170	65.3911	65
14	10000	9710	10000	3	30000	10	30	28.9392	81.6799	75.8186	76
15	10000	9746	10000	3	30000	7	21	31.0769	92.9460	87.0335	87
16	10000	9779	10000	3	30000	7	21	33.1269	105.274	99.0412	99
17	10000	9804	10000	3	30000	9	27	35.9438	118.607	111.853	112
18	10000	9826	10000	3	30000	9	27	36.9898	131.999	125.440	125
19	10000	9841	10000	3	30000	6	18	40.3235	147.415	139.823	140
20	10000	9858	10000	3	30000	7	21	42.4098	162.690	155.018	155
21	10000	9868	10000	3	30000	9	27	45.1280	179.030	171.013	171
22	10000	9879	10000	3	30000	9	27	47.4670	196.433	187.802	188
23	10000	9894	10000	3	30000	13	39	48.7276	213.677	205.399	205
24	10000	9900	10000	3	30000	6	18	52.2157	233.083	223.754	224
25	10000	9908	10000	3	30000	10	30	53.5575	252.646	242.958	243

larger than the optimal. However, the ergodic approach is an improvement over the FLM and SA in the estimation of $\overline{u}_*$ and consequently of $\bar{t}_*$. This method is more computationally expensive than the FLM and SA. For example, if $\lambda = 10$, the average time iterations stopped in SA is $\bar{t}_* = 40.6322$, whereas in the ergodic approach the mean time iterations stopped is between $(T_\lambda \frac{K_*}{K}, T_\lambda) = (28,389, 30,000)$ units of time.

The quadratic regression of the mean number of untended demands and the time it stabilizes is $\overline{u}_* = 0.468\lambda^{1.9324}$ and $RSS = 166.3095$. The linear regression of the mean time the number of unattended demands stabilizes is $\bar{t}_* = 2.3304\lambda - 4.0073$ and $RSS = 13.8337$.

4.2 The DTSP with Random Start Policy

The DTSP with random start (DTSPR) works as the DTSPNN defined in Section 3.2, but after serving a demand, the server starts in a random location uniformly distributed in the unit square. The distance from every visited point to the new random location does not directly affect the process. In other words,

- The distance from every visited demand to the new random location is not included in the length of the path L_n. The length of the path is the sum of the distances from a random location to its closest demand.
- No new demands are generated in the trajectory from every visited demand to the new random location.
- New demands are generated in the time the server travels from its location -after being randomly located- to its closest demand.

Table 4.5 shows the mean state of the simulation when at intervals of times in which $1 - \overline{p}_{\lambda}^{\overline{s}} \approx \frac{i}{10}, i = 2, \ldots, 10$. Each simulation for each λ consists of 10,000 iterations.

Table 4.5: DTSPR with $\lambda = 3, \ldots, 7$

λ	$T_{i,\lambda}$	$p_{s'}(T_{i,\lambda})$	$\bar{t}_s(T_{i,\lambda})$	$\overline{v}_s(T_{i,\lambda})$	$\overline{v}_{s'}(T_{i,\lambda})$	$\overline{u}_{s'}(T_{i,\lambda})$	$\overline{v}(T_{i,\lambda})$	$\overline{u}(T_{i,\lambda})$
3	0.27	0.8902	0.161652	1.01002	1.07785	1.78117	1.0704	1.5856
	0.48	0.7884	0.2618	1.04442	1.33638	2.08473	1.2746	1.6436
	0.78	0.6996	0.366263	1.14614	1.992	2.48999	1.7379	1.742
	1.56	0.5997	0.548835	1.48039	3.98583	3.15408	2.9829	1.8915
	3.48	0.4992	0.915457	2.34225	9.6254	3.89744	5.978	1.9456
	7.14	0.4	1.62532	4.28333	21.2557	4.24225	11.0723	1.6969
	12.51	0.2996	2.79419	7.76299	38.6355	4.31308	17.0124	1.2922
	19.8	0.1999	4.41926	12.8268	62.2951	4.29515	22.7155	0.8586
	32.58	0.1	6.73568	20.1634	104.072	4.19	28.5543	0.419
	163.95	0	11.3218	34.9699	-	-	34.9699	0
4	0.3	0.8921	0.172887	1.02132	1.12297	2.43022	1.112	2.168
	0.65	0.7978	0.302018	1.10237	1.74668	3.0712	1.6164	2.4502
	2.8	0.6992	0.627922	1.81283	8.43435	5.32108	6.4426	3.7205
	17.8	0.6	2.72096	9.1055	68.0432	6.78617	44.4681	4.0717
	41.6	0.5	8.05909	29.8788	165.089	6.8222	97.4839	3.4111
	69.45	0.3999	15.8553	61.0533	278.269	6.74794	147.918	2.6985
	104.6	0.3	25.8728	101.306	421.398	6.742	197.334	2.0226
	152.9	0.2	38.5609	152.555	618.248	6.8605	245.694	1.3721
	236.8	0.1	55.4197	220.986	959.746	6.783	294.862	0.6783
	1114.05	0	85.4416	342.98	-	-	342.98	0
5	2	0.8	0.461732	1.5045	6.22737	6.14925	5.2828	4.9194
	161	0.6999	23.7835	115.115	799.087	10.2303	593.827	7.1602
	376	0.5996	84.3432	416.52	1875.98	10.1503	1291.61	6.0861
	627	0.5	166.831	828.763	3133.05	10.1982	1980.9	5.0991
	932	0.3999	267.723	1333.29	4661.11	10.2001	2664.08	4.079
	1320	0.2998	388.334	1936.66	6607.42	10.2328	3336.95	3.0678
	1889	0.2	537.365	2683.6	9457.46	10.207	4038.38	2.0414
	2874	0.1	736.67	3682.17	14397.7	10.25	4753.72	1.025
	11659	0	1088.53	5445.83	-	-	5445.83	0
6	40	0.8527	0.668841	2.83775	227.604	14.2397	194.496	12.1422
	1760	0.7986	237.441	1418.43	10552	14.3147	8712.52	11.4317
	5320	0.6999	1307.82	7840.5	31920.2	14.3843	24693.9	10.0676
	9520	0.5994	2823.1	16933.1	57129	14.3091	41026.5	8.5769
	14480	0.4994	4640.89	27841.9	86898.1	14.2559	57334.6	7.1194
	20280	0.4	6742.13	40450.6	121712	14.2705	72955.1	5.7082
	27960	0.2998	9197.6	55189.2	167806	14.3386	88951.6	4.2987
	39520	0.1998	12233.7	73411.1	237185	14.485	106133	2.8941
	57760	0.0998	16186.7	97137.4	346643	14.3978	122038	1.4369
	308920	0	23207.7	139274	-	-	139274	0
7	2000	0.891	17.5782	121.881	13986.4	19.3471	12475.2	17.2383
	106000	0.7987	24216.8	169544	742138	19.309	626874	15.4221
	228000	0.6999	71227.7	498669	1.59634e+6	19.3659	1.26693e+6	13.5542
	382000	0.5987	129928	909672	2.67455e+6	19.3708	1.96630e+6	11.5973
	546000	0.4997	195982	1.37215e+6	3.82281e+6	19.4036	2.59675e+6	9.696
	742000	0.3989	271274	1.89931e+6	5.19509e+6	19.3204	3.21400e+6	7.7069
	1.012e+6	0.2996	356279	2.49448e+6	7.08550e+6	19.2079	3.86995e+6	5.7547
	1.408e+6	0.1997	460942	3.22728e+6	9.85813e+6	19.1998	4.55146e+6	3.8342
	2.042e+6	0.0999	597063	4.18033e+6	1.42971e+7	18.9409	5.19100e+6	1.8922
	8.544e+6	0	838161	5.86838e+6	-	-	5.86838e+6	0

The values between Table 4.1 and Table 4.5 seem to be close. We will use the ergodic estimation to have a closer look at the parameters $\overline{u}_*$ when $\lambda = 3, \ldots, 25$. In order to reduce the computational time of the estimations, we reduced the number of iterations used with respect to Table 4.4.

Table 4.6: Ergodic estimation of $\overline{u}_*$ and $\bar{t}_*$ for the DTSPR

λ	K	K_*	N	c	T_λ	j	c_j	$\bar{t}_*$	$\tilde{u}_*$	$\overline{u}_*$	$\dot{u}_*$
3	40000	4570	1000	0.03	30	9	0.27	2.64722	6.46219	4.53485	5
4	10000	3824	1000	0.075	75	16	1.2	4.0533	8.52162	6.81811	7
5	10000	5478	1000	0.5	500	8	4	5.93552	11.6697	10.1721	10
6	8000	3901	10000	1.5	15000	6	9	9.14308	16.4557	14.3714	14
7	8000	6986	10000	1.5	15000	3	4.5	11.1426	21.7039	19.3313	19
8	8000	7307	10000	1.5	15000	4	6	13.4028	27.866	25.0607	25
9	8000	7450	10000	1.5	15000	8	12	16.4146	35.0852	31.5679	32
10	8000	7543	10000	2.5	25000	4	10	19.5861	43.0756	38.8437	39
11	8000	7622	10000	2.5	25000	8	20	21.5265	51.3381	46.9019	47
12	5000	4806	10000	2.5	25000	10	25	24.1466	60.6764	55.7633	56
13	5000	4836	10000	2.5	25000	9	22.5	25.4392	69.8786	65.3909	65
14	5000	4855	10000	2.5	25000	6	15	28.1963	81.1371	75.8144	76
15	5000	4869	10000	3	30000	6	18	31.5275	92.9358	87.0305	87
16	5000	4880	10000	3	30000	7	21	33.2566	105.251	99.0395	99
17	5000	4897	10000	3	30000	4	12	35.7557	118.619	111.844	112
18	5000	4911	10000	3	30000	6	18	37.113	131.938	125.442	125
19	5000	4921	10000	3	30000	5	15	40.2574	147.269	139.828	140
20	5000	4925	10000	3	30000	8	24	42.2477	162.769	155.024	155
21	5000	4932	10000	3	30000	7	21	44.4779	179.145	171.005	171
22	5000	4940	10000	3	30000	8	24	47.6093	196.437	187.800	188
23	5000	4947	10000	3	30000	12	36	48.7583	213.622	205.394	205
24	5000	4953	10000	3	30000	7	21	51.7698	233.122	223.780	224
25	5000	4955	10000	3	30000	10	30	54.2891	252.562	242.979	243

Conclusion

If we compare the values from Table 4.4 and Table 4.6, that is, if we compare the estimated $\overline{u}_*$ using ergodic approach in both the DTSPNN and the DTSPR, we can see that both algorithms perform similarly. This invariance of the performance by the DTSPR leads us to consider the partition policy.

4.3 The DTSP with Delayed Random Start Policy

We will call the DTSP with delayed random start (DTSPDR) the policy obtained by modifying the DTSPR policy as follows: new demands are generated in the trajectory from every visited demand to the new random location and, as in the DTSPNN, during the time that the server takes to travel from a random location to its closest demand. After a demand is served, there is a waiting time from which the server does not serve new demands since it is heading towards the random location; however, new demands might be created during this time. The mean time that takes the server to move from the served demand to the new random location can be consider as the DTRP's mean service time $\overline{s}$ of demands. From Section 3.1, we know that given two uniformly and independently distributed points X_1 and X_2 in a square of area A, then $E\|X_1 - X_2\|^2 = \frac{1}{3}\sqrt{A}$ and $E\|X_1 - X_2\| \approx 0.52\sqrt{A}$. Since we can considered X_1 and X_2 as the location of the last served demand and the later random location of the server respectively then the service first moment is $\overline{s} \approx 0.52$, and the service second moment is $\overline{s^2} = \frac{1}{3}$. The DTSPDR behaves as the DTRP with NN policy, and the system will eventually sweeps all the points with probability 1 only if $\rho = 0.52\lambda < 1$ [5]. This policy in general does not stabilize over time where the number of unattended demands grows to infinity.

4.4 The DTSP with Partitioning Policy

The DTSP with partitioning (DTSPP) is another modification of the DTSPNN. As in the previous problems, the server will generate a path $L_t = L(x_{\sigma(1)}^{p_1,q_1}, x_{\sigma(2)}^{p_2,q_2}, \ldots)$, that connects at a constant unit velocity demands created according to a Poisson random variable $Z_\lambda(t)$ with mean λ and uniformly distributed in the unit square; the process

[5] When $\rho \geq 1$, there is a chance that the DTSPDR is stable since the system can still arrive to $u_{s'} = 0$, and once this happens the system stops. On the other hand, when the DTRP with NN of Section 3.1 arrives to $u_{s'}$, it waits for new demands to arrive.

also stops when either there are no more points to visit or $|L_t| \geq T_\lambda$. However, in the DTSPP, we split the unit square into a set of P^2 disjoint squares of area $\frac{1}{P^2}$. Let $x^{p,q}_{\sigma(t)}$ be the t^{th} served demand in partition $p, q \in P$. After the server attends a demand, the election of the next demand to be served will depend on the evaluation of c_t rather than the closest demand d_t as in the DTSPNN case. We will briefly explain the idea behind c_t.

Starting from the position of the last demand, the algorithm performs the following steps in each partition $p', q' \in P$ of the unit square:

a Calculate the distance from the server to the closest demand of the partition.

b Starting from the closest demand of the partition to the server, calculates the length of the path that sweeps all the demands of the partition using the NN algorithm.

c Divide the sum of distances obtained in (a) and (b) by the number of demands visited in (a) and (b).

The closest demand to the server located in the partition with the minimum value in step(c) is the demand that the server will actually visit; the minimum value will be denoted $c_t = \min\{c_t^{p',q'}\} \ \forall \ p', q' \in P$. Note that once c_t is found, it is used to serve the t^{th} the closest demand from the server, and after the t^{th} demand is served c_{t+1} is evaluated in the same manner to decide which is the next demand -and thus which partition the server will visit next-. The idea behind using c_t rather than d_t is to force the server to move to areas with "high density" of untended demands though the decision might involve visiting demands that are not the nearest to the server. We are interested in the behaviour of the DTSPP when P is rather small. When $P \to \infty$, DTSPP will behave as the DTSPNN since, with high probability, each partition will have at most one demand, that is, for most of the time the length in step (b) will be zero and $c_t = d_t$. A more detailed explanation in the evaluation

of c_t is as follows. Assuming the server has served demand $x^{p,q}_{\sigma(t-1)}$, we will introduce the following notations:

- The set of unattended demands in a partition p', q' is defined $U^{p',q'}(t-1)$ and its size by $|U^{p',q'}(t-1)|$.
- The server's closest unattended demand in a partition p', q' is $\dot{x}^{p',q'}_{\sigma(t)}$[6].
- The length of the path $L_{\left\{U^{p',q'}(t-1)\right\}}\left(\dot{x}^{p',q'}_{\sigma(t-1)}\right)$ generated by starting from $\dot{x}^{p',q'}_{\sigma(t-1)}$ and visiting all the untended demands $U^{p',q'}(t-1)$ under the NN policy.

Then,

$$c_t^{p',q'} = \frac{\|x^{p,q}_{\sigma(t-1)} - \dot{x}^{p',q'}_{\sigma(t)}\|_2 + L_{\left\{U^{p',q'}(t-1)\right\}}\left(\dot{x}^{p',q'}_{\sigma(t-1)}\right)}{|U^{p',q'}(t-1)|},$$

is the average distance between the path obtained by visiting all the demands in partition p', q' using the NN policy and starting from $x^{p,q}_{\sigma(t-1)}$ over the number of nodes in partition p', q'.

Finally

$$c_t = \min\left\{c_t^{p',q'}\right\}, \forall\ p', q' \in P.$$

The DTSPP can be described as follows:

1. Start with $t = 2$ and two random demands $x_1^{p,q}$ and $x_2^{p',q'}$ where $x_1^{p,q}$ is the starting position, so $L_2 = L(x_{\sigma(1)}, x_{\sigma(2)}) = L(x_1^{p,q}, x_2^{p',q'})$.
2. Generate $Z_\lambda(d_t)$ demands where $d_t = \|x^{p',q'}_{\sigma(t-1)} - x^{p,q}_{\sigma(t)}\|$.
3. Evaluate c_t. Consider the partition p^*, q^* for which $c_t = c_t^{p^*,q^*}$ and visit $\dot{x}^{p^*,q^*}_{\sigma(t+1)}$.
4. If either

$$\bigcup_{k=2}^{t} \theta_{(d_k)} \setminus \{x_{\sigma(1)}, \ldots, x_{\sigma(t+1)}\} \neq 0,$$

[6]If $U^{p',q'}(t-1) \neq \emptyset$

or

$$|L_t| = \sum_{k=2}^{t} d_k < T_\lambda,$$

set $t = t + 1$ and go back to step 2. Otherwise, stop.

Table 4.7: DTSPP with $\lambda = 3, 4, 5, 6$ and different values of P

(a) $\lambda = 3$

P	T_λ	$\bar{t}$	$\overline{v}$
1	150.83	11.4618	35.4313
2	250	13.3341	40.9301
4	250	12.9576	39.881
6	200	12.3051	37.9046
8	200	12.0114	37.0938
10	200	11.6709	35.9908
12	200	11.6424	35.9211
14	200	11.714	36.1038
16	200	11.5191	35.4576
18	200	11.5615	35.6516
20	150	11.601	35.7473
22	150	11.5009	35.474
24	200	11.5668	35.6605
26	250	11.4513	35.2965
28	200	11.3886	35.0859
30	200	11.4675	35.3437
32	200	11.384	35.0672
34	150	11.4667	35.3176
36	200	11.4944	35.4528
38	200	11.526	35.5474
40	200	11.4865	35.4224
50	200	11.4727	35.3763
60	200	11.4991	35.4683
1000	150	11.4798	35.417

(b) $\lambda = 4$

P	T_λ	$\bar{t}$	$\overline{v}$
1	1391	86.5959	347.283
2	1712	146.684	588.018
4	1692	128.023	512.931
6	1338	111.366	446.215
8	1290	99.516	399.175
10	1492	93.319	374.304
12	1618	91.639	367.589
14	1070	89.548	358.901
16	1422	89.637	359.547
18	1040	87.506	351.137
20	1046	86.436	346.851
22	948	87.251	349.927
24	1078	87.041	349.090
26	1056	84.600	339.554
28	1170	84.628	339.598
30	1324	84.721	339.720
32	1042	84.657	339.607
34	1254	84.140	337.309
36	1080	86.265	346.182
38	1160	83.475	334.747
40	1036	83.933	336.800
50	1228	83.508	334.980
60	1212	84.689	339.946
1000	1031	82.397	330.591

(c) $\lambda = 5$

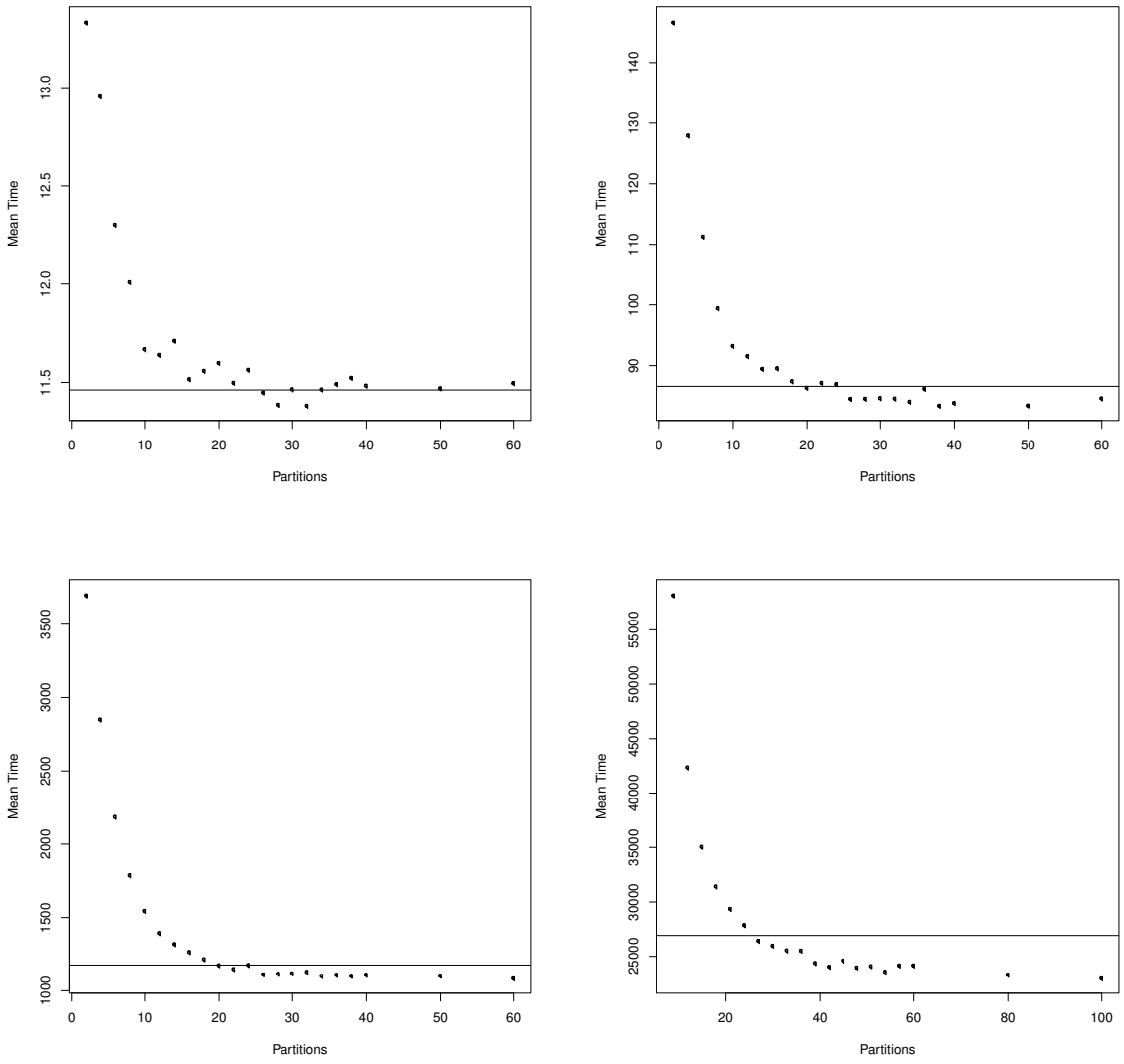
P	T_λ	$\bar{t}$	$\overline{v}$
1	12593	1175.75	5879.56
2	40420	3700.46	18508.6
4	33580	2853.70	14274.1
6	36960	2189.04	10949.8
8	24320	1790.82	8957.48
10	23465	1548.19	7744.07
12	17385	1397.16	6988.60
14	16020	1321.74	6611.11
16	12540	1267.76	6341.92
18	15855	1218.89	6098.25
20	11880	1178.00	5891.64
22	14125	1151.74	5759.24
24	13700	1179.22	5898.23
26	16570	1115.09	5577.22
28	13050	1118.55	5594.39
30	11275	1122.29	5613.86
32	13435	1132.04	5662.76
34	12250	1104.68	5524.78
36	10880	1111.89	5561.31
38	11055	1105.01	5527.52
40	11145	1112.19	5563.82
50	11230	1106.29	5533.71
60	11480	1087.70	5441.27

(d) $\lambda = 6$

P	T_λ	$\bar{t}$	$\overline{v}$
1	446080	26920.7	161493
3	-	-	-
6	-	-	-
9	637500	58212.3	349360
12	448500	42415.6	254553
15	427500	35100.5	210655
18	414000	31462.3	188817
21	328500	29405.4	176472
24	345000	27916.8	167535
27	373500	26475.0	158883
30	267000	26025.3	156180
33	246000	25593.3	153598
36	294000	25564.7	153420
39	241500	24429.7	146611
42	256500	24091.5	144581
45	234000	24665.8	148023
48	336000	24016.4	144128
51	238500	24142.8	144887
54	253500	23627.0	141791
57	237000	24204.7	145263
60	303000	24214.3	145315
80	238500	23347.9	140115
100	247500	23014.1	138111

Table 4.7 shows the mean time at which all iterations vanished and the mean number of demands visited for the DTSPP simulations when different number of partitions are used - including the results from Table 4.1 when $P = 1$ [7]. Figure 4.3 shows the results of Table 4.7, where the vertical line represents the mean time the iteration vanish when $P = 1$.

Figure 4.3: DTSPP with $\lambda = 3, 4, 5, 6$ and different values of P



[7] When $\lambda = 6$ and $P = 3, 6$, there where iterations that did not vanished at the time of the last observation $T_\lambda = 500,000$, so these results were not included in the table.

When P is small, the mean time $\overline{t}$ at which the iterations vanish is larger than when $P = 1$, and as P increases, the values of $\overline{t}$ decreases, until $\overline{t}$ get smaller than is for $P = 1$. There is an improvement in the DTSPP if our objective is to reduce the mean time the iterations will vanish; this improvement is more apparent when λ increase as there is more chances for partitions to have more than one demand.

If we calculate, among those simulations where all the iterations vanished, the ratio between the mean number of nodes visited and the mean time at which iterations vanish, it remains close to λ. That is,

$$\frac{\overline{v}}{\overline{t}} \approx \lambda, \text{ for every } P \text{ and } \lambda;$$

thus, regardless of the value of P, the DTSPP visits demands at the same rate as the DTSPNN [8] though for some values of P the DTSPP finds conditions for which all the demands can be swept from the unit square faster than the DTSPNN.

[8] Otherwise, the number of unattended nodes would explode if DTSPP visited nodes at a lower rate than the arrival rate of demands, or iterations would be short lived if the DTSPP visited nodes at a higher rate than the arrival rate of demands.

Chapter 5

Conclusion

The NN policy was first used for the DTSP as it has the same performance order as the optimal lower bound and slightly more efficient than the SFC policy for the DTRP, a closely related problem to the DTSP. The DTSPNN could be modelled using Markov chains since, once the server visits a demand, the next demand to be served is dominated by the system's current configuration regardless how it arrived to that state. At the moment of modelling the problem into a Markov chain we faced the difficulty of finding a rigorous analytical expression for the distance between demands because of the dependencies among the travel distances. Bertsimas et al expected distance lower bound for the DTRP with NN policy was calculated assuming that the expected distance is bounded by inequality 3.1.7.

In order to have some insight how the DTSP behaves with different policies, we used Monte Carlo simulations on either deterministic or randomized algorithms. Simulations showed that regardless of the Poisson rate, the DTSPNN eventually arrives to a situation where the server sweeps all the demands in the region though the speed of the server remains constant. The larger the rate, the larger the expected time this situation will occur as the expected number of unattended demands present in the system increases with the rate λ. It should not be surprising that the process

terminates eventually -if the server visits demands ad infinitum, it will eventually find a condition where it can sweeps all the demands before new ones are generated.

We have also seen that the mean number of unattended demands stabilizes after a relative short period of time and that iterations start vanishing according to a geometric distribution. Thus, if we know the mean time iterations stabilize and, on any later interval to that time, we calculate the proportion of iterations then we know the parameter of the geometric distribution. This distribution can be used to predict the proportion of iterations that will (not) vanish in the future. We have considered several methods to estimate the expected time iterations stabilize. Simulated annealing, first local maximum, and ergodic estimations were proposed. The ergodic approach was the most accurate of the three as it has the smallest absolute distance between estimated mean number nodes the system stabilizes and the values observed on Table 4.1; consequently, the estimated mean time the number of unattended demands stabilize is also preferable from the ergodic estimation. If we averaged the mean number of unattended nodes of Table 4.1, we would be producing a rough ergodic estimator; hence, the closeness of the results with Table 4.1. The ergodic approach is the most computationally expensive estimator of the three methods.

We introduced the NN with random start policy that consists in instantly [1] allocate the server in a random position after a demand has been visited. The efficiency of this method is equal to the NN policy since in both cases the server starts from a random location. In the NN policy, the random location is the last demand's location, whereas in the NN with random start the starting point is explicitly declared. If we considered that in the trajectory the server travels from the last served demand to the random location new demands are generated (DTSPDR), the problem can be treated as the DTRP with NN policy, so iterations might not vanish and become unstable if $\lambda \to \frac{1}{0.52}$.

[1] No new demands are created in the server's trajectory from the position of the last served demand to its new random location.

Finally, we have proposed the partition policy that forces the server to go to those partitions with high density of demands; density of each partition is calculated using the NN policy, so this policy does not always choose the closest demand at that time. In terms of the mean time iterations vanish, the partition policy performs worse than the NN policy when the number of partitions is small, but as the number of partitions increases, this policy produces better results than the NN policy. However, with large number of partitions this policy would have similar performance to the NN policy since with high probability each partition will contain at most one demand, so the server will go to the partition where the closest demand is. Even though the partition policy performance differs from the NN, both algorithms visits demands with the same rate $\frac{1}{\lambda}$; this agrees with the fact that otherwise, the expected number of unattended demands would monotonically increase or decrease with the partition policy.

Bibliography

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *The Traveling Salesman Problem. A Computational Study.* Princeton University Press, 2006.

[2] J. Beardwood, J. H. Halton, and J. Hammersley. The shortest path through many points. *Proceeding of the Cambridge Philosophical Society*, 55:299–327, 1959.

[3] D. Bertsimas and G. J. van Ryzin. A stochastic and dynamic vehicle routing problem in the Euclidean plane. *Operations Research Society of America*, 39(4), August 1991.

[4] M. Chen. On three classical problems for Markov chains with continous time parameters. *Journal of Applied Probability*, 28(2):305–320, 1991.

[5] R. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling.* Addison-Wesley, Reading, Mass., 1967.

[6] M. Davis. *Markov Models and Optimization.* Chapman Hall, London, 1993.

[7] F. G. Foster. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, 24(3):355–360, 1953.

[8] M. Garey and D. Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[9] M. N. Ghosh. Expected travel among random points in a region. *Calculta Statistical Association Bulletin*, 2:83–87, 1949.

[10] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford, third edition, 2005.

[11] R. Hogg, J. McKean, and A. T. Craig. *Introduction to Mathematical Statistics*. Pearson Prentice Hall, sixth edition, 2008.

[12] D. Johnson. Tokio, 1988. Presented at the Mathematical Programming Symposium.

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

[14] R. Karp. Reducibility among combiatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press. New York, USA, 1972.

[15] J. Kingman. Some inequalities for the queue GI/G/1. *Biometrika*, 49(3/4):315–324, 1982.

[16] L. Kleinrock. *Queueing Systems. Volume 1: Theory*. John Wiley, New York, 1976.

[17] L. Kleinrock. *Queueing Systems. Volume 2: Computer Applications*. John Wiley, New York, 1976.

[18] D. E. Knuth. *The Art of Computer Programming*, volume 3. Addison-Wesley, second edition, 1998.

[19] R. Larson and A. Odoni. *Urbans Operations Research.* Prentice Hall, Englewood Cliffs, N.J., 1981.

[20] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons Ltd., 1983.

[21] G. F. Lawler. *Introduction to Stochastic Processes.* Chapman and Hall/CRC, 1995.

[22] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Society, 2008.

[23] O. Madsen, A. Larsen, and M. M. Solomon. *Dynamic Vehicle Routing Systems Survey and Classification.*, volume 38, pages 19–40. Springer US, 2007.

[24] E. Marks. A lower bound for the expected travel among m random points. *Annals of Mathematical Statistics*, 19:419–422, 1948.

[25] S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability.* Springer-Verlag, 1993.

[26] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[27] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[28] C. Nilsson. Heuristics for the traveling salesman problem. Technical report, 2003. Tech. Report, Linköping University, Sweden.

[29] J. Norris. *Markov Chains.* Cambridge Series in Statistical and Probabilistic Mathematics, 1997.

[30] L. K. Platzman and J. Bartholdi. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36:719–737, October 1989.

[31] P. Pollett. Analytical and computational methods for modelling the long-term behaviour of of evanescent random processes. In *Proceedings of the 12th National Conference of the Australian Society for Operations Research, Australian Society for Operations Research, Adelaide*, pages 714–535, 1993.

[32] N. Psaraftis. Dynamic vehicle routing problems. *Vehicle Routing: Methods and Studies*, 16:223–248, 1988.

[33] A. Regan, J. Herrmann, and X. Lu. The relative performance of heuristics for dynamic traveling salesman problem. In *proceedings of the 81st meeting of the Transportation Research Board*, February 2002.

[34] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the Association for Computing Machinery*, 23:555–565, 1976.

[35] C. L. Valenzuela and A. J. Jones. Estimating the Held-Karp lower bound for the geometric TSP. *European Journal of Operational Research*, 102(1):157–175, 1997.

[36] E. A. van Doorn. Quasi-stationary distributions and convergence to quasi-stationarity of birth-death processes. *Advances in Applied Probability*, 23(4):683–700, 1991.

[37] G. G. Yin and Q. Zhang. *Continuous-Time Markov Chains and Applications: A Singular Perturbation Approach (Stochastic Modelling and Applied Probability).* Springer, 1998.

[38] H. Zhang, F. Dufour, Y. Dutuit, and K. Gonzalez. Piecewise deterministic markov processes and dynamic reliability. *Journal of Risk and Reliability*, 222(4):222–545, 2008.