Road Distance Estimation From Co-ordinates Using Adaptive Tertiary Search Schemes

By

Jessica Havelock

A thesis proposal submitted to

the Faculty of Graduate Studies and Postdoctoral Affairs

in partial fulfilment of

the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science

School of Computer Science

Carleton University

Ottawa, Ontario

September 2011

Your file Votre référence
ISBN: 978-0-494-83182-3
Our file Notre référence
ISBN: 978-0-494-83182-3

Abstract

In this thesis, we consider the problem of Distance Estimation (DE) to obtain accurate real (road) distances using the x and y coordinates as the inputs. For this purpose, typically, one uses *parametric* Distance Estimation Functions (DEFs), where the parameters are learned by training with the data involving a *subset* of the true road distances. We propose to use Learning Automata (LA)-based strategies. In particular, we consider using the Adaptive Tertiary Search (ATS) strategy proposed by Oommen *et al.* to calculate the parameters of the DEF. Rather than minimizing a "goodness-of-fit" function, the ATS uses current distance estimates, feedback from the environment, and the known distances, to determine these unknown parameters. Goodness-of-fit functions, on the other hand, have been used to show that the results are competitive, but in and of themselves, they are not necessary to compute the parameters. By rigorous experimentation, we have shown that the ATS applied to 3 Dimensional DE (3DDE) can provide far superior results to those obtained using 2-dimensional DE (2DDE).

Acknowledgments

I would like to thank Prof. John Oommen for his unlimited time and patience. Throughout this process, the assistance he has given has been invaluable in helping me to develop this piece of work. Without his guidance, this would not have been possible.

I would also like to extend my gratitude to my friends and family, especially my Mother, Father and Aunt Mary Ann for their love and assistance throughout the past two years. Without their support, these years would have been less enjoyable and much more challenging. I am also grateful to Tim Elgin for his help and support during these stressful and equally rewarding times.

Finally, a most appreciative thanks to Carleton's School of Computer Science for allowing me to study a new field and to gain valuable learning experience which has helped me for my future endeavours.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Distance Estimation (DE) methods are currently used to approximate distances in a network. Whenever there is a set of points that have true, but potentially unknown corresponding inter-point distances, DE can be applied to approximate the inter-point distances. Consider a table of distances in which the points are listed in the column and row headings. The table is, typically, populated with the corresponding inter-point distances. However, consider the scenario, which more truly resembles the real world, where only a subset of these distances is known. The goal of DE is to approximate the missing distances. DE is especially useful when the distances are hard to measure, because the problem space has limited physical accesses to computing the exact true distances, or when the measurements are difficult to obtain. The ability of DE to produce useful estimates in these difficult situations makes it beneficial for both research and practical purposes.

Although DE is typically used for approximating distances in road networks, this

does not mean that it is limited in its applications. The most obvious application is in modelling travel times. For example, DE has been used by companies to schedule deliveries [6]. A more "academic" problem for the use of DE in modelling travel times would be to apply location and transportation problems (like the travelling salesman problem) in areas where the distances may not be entirely known. DE is also useful in resolving location analysis and optimization problems. It has also been used for urban planning. For example, in Turkey, DE was used to determine the optimal location for fire stations [7]. Indeed, whenever a problem requires the knowledge of all the distances between all possible points in the domain, DE can be highly advantageous.

Any system that consists of inter-connected points, like a road network, can utilize DE to model and estimate the inter-point distances. To achieve this, one typically resorts to Distance Estimating Functions (DEFs). These functions can take on any form, but the ideal functions are good estimators that also have low computational requirements. Love and Morris first introduced the concept of using simple parametric functions that employ the x and y co-ordinates for approximating distances in 1972 [3]. The first DEFs were based on common norms, most of which are still used. All these DEFs involved parameters whose values are obtained by a "training" phase in order for them to best fit the data of the system under study. Consequently. some distances in the system must be known *a priori*, and they must be used to "learn" these parameters. The accuracy of the estimations thus depends on the DEF, the system and the available data.

Since 1972, DEFs have been applied in DE and the methods utilized have been extensively tested and modified [3. 4, 8, 9, 10, 11, 12]. Since then, new DEFs have been proposed, based on other distance measures [13, 14]. These new DEFs have been derived as a consequence of applying more complex principles. For example, some

researchers have proposed using a weighted linear combination of two more primitive DEFs [13, 14]. Another novel idea for a DEF is to use a *nonparametric* method [15].

A new strategy by which DE has been improved, is by modifying and adapting the basic method involved in the DE itself. Brimberg *et al.* proposed a way of improving DE by rotating the co-ordinate axes [6]. This is done to minimize the rotational bias of a co-ordinate system and to thus improve the accuracy of the estimate. Others have tried a multi-regional approach to DE. These approaches divide the original region into smaller sub-regions, and each sub-region has its own trained parameters. One example of a multi-regional approach to DE was presented by Fildes and Westwood [5]. The scheme estimates distances based on the sub-regions that lie between the two points. The sub-regions themselves determine the weighting of the parameters used in the DEF, which, in turn, is based on the proportion of the Euclidian distance that lies in each sub-region. Another multi-regional approach uses Vector Quantization (VQ) [1, 16]. This strategy utilizes the points learned by the VQ to represent closed groups of points. The parameters that are used for inter-regional and intra-regional distance estimation themselves are obtained by training using the VQ points. Besides invoking methods that have involved DEFs, Alpaydin *et al.* suggested applying neural networks and a non-parametric method to tackle DE [15]. They compared these two methods with voting and stacking of typical DEF-based DE methods. The overall leader was the one which used stacking of simple DEFs.

In this thesis, we will attempt to contribute to the field of DE by applying a new method of determining the DEF. This method is called the Adaptive Tertiary Search (ATS) which was derived by Oommen and Raghunath [17]. To date, it has been applied to two problems, namely, the continuous stochastic point location problem [17], and the problem of parameter learning from a stochastic teacher or a stochastic

compulsive liar [18]. Both of these problems work within a stochastic domain analogous to that of DE. The ability of the ATS to perform in these stochastic domains renders it an ideal search strategy which can be used in DE.

The ATS is a search method that uses Learning Automata (LA) to perform a stochastic search "on a line" to find the parameters sought for. It searches a bounded interval by comparing the current estimate to the feedback from the environment presented in terms of the known distances and point pairs. After comparing the current estimate with the latter, a new smaller search interval is chosen. The search interval is continually decreased until it reaches a small-enough size. The aim of using the ATS is to improve the accuracy of the DEFs by removing the dependencies on the goodness-of-fit functions.

1.1.1 Goals

The purpose of this thesis is to apply ATS to DE. More precisely, the intention is to demonstrate that the ATS produces competitive results in the domain of DE. By showing that the ATS is competitive with the current state-of-the-art DE methods, it can then be extended to applications with similar domains. This is thus an important step in determining the ensemble of applications for this novel search method.

Another goal of our research is to extend the application of the ATS to the domain of 3D DE. The hope is that by doing this, we can augment the state-of-the-art DE to problems involving three dimensional DEFs. As far as we know, no research has been reported for DE in three dimensions. Thus, we hope that we can encourage research in this direction because we believe that it can improve the accuracy for "hilly" terrains.

1.1.2 Problem Statement

In this thesis we will address two problems. The first involves applying the ATS search in the DE domain. This will require some adaptations to the search strategy as the DE problem often requires multiple parameters to be determined *simultaneously.* Besides, we will also define the new environment that the ATS interacts with.

The second problem we address is the applicability of the ATS to the 3DDE problem domain. We examine the question of whether there would be any benefit in considering a third dimension in the field of DE, seeing that this field had predominantly been studied in two dimensions. Indeed, as we live in a 3D world, this limitation is not always beneficial.

1.1.3 Contributions

This thesis has three primary contributions. The first contribution is the extension of the ATS so as to involve determining multiple parameters simultaneously, and utilizing it for its application of the ATS in DE. The second involves the improvements to the field of DE itself by virtue to the use of the ATS. We argue that the application of the ATS eliminates the dependence on Goodness-Of-Fit functions, and show that it improves the overall performance by reducing errors caused by overtraining. The last contribution is again to the field of DE, where we have produced preliminary results in the field of 3DDE. These results are encouraging and show that the 3DDE problem for estimating distances over certain types of "hilly" terrains is solvable!

1.1.4 Scope

This thesis involves two areas of study, LA and DE. To focus our research we will limit the study of LA to the Adaptive Tertiary Search (ATS) as it was presented by Oommen and Raghunath in [17]. To set the ATS in the right perspective we also present other background information on the field of LA because the method that we have implemented to determine the parameters will be restricted to the ATS that uses the Linear Reward-Inaction (L_{RI}) LA.

For the field of DE, we focused on estimating distance in "road" networks. This has been done using many different methods. The method that we will implement was first presented by the authors of [3] which uses parametric DEFs to predict the distances. The parameters are first obtained by training using a known set of distances in the region in which the ATS will be used. In this setting, the family of DEFs that will be considered involves the DEFs based of the L^p norm. Although these norms may not be the optimal DEFs for all regions, they are commonly seen in the literature, and are useful for different types of regions, rural and urban.

1.1.5 Outline

The thesis starts, in Chapter 2, with a overview of topics related to the field of LA and DE. The objective of this chapter is to present the reader with relevant background information and the collection of research, in a well-organized fashion. Chapter 3 deals with the adaptations required in order for the ATS to function in a DE domain. The adapted ATS and the environments that are defined in this chapter are then tested in the subsequent chapter.

Chapter 4 contains the two dimensional analysis of the ATS. This includes the testing procedure, the test results and a discussion for the two dimensional ATS. In

two dimensions, the intention is to demonstrate that the ATS operates expediently in the domain of DE and that it produces competitive results.

To perform the tests, we shall utilize three types of data sets, namely, noiseless, noisy and those from the real world. The first set demonstrates that the behavior of the ATS in the *ideal* DE environment is flawless. The second is used to show that the ATS can function in a noisy DE environment. The experiments for the real world data measure how the ATS can function in a practical application.

In order to show that the ATS produces competitive parameters, we have invoked a hill-climbing search, for purpose of comparison. This is quite realistic, because in the literature, the parameters were either found by a linear regression or a hill-climbing search [4, 6, 13]. As a result, the parameters found by means of such a hill-climbing search proved that it is a natural method of showing that the ATS's parameters are competitive.

The 3D aspect of this thesis is presented in Chapter 5. It starts with a brief introduction to motivate the research of DE in three dimensions. We then present the three dimensional weighted L_p DEF that is tested in this chapter, as well as the method for *generating* the testing data. The testing will be done on two types of data, noiseless 3D data and noisy 3D data. Both these tests are used to show the superiority of the three dimensional weighted L_p DEF over the two dimensional weighted L_p DEF on some basic terrains. We advocate that our results should further stimulate research for DE in three dimensions.

The concluding chapter discusses important observations made throughout the thesis and presents our final conclusions and contributions. In addition, we propose directions for future work, i.e., for both the applications of the ATS and the study of DE.

Chapter 2

Literature Review

2.1 Distance Estimation

There are many well-studied problems whose solutions depend upon the distances between points in the Cartesian plain or in a geographic region. The travelling salesman problem, and vehicle scheduling problems are common examples of real-life scenarios that rely on distance information. Two common vehicle scheduling software packages, ROADNET and TRUCKSTOPS 2, use Distance Estimation (DE) methods when determining the distances between the suppliers and the customers [6]. The input to these problems are, typically, the start and end locations in the form of x and y co-ordinates of the locations in the Cartesian plain, or the latitude and longitude in the geographic region. To determine the direct distance (i.e. as the bird flies), that must be travelled between a pair of known locations, is a simple problem. However, determining the actual "road distances" (the physical distance to be travelled on the "roads" built in the community) for an area is more challenging. These road distances (also synonymously known as travelling distances, or "true" distances), can depend

on the network, the terrain, the geographical impediments like rivers or canyons, and of course, the direct distance between the respective points which serves as a lower bound for the "true" distances. The problem of distance estimation involves finding the best estimator for the true distances. This problem has been studied for over three decades, and its solutions have been put to use in many practical applications, such as in developing vehicle scheduling software, vehicle routing, and in the partitioning of districts for firefighters [6, 7, 16].

In 1972, Love and Morris proposed a method for distance estimation that uses explicit functions to estimate the true distances for a region or a large set of points [3]. These Distance Estimation Functions (DEFs) estimate the distance between a pair of points by using the co-ordinates (or latitudes and longitudes), and a set of learnt parameters. The parameters are typically learnt by using an optimization algorithm on a goodness-of-fit function, which are, in turn, provided with a set of training points and distances.

Ever since the original proposal and use of DEFs to solve this problem, new ideas have emerged to improve the accuracy of distance estimation methods. These range from formulating novel DEFs to utilizing *nonparametric* distance estimation methods. DEFs are typically founded on the principles of *known* norms; however, they have also been constructed so as to be more adaptive to the region in question. One way of achieving this is to formulate the DEF as a weighted sum of other more elementary or primitive DEFs [13, 14].

Most of the reported schemes for improving distance estimation still use DEFs in one way or another. In [6], Brimberg *et al.* proposed the use of axes rotation to eliminate the rotational bias of a co-ordinate system and to improve the accuracy of the estimated distances within a given area. Other methods that also use

a multi-regional approach have been reported. Generally speaking, they operate by considering a region and dividing it into sub-regions, or by using a Vector Quantization (VQ) approximation [1, 5, 16]. The latter uses VQ to quickly locate a set of points in the space (those which can most–accurately approximate the entire set) to approximate the distances between the co-ordinate pairs. In this case, each set of co-ordinates are represented by its closest quantized point. The idea is to have a relatively small number of quantization points compared to the number of co-ordinate pairs in order to find a better estimator. These methods still use DEFs but augment it by using additional information about the region in order to produce a better distance estimator. Alpaydn *et al.* proposed using a neural network instead of a *parametric* DEF [15]. They compared their schemes involving the prior art that utilized a parametric DEF against two nonparametric neural methods.

In this chapter, we will present a brief overview of the problem of DE and some of the methods that have been proposed. We start by formally defining the DE problem. We then present the reported goodness measures, referred to as "Goodness-Of-Fit" functions, that have been used to compare the different methods. Since most of the methods for DE involve using norms in their approaches, we include some common DEFs along with their corresponding published results. The last part of this chapter describes the distance estimation methods that will be used as a benchmark basis for comparison later in this thesis.

2.1.1 Formally Defining Distance Estimation

The distance between two points or objects is a measure of how much they differ. This difference may be with respect to their geographical locations or in their physical structures. The distance is a quantified numerical difference between the two objects

(please see Figure 2.1). For this study, we will consider the "direct" distance between the two objects to be the shortest path between their point representations. The length of the shortest path depends not only on the location of the points but also on the space (or area) in which they lie. The shortest path must be passable, implying that one should be able to reach all the intermediate points when moving from one extreme point to the other.

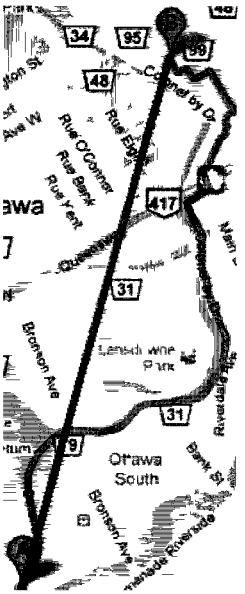

Figure 2.1. Blue path represents the true path (road path) and the red line is the direct path (as the bird flies) between Carleton University and Ottawa University

As opposed to this, there are many ways of determining the true distance between points. The simplest strategy is to maintain a table or database of the points and their known distances. Such databases often contain all the distances between large intersections for a single city, or between many (or all) the cities in a large region. These databases are difficult to use if one is to employ multiple databases so as to

determine the overall true distance. Combining these databases can result in errors due to the changes caused by different coordinates systems or by the points from which the distances are measured [4]. For example, in 1982, the Province of Ontario published all the distances between large intersections throughout the province [4], which was complied in a document referred to as the "Distance Table". While the collection of this data is impressive, it is still difficult to calculate the distance between two locations whose pertinent data is not included in it. Rather, to obtain this, one must find all the relevant distances between the intermediate intersections and thereafter, compute the overall true distances. This, in itself is not a problem when finding a *single* true distance, but it becomes challenging when one needs to quickly determine a set of distances in the region.

For mathematical rigor, distance functions are traditionally used to determine the real distances between points. These distance functions are specific to the terrain or space under consideration. For example, the distance between points on an axis is the absolute value of the difference between their coordinates. The distance between points on a "smooth" Cartesian plain is the Euclidian distance. Both of these distance functions report the true inter-point distance in their respective spaces. However, determining the true distances in a space that is non-uniform and unknown is much more difficult. To determine the true distance between points in a network or on a non-uniform or hilly terrain, one requires additional information so as to obtain the "correct" distance function. In fact, the problem of finding the distance function that yields the true distance for *all* points in these more complex spaces may actually be both infeasible and unreasonable. Rather, one may have to resort to approximate prediction of the true distance due to the complex computations involved in *exactly* determining them, or due to the fact that the true distance cannot be computed with

just the information that is provided.

The prediction or Distance Estimation (DE), is typically done by finding an appropriate DEF A DEF is a mapping from $R^n X R^n$ to R, and returns the estimate of the true distance. The inputs to the DEF are the locations of the two points, and it produces an estimate of the distance between them by incorporating the set of parameters into the DEF. One observes that the set of parameters alluded to must be *learnt* in order for the DEF to best represent the space.

Definition 1. *A **Distance Estimation Function (DEF)** is defined as a function* $\pi(P_1, P_2|\Theta) \cdot R^n X R^n \longrightarrow R$, *where* $P_1 = (x_1, x_2, ..., x_n)$ *and* $P_2 = (y_1, y_2, ..., y_n)$ *are points in* R^n, *and* Θ *is a set of parameters whose values characterize* π, *and which must be learnt using a set of training points with known true inter-point distances.*

The set of parameters, Θ, is typically learnt by minimizing a goodness-of-fit function. These goodness-of-fit functions are used to measure how well a network or region is represented by the DEF.

2.1.2 Goodness-of-Fit

Goodness-of-Fit (GoF) functions are measures of how good a DEF estimates the true (but unknown) distances. Several GoF functions have been consistently utilized in the literature pertaining to the field of DE. The simplest and most commonly-used one is the Absolute-value Difference (AD) given be Equation (2.1) [1, 3, 5. 6, 8, 9, 10, 11, 14, 15, 16], which was originally proposed by Love and Morris. It is given by the sum of the absolute values of differences between the estimated distances and the actual distances. Observe that this GoF measure leads to a natural way of minimizing the overall error of the DEF, although larger distances are given more weight than smaller ones [1, 3, 6, 16].

$$AD = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |A(P_i, P_j) - \pi(P_i, P_j|\Theta)|. \tag{2.1}$$

Although the AD is both a common and conveniently simple GoF function, it is not useful when comparing regions with varying topographies (or geographies). This is due to the fact that it favors larger distances, and that it does not take into account the *relative* values of these distances. To compare different regions and to also incorporate such relative quantities, the Relative Absolute-value Difference (RAD) function given in Equation (2.2) is often recommended [9]:

$$RAD = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |A(P_i, P_j) - \pi(P_i, P_j|\Theta)|}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} A(P_i, P_j)}. \tag{2.2}$$

The next and third most-common GoF function is the Normalized Absolute-value Difference (NAD), given by Equation (2.3) [6, 8, 19]. Like the AD, the NAD uses the sum of the absolute values of the differences between the distances; however, the NAD normalizes these differences with respect to their respective actual quantities. This allows the errors for both large and small distances to be weighted equally [6, 19]. While the AD measures the overall error, the NAD quantifies the weighted percentage error for each distance as follows:

$$NAD = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{|A(P_i, P_j) - \pi(P_i, P_j|\Theta)|}{A(P_i, P_j)}. \tag{2.3}$$

In spite of the advantage of the above indices, the most commonly-used GoF function is the sum of Square Deviation (SD) given by Equation (2.4) [1, 3, 5, 6, 8, 9, 10, 11, 15, 16]. This index was also proposed by Love and Morris, and has properties which are useful, such as its statistically significance and continuity. The SD is both continuous and differentiable with respect to the parameter Θ. This allows the user

to employ a gradient decent search scheme to determine the "best" parameter Θ [1, 8, 15, 16]. Although the SD favors larger distances, it does not possess as large a bias as the AD does [3, 6, 15].

$$SD = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{A(P_i, P_j) - \pi(P_i, P_j|\Theta)}{\sqrt{A(P_i, P_j))}} \right)^2 . \tag{2.4}$$

2.1.3 Distance Estimation Functions (DEFs)

In this section we shall present a brief overview of some of the common DEFs and their properties, and proceed to compare their relative performances.

Metrics, Norms and DEFs

The DE problem can be applied in very complex spaces. Indeed, the question of whether the space is smooth or even continuous is not a functional requirement. Despite the difficulty of the problem, the most useful DEFs are modelled in terms of simple mathematical distance measures. The corresponding metrics and norms are the two features of such measures.

Definition 2. *A **metric** is defined as a function $d(X,Y) : R^n X R^n \longrightarrow R$, where $X, Y \in R^n$ and $d(X,Y)$ satisfies the following properties:*

1. $d(X,Y) > 0$ *if* $X \neq Y$ *(Non-negativity)*
2. $d(X,Y) = 0$ *if and only if* $X = Y$ *(Identity)*
3. $d(X,Y) = d(Y,X)$ *(Symmetry)*
4. $d(X,Z) \leq d(X,Y) + d(Y,Z)$ *(Triangular inequality).*

The properties associated with a metric are also pertinent to true distances, and are thus very useful when estimating them. Further, scalability and the triangular inequality are crucial in the study of DE because many applications require unit conversions or vector operations (i.e., the operations of adding two distances or vectors), and must be valid independent of the scale to which the "map" is drawn. Positive-definiteness is another important property due to the fact that it is meaningless to report negative distances in the real world. Observe that if a DEF is not symmetric, the distance between two points would be direction-dependent. Although this is sometimes the case for geographical reasons (for example, involving cities with one-way roads, or for uphill-downhill traffic) generally speaking, the distance between two points is the same in both directions. In order to represent distances for non-symmetric domains, the concept of a weak metric is often invoked.

Definition 3. *A **weak metric** is defined as a function* $d(X,Y) : R^n X R^n \longrightarrow R$, *where* $X, Y \in R^n$ *and* $d(X,Y)$ *satisfy the following properties:*

1. $d(X,Y) \geq 0$ *(Non-negitivity)*
2. $d(X,Y) = 0$ *if and only if* $X = Y$ *(Identity)*
3. $d(x,z) \leq d(x,y) + d(y,z)$ *(Triangular inequality).*

By modifying the property of non-negativity and removing the symmetric requirement, the above metric becomes a much weaker condition, which is more easily satisfied. It also permits the user to model more diverse networks, such as those involving one-way streets.

The last family of measures involved are called norms which are defined below.

Definition 4. *A **norm** is defined as a function* $d(X) : R^n \longrightarrow R$, *where* $X \in R^n$ *and* $d(X)$ *satisfies the following properties.*

1. $d(aX) = |a|d(X)$ *where* $a \in R$ *(Positive scalability)*

2. $d(X) = 0$ *if and only if* $X = 0$ *(Separates points)*

3. $d(X + Y) \leq d(X) + d(Y)$ *(Triangular inequality)*

Norms possess stricter constraints than metrics because they are characterized by the same properties of the latter, except symmetry Although norms are defined from $R^n \longrightarrow R$, a norm can, just as well, be defined over the domain $R^n X R^n$ and be required to be symmetric. Consequently, a well-defined norm can also be used to represent a metric. A norm and a metric differ, however due to the property of scalability.

L^p-based DEFs

The properties of norms and matrices are ideal for a DEF because they are capable of yielding measurable and "usable" distances. Consequently, the simplest way of creating a DEF, that also possesses the properties of norms or metrics, is to use a well-known norm/metric.

The most common types of DEFs are those based on the family of L^p norms, traditionally use for computing distances. The set of L^p norms for various values of p are given below:

$$L_1(X) = \sum_{i=1}^{n} |x_i| \tag{2.5}$$

$$L_2(X) = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \tag{2.6}$$

$$L_p(X) = \left(\sum_{i=1}^{n} (|x_i|^p \right)^{1/p} \tag{2.7}$$

$$L_\infty(X) = MAX(|x_1|, |x_2|, ...|x_n|), i = 1, 2...n \tag{2.8}$$

Although these measures are norms as defined above, their specific characteristics depend on the value of p. For example, if $p = 1$, one obtains the so-called Taxicab norm, which is the L^1 norm in Equation (2.5). The quantity is merely the sum of the absolute values of the co-ordinates, and derives its name from the "distance" a taxi would have to drive from one point to another on a rectangular grid. When $p = 2$, the L^2 norm of Equation (2.6) becomes the Euclidean norm, and is typically used to measure distances on a plane. Although the L^1 and L^2 norms are the ones most frequently used, L^p norms for other values of p ($p \in R$) also have significance. When $p = \infty$, Equation (2.8) returns the largest absolute value in the vector.

The various L^p norm have been used as stepping stones to design DEFs. Indeed, some of the most common DEFs have, in turn, been derived from the L^p norms as shown in Table 2.1 (Equations (2.9) to (2.14)), also initially suggested by Love and Morris [3, 10]. The input to these functions are the co-ordinates of the input vectors, X_1 and X_2.

In practice, these DEFs are first trained on a set of co-ordinates and their known inter-point distances for a specific region. This training is done so as to obtain the

Norms Derived from L^p norms				
DEF	Expression	Name of DEF	Parameters	Equation
F_1	$(\sum_{i=1}^{n} \lvert x_{1i} - x_{2i} \rvert^p)^{1/p}$	L^p	p	(2.9)
F_2	$k\left(\sum_{i=1}^{n} \lvert x_{1i} - x_{2i} \rvert^2\right)^{1/2}$	*Weighted Euclidean*	k	(2.10)
F_3	$k\left(\sum_{i=1}^{n} \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/p}$	Weighted L^p	k and p	(2.11)
F_4	$k\left(\sum_{i=1}^{n} \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/s}$	kL_{ps}	k, p, and s	(2.12)
F_5	$(\sum_{i=1}^{n} (k_i * \lvert x_{1i} - x_{2i} \rvert^p)^{1/p}$	*Generalized WeightedL$_p$*	$\{k_i\}$	(2.13)
F_6	$[(X_1 - X_2)M(X_1 - X_2)]^{1/2}$	Elliptical DEF	$M = \begin{bmatrix} k & p/2 \\ p/2 & s \end{bmatrix}$	(2.14)

Table 2.1: List of DEFs related to various L^p norms and their corresponding parameters

"best" parameters for the DEF given the training data. Once the parameters have been determined, the DEF can then be used for estimating distances in the same region.

Each of these DEFs has been extensively tested on different data sets [3, 6, 8. 9, 10, 11, 15, 16, 19, 20, 21]. Equation (2.9) performed well over a small area, usually described by the boundaries of a given city, but did not perform well over larger areas [3]. Equation (2.10) produced usable results for both a network of roads from within a city and which extended to other cities [3]. The weighted Euclidean DEF in Equation (2.10) out preformed the L^p DEF in Equation (2.9); however, both of these were out-performed by the weighted L^p DEF given by Equation (2.11) [3].

Equation (2.11) has an advantage[1] over the previous *two* DEFs because it contains two learnt parameters. As a result, this DEF is more flexible and better able to adjust to the environment. The percentage improvement for Equation (2.11) over Equation (2.10) has a large variation depending on the geographical area being studied. In a study by Berens, the percent improvement ranged from 0% to 11.27% [12, 15].

[1]In preforming comparisons, there are essentially two main GoF criteria: the AD in Equation (2.1), and the SD in Equation (2.4).

This improvement was based on the GoF criteria discussed in Section 2.1.2. Berens and Korling felt that the additional calculation of multiple parameters could not be justified. The difference in improvement must be viewed against the backdrop of the considerations due to the environment. Berens and Korling reported a AD improvement of 0.12% and a SD improvement of 0.03% in Germany with 117 cities and 6786 distances [11]. Love and Morris reported AD and SD improvements of 2.4% and 8.3% respectively in Germany with 15 cities and 100 distances [11]. In the United States, Love and Morris reported AD and SD improvements of 10.55% and 34.63% respectively when using Equation (2.11) over Equation (2.10). Some of the reasons that have attributed to the large difference in improvement are the size of the network tested, the road density, and the difference in the road structures. Berens and Korling state that "The Federal Republic of Germany is a comparatively small country, with a well developed road network. Thus there is hardly any room for improvement in the accuracy by abandoning d1" where d1 is Equation (2.11). While improvements are small for a well-developed area such as the Federal Republic of Germany, the weighted L^p DEF shows an improvement over the weighted Euclidean DEF, especially in more rural or less developed road networks.

The weighted L^p DEF shows good results but is out performed by the kL_{ps} DEF in Equation (2.12) [10]. Equation (2.12) is characterized by three parameters and is shown to be statistically stronger then the weighted L^p DEF. Of all the DEFs listed in Table 2.1, Equation (2.11) performed second-best in all types of regions. It performed well in both rural and urban networks, but was out-performed by Equation (2.12) [3]. Equation (2.12) was the best estimator in urban settings [10]; however, the best DEF for urban areas was the Elliptical DEF in Equation (2.14). The latter performed well in networks that were not highly developed but did not perform well otherwise [10].

In 2000 Uster and Love proposed using a generalized weighted L^p DEF, given by Equation (2.13). They reported that the generalized weighted L^p norm has great improvement for areas with direction "non-linearity"; however, in areas with little direction "non-linearity", (i.e., when the k's are equal), the generalized weighted L^p has no significant improvement over the weighted L^p.

Overall, considering all the DEFs shown in Table 2.1, Equation (2.12) was the best-performing DEF and is the best estimator for the generic situation [3, 10].

The general conclusion offered in the literature is that the specific DEF should be chosen based on the properties of the geographical area in question. Of course, as explained later, a multi-regional approach for areas with different geographical features, is always superior.

The Spherical Distance Function

Another group of DEFs that was proposed by Love and Morris is based on spherical distance functions [3]. The spherical distance function is used to find the shortest distance between two points that lie on the surface of a sphere. Such functions are appealing because points on the earth are essentially on a spherical surface. The spherical distance function uses the longitude and latitude of the points in question to find the corresponding shortest distance over the spherical surface as shown in Figure 2.2. This function is, in turn, shown in Equation (2.15).

$$SD(X_1 - X_2) = arccos[cos(b - a) + sin(a) * sin(b * (1 - cosC))]. \quad (2.15)$$

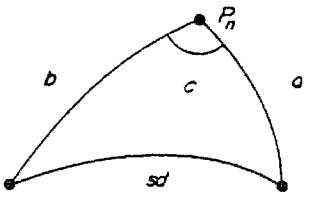

Figure 2.2: Spherical distance, where P_n is the north pole [3].

In the above equation:

$$\begin{aligned} a &= 90^o - x_{11} \quad \text{where } x_{11} \text{ is the latitude of } X_1 \\ b &= 90^o - x_{21} \quad \text{where } x_{21} \text{ is the latitude of } X_2 \\ C &= |x_{12} - x_{22}| \quad \text{where } x_{12} \text{ and } x_{22} \text{ are the} \\ & \qquad \text{longitudes of } X_1 \text{ and } X_2 \text{ respectively.} \end{aligned}$$

Spherical DEFs

From the above spherical distance function, one can derive three distinct DEFs, shown in Equations (2.16) to (2.18):

$$F_1(X_1 - X_2) = k(SD) \tag{2.16}$$

$$F_2(X_1 - X_2) = SD^p \tag{2.17}$$

$$F_3(X_1 - X_2) = k(SD)^p, \tag{2.18}$$

where k, p,and s are the learnt parameters of the respective functions.

Overall, the spherical DEFs did not perform as well as the DEFs derived from the L^p norms given by Equation (2.9) to (2.14). Of the spherical DEFs, Equation (2.17) had the worst performance, while Equation (2.18) performed the best [3]. Equation

(2.18) showed a similar performance as Equation (2.10) when used on data for larger regions; however, it performed poorly when tested on smaller regions [3].

In conclusion, the usefulness of the spherical DEFs has been shown to be inferior to the L^p DEFs.

Round Vs. Block DEFs

Block and round norms are identified by their contours. The L^p norm is considered a round norm because for $1 < p < \infty$, the unit contour has no corners. Alternatively, block norms are norms that have polygons as their unit contours in R^2.

Two of the simplest block norms are the L^1 (Equation (2.5)), and the L^∞(Equation (2.8)), where their unit contours can be seen in Figure 2.3. A more complicated example of a block norm is also shown in Figure 2.3. This block norm has eight parameters and is defined in Equation (2.19):

$$F(X_1, X_2) = \sum_{g=0}^{7} z_g |(sin(180\frac{g}{8}), cos(180\frac{g}{8}))(X_1 - X_2))|. \tag{2.19}$$

$$\text{where } X_i = (x_{i,1}, x_{i,2})^T \text{ for } i = 1, 2 \text{ and } z_g \in R \; \forall g = \{0, 2..., 7\}.$$

Ward and Wendell proposed using block norms as DEFs to improve the accuracy of the distance estimation [22]. Walker found that block norms became more efficient with more parameters but did not always out-perform round norms with less parameters [4]. Indeed, a block norm with eight parameters performed "marginally" better than using the kL_p with two parameters [4]; however, the additional computational cost was recorded to be extremely high for a "marginally" better result.

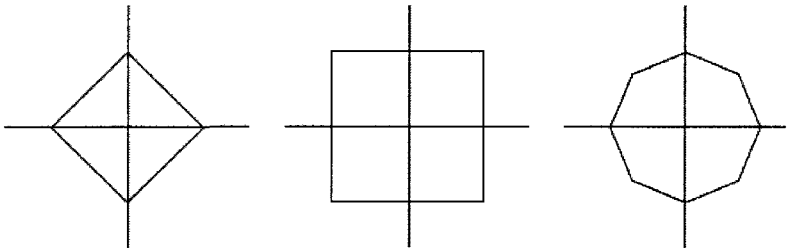

Figure 2.3: Block Norms: Left; L^1 norm, Center; L^∞ norm, Right; eight parameter block norm from Equation (2.19).

2.2 Distance Estimation Methods

In this section we briefly report various distance estimation methods that have been proposed. For each method we outline the procedure required and give a brief overview of the available results.

2.2.1 DEFs using Axis Rotation

The family of L^p norms have directional biases when $p \neq 2$. When a norm has a directional bias it produces a smaller measure for routes in one direction than in another. In other words, the coordinate system also affects the measure. For the L^p norm in which $p > 2$, the distance between two points is minimized when the direction of travel is 45 degrees from one of the axis, and maximized when the direction is along the axes. When $0 < p < 2$, the distance is minimized along the axis and maximized at 45 degrees to the axis, as shown in Figure 2.4.

In order to improve the accuracy of the family of kL_p norms, Brimberg *et al.* proposed rotating the axis to optimize the coordinate system for the chosen DEF [6],

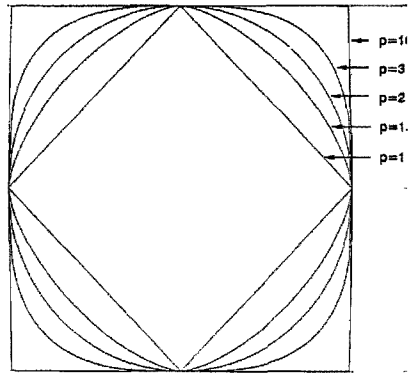

Figure 2.4: Unit contour of the L^p norms [4].

and more specifically when one is using norms with a directional bias. The directional bias of a norm $K(X)$ is shown in Equation (2.20), where $X = A - B$ and A, B are points in R^2.

$$r(\theta) = K(X)/l_2(X) \tag{2.20}$$

The variable θ is the angle of travel from the x-axis, and is given by $\theta = tan^{-1}(x_2/x_1)$. When $K(x)$ is a L^p norm, $r(\theta)$ is periodic with a period of $\frac{\pi}{2}$; therefore, the largest angle of rotation for the coordinate system, $\gamma \in [0, 90]$, is 90 degrees, where γ is used to rotate the coordinate system by means of a simple transformation given in Equation (2.21):

$$(a_{1new}, a_{2new}) = (a_1, a_2)A = \begin{bmatrix} cos\gamma & -sin\gamma \\ sin\gamma & cos\gamma \end{bmatrix}. \tag{2.21}$$

To obtain results, Brimberg *et al.* compared the kL_p norm, and the weighted Euclidian norm, kL_2, to the kL_p norm augmented with an axis rotation, termed as the

$kL_{p,\gamma}$ norm. To determine the values of k, p, and γ, the goodness-of-fit functions (2.1), (2.3), and (2.4), were minimized using an incremental process. This was performed for 18 regions, each containing 14 training points. In some regions the kL_2 norm outperformed the kL_p norm. When the $kL_{p,\gamma}$ norm was applied to these areas, the functions preformed better than both the kL_p norm, and the kL_2 norm as $p \to 1$. When the $kL_{p,\gamma}$ norm was applied to the areas where the kL_p norm outperformed the kL_2 norm, the $kL_{p,\gamma}$ norm had different values for its parameters in all but a single case in which $\gamma = 0$. The improvement in the performance for the $kL_{p,\gamma}$ norm was dependent on the the bias of the network. The regions that had a predominantly rectangular network benefited more from the $kL_{p,\gamma}$ norm. This modification has been suggested for the family of L^p norms, but it can also be applied to any multi-parameter DEF.

2.2.2 Multi-regional Distance Estimation

A multi-regional method of DE was developed by Fildes and Westwood [5]. Their model for DE in a region involved dividing that region into subregions. Each of these subregions would then have their own set of parameters, Θ_k. The idea is that the smaller the region, the more accurately the set of parameters can represent the natural obstacles and detour factors.

To support their study, these authors used the simple DEF in Equation (2.10). They choose this DEF because of the obvious manner in which it shows the detour factor, k, and to more easily display how the regions dealt with natural obstacles. To calculate the distances, one had to consider two cases, i.e., the intra-regional and the inter-regional distances. The intra-regional distances, where both points A and B are in the same subregion m, involve a simple calculation using the traditional

method that works with a DEF. However, for the case of the inter-regional distance, if points A and B are in different subregions, their method requires more intensive computations. An example of this is shown in Figure 2.5, where both the distances traveled within Region 1 and Region 2 must be considered.

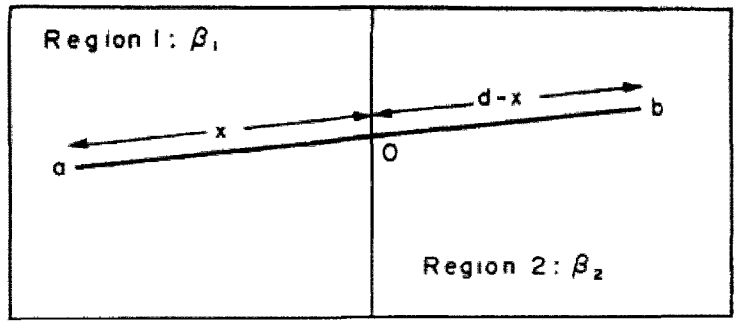

Figure 2.5: Multi regonal Distance [5].

Examples of the above distances are given below:

Intra-regional distances:

$$F(a,b) = K_m \sum_{i=1}^{n} (|a_i - b_i|^2)^{1/2}.$$

Inter-regional distances:

$$\begin{aligned} F(a,o) &= K_m \sum_{i=1}^{n} (|a_i - o_i|^2)^{1/2} \\ F(o,b) &= K_n \sum_{i=1}^{n} (|o_i - b_i|^2)^{1/2} \\ F(a,b) &= F(a,o) + F(o,b) \\ &= K_m \sum_{i=1}^{n} (|a_i - o_i|^2)^{1/2} + K_n \sum_{i=1}^{n} (|o_i - b_i|^2)^{1/2} \\ &= \sum_{j=1}^{n} (|a_j - b_j|^2)^{1/2} \cdot \\ &\quad \left(K_m \frac{\sum_{i=1}^{n} (|a_i - o_i|^2)^{1/2}}{\sum_{i=1}^{n} (|a_i - b_i|^2)^{1/2}} + K_n \frac{\sum_{i=1}^{n} (|o_i - b_i|^2)^{1/2}}{\sum_{i=1}^{n} (|a_i - b_i|^2)^{1/2}} \right) \end{aligned}$$

Observe that the inter-regional distance is based on the proportional distance spent in each region, and one can thus easily generalize this to n subregions as below:

$$\begin{aligned} F(a,b) &= \sum_{j=1}^{n} (|a_j - b_j|^2)^{1/2} \left(\sum_{m=1}^{n-1} K_m \frac{\sum_{i=1}^{n} (|a_i - o_{k,i}|^2)^{1/2}}{\sum_{i=1}^{n} (|a_i - b_i|^2)^{1/2}} + K_n \frac{\sum_{i=1}^{n} (|o_{n,i} - b_i|^2)^{1/2}}{\sum_{i=1}^{n} (|a_i - b_i|^2)^{1/2}} \right) \\ &= (totalDistance) \sum_{m=1}^{n} (K_m * (\text{proportion of distance traveled in region m})). \end{aligned}$$

To compare the multi-regional method with the traditional single regional method, the authors considered 220 points across the UK. There were originally 11 different regions; however, after determining the best detour factor for each region, eight regions had very similar detour factors. As a result, eight of these regions were combined into one leading to a map involving four different regions. The detour factor that was

used for the combined eight regions was very similar to the one used in [3] by Love and Morris on distances in Wisconsin. The regions that did not have similar detour factors also had noticeable obstacles within the regions.

The multi-regional approach demonstrated a noticeable improvement over the single region approach [5]. The mean absolute percentage error for the single region approach was 8.31%, while the multi-regional approach had a mean absolute percentage error of only 6.15%. The error for the multi-regional approach was even further improved when the regions were modified so that the natural obstacles were more closely considered. With this modification, the mean absolute percentage error became 5.09%. Overall, the multi-regional approach yielded significant improvements over the single region approach. This improvement is more noticeable when there are large obstacles that affect the detour factor in a certain region. Although there are more computations involved, the benefits of utilizing a complex multi-regional strategy may be well worth the additional computation.

2.2.3 Vector Quantization for Distance Estimation

Vector Quantization (VQ) uses codebook vectors to represent sets of points. The aim is to minimize the distortion of the data by matching the data points to the "closest" codebook vectors. This problem is complex due to the possible combination of codebook vectors and corresponding sets of data points.

The method of DE used by Oommen *et al.* combines VQ and LA in the modeling and estimation processes [1, 16]. The VQ is used to group the points into sets ($C_k = \{P_{k,i} : 1 \leq i \leq N_k\}(1 \leq k \leq R)$, where $P_{k,i}$ is i^{th} point in the k^{th} region), so that a parametric DEF can be used to estimate the true distance. The set C_k consists of the points closest to the k^{th} codebook vector ($\{Q_{k,j} : 1 \leq j \leq M\}$, where M is the

number of points used to quantize the space). These codebook vectors are determined though an algorithm consisting of two phases, i.e., an intra-regional polarization and an inter-regional polarization.

The intra-regional polarization is the phase in which the codebook vectors move with regard to the points *within* the same cluster. For this phase of the VQ, the codebook vectors are updated based only on the points that are their nearest neighbours, as shown in Equation (2.22). The reason for only using the nearest neighbours is that the order of the codebook vectors is irrelevant, thus simplifying the computational complexity.

$$Q_{k,j}(t+1) = \begin{cases} (1-\alpha)Q_{k,j}(t) + \alpha P_{k,i} & \text{If } Q_{k,j} \text{ is the closest code book vector} \\ & \text{to the data point } P_{k,i} \\ Q_{k,j}(t) & \text{Otherwise.} \end{cases} \tag{2.22}$$

To discourage premature convergence and to also allow for fine tuning for the position of the codebook vectors, the parameter α is linearly decreased from 1 to 0.9 for the first $1,000$ iterations. After these $1,000$ iterations, the fine turning was accomplished by decreasing α to 0.2 and then further decreasing it linearly to 0.1 over $2,000$ time steps. Once the codebook vectors are positioned with respect to the points within the same cluster, the distance between the other clusters is taken into account. This is done in the inter-regional polarization phase.

The inter-regional polarization phase has the goal of maximizing the separation of the corresponding codebook vectors in order to minimize the misclassification. This is done by comparing the classification of each individual point, P, in the training set to the closest two codebook vectors, say Q_a and Q_b. By using this comparison, the positions of the codebook vectors can be modified so that they migrate away from

data points that belong to other clusters.

The update process is shown below, where α is kept constant. Oommen *et al.* used the values $\alpha = 0.1$ and $\epsilon = 0.25$. The value W was set to be one-hundredth of the distance between the codebook vectors, Q_a and Q_b.

Algorithm 1 VQ update process

if $Q_a, Q_b \in C_k$ $P \in W$ **then**
  $Q_a(t+1) = (1 - \epsilon\alpha)Q_a(t) + \epsilon\alpha P$
else if $Q_a, Q_b \in C_k$ $P \in W$ **then**
  $Q_b(t+1) = (1 - \epsilon\alpha)Q_b(t) + \epsilon\alpha P$
else if $Q_a \in C_k$, $Q_b \in C_j \neq C_k$ $P \in W$ **then**
  $Q_a(t+1) = (1 - \alpha)Q_a(t) + \alpha P$
else if $Q_a \in C_k$, $Q_b \in C_j \neq C_k$ $P \in W$ **then**
  $Q_b(t+1) = (1 + \alpha)Q_b(t) - \alpha P$
else if $Q_a \in C_j \neq C_k$, $Q_b \in C_k$ $P \in W$ **then**
  $Q_a(t+1) = (1 + \alpha)Q_a(t) - \alpha P$
else if $Q_a \in C_j \neq C_k$, $Q_b \in C_k$ $P \in W$ **then**
  $Q_b(t+1) = (1 - \alpha)Q_b(t) + \alpha P$
else
  $Q_a(t+1) = Q_a(t)$; $Q_b(t+1) = Q_b(t)$
end if

Once the codebook vectors are determined, the parameters for the DEF must be found. Oommen *et al.* used a parametric DEF where the distance between points A and B, that belong to the i^{th} and j^{th} cluster respectively, is the weighted Euclidian norm:

$$F(A, B) = K_{i,j}||A - B||. \tag{2.23}$$

Since the value $K_{i,j}$ must be computed for *all* the cluster pairs, Oommen *et al.* used the following algorithm (Algorithm 2) to accomplish this.

The VQ method of DE was validated on a set of 103 cities in Turkey. Of these 103 cities, 80 were used in the training phase and 23 were used to test the accuracy of the

Algorithm 2 GetCoffByVQ

Input: The set of codebook vectors, the training set L and the distances $F(P_i, P_j)\ \forall P_i, P_j \in L$

Output: The set of parametized coefficients, $\{k_{a,b} : 1 \leq a, b \leq R \times M; a \leq b\}$

Method:

1: **for** $a = b$ to RxM **do**
2: $\lambda_{a,b} = 1$
3: $k_{a,b} = 0$
4: **end for**
5: **repeat**
6: Get any distinct pair of points P_i, $P_j \in L$
7: **if** Closest codebook vector to P_i, P_j and Q_a, Q_b respectively **then**
8: $k_{a,b} = (1 - \lambda_{a,b})k_{a,b} + \lambda_{a,b}(F(P_i, P_j)/||P_i - P_j||)$
9: Decrease $\lambda_{a,b}$
10: **end if**
11: **until** satisfied

End Algorithm

resulting distance estimation. To compare results, four common DEFs were modified to be the parametric DEFs, and these results were compared against others reported in the literature, i.e., Equations (2.9), (2.10), (2.11), and (2.12), with the parameters $K_{i,j}$, $S_{i,j}$, and $P_{i,j}$, $\forall i, j \in \{1, 2, ..., M\}$ where $i \neq j$. The training and testing errors are reported in Table 2.2.

Table 2.2: Errors obtained for DE by using VQ with the corresponding DEFs [1]

DEF	Training Error	Testing Error
L^p DEF	8.024%	12.396%
Weighted Euclidean DEF	3.65%	8.41%
Weighted L^p DEF	3.36%	8.94%
kL_{ps} DEF	3.18%	8.14%

The VQ method for DE has as also been implemented using a discretized VQ by Oommen *et al.* in [16]. This method uses the same process to determine the codebook vectors and the parameters for the DEFs. The only difference is that the

codebook vectors lie in a discretized space. This allows for less computation and faster convergence for the codebook vectors. The overall results found by Oommen *et al.* for the Discrete VQ method were almost identical to those obtained for the continuous VQ method.

2.2.4 Neural Networks for Distance Estimation

Alpaydin *et al.* introduced two new methods for DE using Neural Networks (NNs). The first was a multi-layer NN trained using back-propagation. The second method uses a regression-based NN implementing nonparametric regression using Gaussian kernels [15]. In addition to these two new methods of DE, Alpaydin *et al.* compared the traditional DE methods, with two combinations of multiple estimators, voting and stacking.

The first new method of distance estimation presented in [15] is the multi-layer Perceptron. It uses a feed forward NN, where the input is presented to the first layer and this propagates to the outer layers via the intermediate neurons. Each unit performs a weighted sum of its inputs, where the weights, $\{\theta\}$ are the learnt parameters. Since distance estimation uses a function to approximate distances, the NN only requires a single hidden layer. Without loss of generality, a NN with one hidden layer can approximate any function. The output of this multi-layer NN is:

$$F(x|\theta) = F(x|T, W) = \sum_{h=1}^{H} T_h g_h(W_h, x) + T_0, \tag{2.24}$$

where x is the point coordinates, T_hs are the weights for the hidden layer, where there are H units, T_o is the bias weight and $g_h(W_h, x)$ (given below) is the smoothed output of the hidden layer as shown in Equation (2.25), W_hs are the weights for the h^{th} unit in the hidden layer, and where there are k units in the first layer.

$$g_h(W_h, x) = \frac{1}{a + exp(-\sum_{i=1}^{k} u_i W_{ih} - W_{0h})}. \tag{2.25}$$

Back-propagation is used to train the weights, $\{\theta\} = T \cup W$. This process involves a gradient decent of the error function, e. The initial weights are arbitrarily defined and the weights are trained by repeatedly changing the values according to the difference calculated in Equation (2.26).

$$\delta\theta_p = -\eta \frac{de}{d\theta_p} \quad \text{where } \eta \text{ is the step size in the gradient descent.} \tag{2.26}$$

The nonparametric method of DE that using kernel density estimation is called a Regression NN. Overall, it determines the expected values of the distances by approximating their corresponding distribution. It does this by approximating the distribution of x ($\hat{p}(x)$) as in Equation (2.27).

$$\hat{p}(\dot{x}) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{i=1}^{n} K(\frac{x - X_i}{h}.) \tag{2.27}$$

The n dimensional training points for $p(\dot{x})$ are X_i, K is the kernel density function, and h is the window width of the kernel. Then, the estimate of the joint density function of the points and their distances, $\hat{f}(x, y)$, in Equation(2.28) can be used to determine $\hat{y}(x)$ in Equation (2.32).

$$\hat{f}(x, y) = \frac{1}{\sqrt{2\pi}\sigma} \sum_{i=1}^{n} K(\frac{x - X_i}{h}) \frac{1}{2\pi\sigma} exp(-\frac{(y - Y_i)^2}{2}\sigma^2). \tag{2.28}$$

Using $\hat{f}(x, y)$, $E[y|x]$ is found as:

$$E[y|x] = \frac{\int y\hat{f}(x, y)dy}{\int \hat{f}(x, y)dy}, \text{ and} \tag{2.29}$$

$$\hat{y}(x) = \frac{\sum_i Y_i K(\frac{x-X_i}{h})}{\sum_i K(\frac{x-X_i}{h})}. \tag{2.30}$$

Where $K(u)$ be the Gaussian Parzen Window given by:

$$K(u) = \frac{1}{\sqrt{2\pi\sigma}}^{d} exp(-\frac{||u||^2}{2}). \tag{2.31}$$

Finally the estimate $\hat{y}(x)$ is shown below:

$$\hat{y}(x) = \frac{\sum_i Y_i exp(\frac{-||x-X_i||^2}{2h^2})}{\sum_i exp(\frac{-||x-X_i||^2}{2h^2})}. \tag{2.32}$$

Using this estimation method requires little training. Besides, it requires only a single iteration. The down side of this scheme is that the training values must be stored in order to calculate the estimated distance $\hat{y}(x)$. To improve robustness, a combination of multiple estimators was considered in the form of voting and stacking. Voting is a simple act of taking the average of the estimators. It is therefore a regular practice to use an odd number, especially for binary cases. In this case, Alpaydin *et al.* used three estimators for both voting and stacking. Stacking is also a very simple concept. Two of the estimators are trained on the same training set. The last is trained on a separate training set and its input is the output of the two previous estimators.

These four methods were compared against the typical parametric model using Equations (2.9) to (2.12). The data consisted of 51 cities from Turkey in which twenty-eight were used in the training set and 23 were used for testing.

Of the four typical parametric models, the one using Equation (2.12) performed the best. It had an average error of 8.1%. The multi-layer perceptron method gave a slightly better result by about 5%. It took approximately 100 epochs for the values to converge, and as a result, it was computationally expensive. The regression NN

was much faster to train. However; it did not perform as well as either the multi-layer perceptron method or the typical parametric model. It had its lowest average error of approximately 8.50% when $h = 55$. The two best distance estimators involved were Voting and Stacking. They had an average error of 7.69% and 7.41% respectively. Both of these combinations of estimators improved on all of the single estimators; however, they both required much more training and computation than any one of the single estimators.

2.2.5 Confidence Intervals

All of the methods that have been proposed for DE have tried to minimize the error of the estimator function. None of these schemes have led a method of estimating the *error* for the distances in question. Love *et al.* used intervals to estimate the distances, and combined these intervals with a confidence measure [19]. This DE method led to a confidence measure for the distance between two points within the estimated interval.

To estimate the distances, one can invoke any DEF. However, it is much easier to use a symmetric DEF, which is what we will consider here. The training is performed the same as for the typical DE method. The training data set is used to determine the best parameters for the DEF, as well as to calculate the errors, which are assumed to be independent and random. The distribution of the errors is then used to produce a confidence measure.

The confidence measure $(1 - a)$ states how confident the estimator is that the true distance lies in the corresponding interval, where a $\in [0, 1]$. Let μ_ϵ and σ_ϵ^2 be the mean and the variance of the error function. The error function used here is:

$$\epsilon(X_i, X_j) = A(X_i, X_j) - d(X_i, X_j), \tag{2.33}$$

where $A(X_i, X_j)$ is the true distance and $d(X_i, X_j)$ is the estimated distance between points X_i and X_j. In what follows, we write $\epsilon(X_i, X_j) = \epsilon$, in the interest of simplicity. Given this error function, the confidence level for the estimate of $d(X_i, X_j)$ is:

$$P(\epsilon_1 \leq \epsilon(X_i, X_j) \leq \epsilon_2) = (1 - a) \quad \text{where } P(\epsilon_1 \leq \epsilon) = P(\epsilon \leq \epsilon_2) = a/2 \tag{2.34}$$

$$\text{for } min_{\epsilon_2 \epsilon_1}(\epsilon_2 - \epsilon_1),$$

In which the difference between the two bounding errors is minimized. This expression is hard to compute if the estimator function is not symmetric.

Since we only discuss the case for a symmetric ϵ, we let $z_1 = \frac{(\epsilon_1 - \mu_\epsilon)}{\sigma_\epsilon}$ and $z_2 = \frac{(\epsilon_2 - \mu_\epsilon)}{\sigma_\epsilon}$. If the error function is symmetric then $z_1 = -z_2$, and Equation (2.34) can be rewritten as:

$$\begin{aligned} P(\mu_\epsilon + z_1\sigma_\epsilon \leq \epsilon \leq \mu_\epsilon + z_2\sigma_\epsilon) &= (1 - a) \\ P(\epsilon - z_2\sigma_\epsilon \leq \mu_\epsilon \leq \epsilon - z_1\sigma_\epsilon) &= (1 - a) \end{aligned}$$

At this junction we substitute the mean and variance of the error function. Where $\mu_\epsilon = 0$ and σ_ϵ^2 is proportional to the variance of the variance of the DEF (i.e., $\sigma_\epsilon^2 = \sigma^2 A(X_i, X_j)$), we get :

$$
\begin{aligned}
P(\epsilon - z_2\sigma_\epsilon \leq 0 \leq \epsilon - z_1\sigma_\epsilon) &= (1-a) \\
P(z_1\sigma_\epsilon \leq \epsilon(X_i, X_j) \leq z_2\sigma_\epsilon) &= (1-a) \\
P(z_1\sigma_\epsilon \leq A(X_i, X_j) - d(X_i, X_j) \leq z_2\sigma_\epsilon) &= (1-a) \\
P(z_1\sigma_\epsilon + d(X_i, X_j) \leq A(X_i, X_j) \leq z_2\sigma_\epsilon + d(X_i, X_j)) &= (1-a)
\end{aligned}
$$

The probability of the true distance being in the interval $\{z_1\sigma_\epsilon + d(X_i, X_j), z_2\sigma_\epsilon + d(X_i, X_j)\} = \{A, B\}$ is (1-a). From this we can obtain the interval for the distance estimation $\{A, B\}$ and the confidence level 100(1-a)%.

This concludes our brief survey of DE.

2.3 Learning Automata

2.3.1 Introduction to Learning Automata

LAs are autonomous or self-operating machines that have the ability to learn. They are typically used to determine information in environments that have incomplete knowledge [23]. They are able to learn in these environments because the search for the information is actually preformed in a probability space and not the action space itself [24]. The probability space alluded to is traversed intelligently through the learning process. Typically, the process of learning for human beings requires some form of inference from experience, and a mechanism of decision making. In the case of LA, the experience, or input, usually comes in the form of a sequence of responses from the environment that the LA can observe. For a LA, the mechanism of decision making depends on its structure and the environment it is in. Throughout

this section we will examine LA that are relevant to the development and study of the problem that we are investigating.

The primary focus of this survey will be in the subsequent chapter on the family of Variable Structure Stochastic Automata (VSSA) because our implementation requires the LA to possess the superior properties of these machines. A VSSA has the ability to quickly learn from a stochastic "teacher", i.e., from an environment that can provide stochastic feedback. In the case of the DE, the LA must learn the ideal parameters that best suit the desired map from the "teacher", which is the set of training distances.

The study of LA dates back to the 1960's when Tsetlin used mathematical structures to model biological learning [25]. These structures and the corresponding method of learning were characterized using the terminology "Learning Automata" since 1974 by Narendra and Thathachar [26]. Since the inital proposal of LA, the field has grown to encompass numerous applications and theoretical problems. These applications include solutions in network and communications [27, 28, 29, 30], traffic control and scheduling [31, 32, 33, 34], training hidden Markov models [35], neural network adaptation [36], and intelligent vehicle control [37]. In addition to the diverse application of LA many researchers includes Narendra, Thathachar, Lakshmivarahan, Obaidat, Najim, Poznyak, Baba, Mason, Papadimitriou, Oommen, and their co-authors have conducted theoretical and qualitative studies on the performance of difference types of LA. Throughout this section, we will mention a few of their results in order to better evaluate and compare implementations.

2.3.2 Formal Definition of a LA

A LA consists of five components: a set of outputs or actions, a set of inputs, a set of states, a function that maps the current state and input to the next state, and a function that maps a current state to an output [38].

Definition 5. *A LA is defined by a quintuple* $\langle A, B, Q, F(.,.), G(.)\rangle$, *where:*

1. $A = \{\alpha_1.\alpha_2, .., \alpha_r\}$ *is the set of outputs or actions.*
2. $B = \{\beta_1, \beta_2, ..., \beta_m\}$ *is the set of inputs to the automaton.*
3. $Q = \{q_1, q_2, ..., q_s\}$ *is the finite set of states.*
4. $F(.,.) : QX\beta \longrightarrow Q$ *is the transition function. F is used to determine the new state of the LA based on the previous state and the input it receives from the environment. This mapping can either be deterministic or stochastic.*
5. $G(.) : Q \longrightarrow A$ *is the output function. The output is deterministically chosen based on the state of the LA. This mapping is, without loss of generality, deterministic.*

In general, q(t), $\beta(t)$, and $\alpha(t)$ denotes the state, the input, and the action chosen by the automaton, at time t, respectively. Throughout our study, we will consider the case of $m = 2$. The LA can thus receive two possible inputs from the environment, $B = \{0, 1\}$, a reward $\beta = 0$, or a penalty $\beta = 1$.

An automaton is said to be finite if Q, β, and A are all finite sets, which will be the focus of our study.

2.3.3 Environment

The environment is the external world in which the LA resides and operates. We consider the world in which the organism has all of its interactions with this environment. All of its possible actions, feedback, and input pertain to this environment. The environment dictates the possible actions and all of the feedback that the LA can receive. For this study, we will restrict our investigation to single LA and not consider the case of multiple interacting LA or a network of LA.

The environment (E) mentioned consists of three parts: the set of actions it offers to the LA, the output set of the Environment, and set of probabilities for each action.

Definition 6. *The environment is defined by a triple* $\langle A, C, B \rangle$*, where:*

1. $A = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ *is the set of actions.*

2. $B = \{\beta_1, \beta_2. \ . \ , \beta_m\}$ *is the set of outputs of the Environment. Again, we consider the case when* $m = 2$*, i.e., with* $\beta = 0$ *representing a Reward, and* $\beta = 1$ *representing a Penalty.*

3. $C = \{c_1, c_2, ..., c_r\}$ *is the set of penalty probabilities, where element* $c_i \epsilon C$ *corresponds to an input action* i*, and is defined as:* $c_i = P(\beta = 1 | \alpha = \alpha_i)$.

Now that both the LA and the environment have been formally defined, we will examine how they interact. Consider the process of learning in the presence of an "instructor". In order to learn, one must try an action and observe the result so as to gain information. This process may need to be repeated many times before one can come to a conclusion about the best action. This is exactly what a LA does. It performs a learning loop, shown in Figure 2.6, to gain information about the response it would get for the various actions. The LA's learning loop consists of the

LA performing (choosing) the action and observing the result in its environment. This learning loop is usually conducted a large number of times in order to gain sufficient information about the actions, while, all the time, the LA attempts to minimize the losses or penalties.

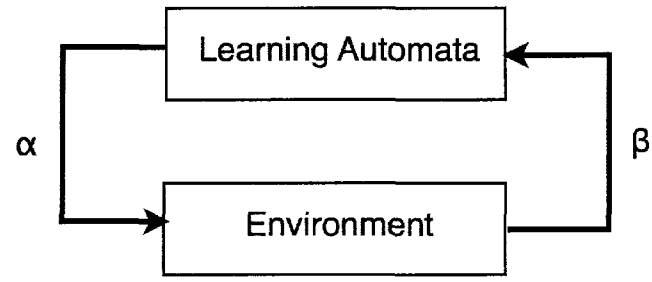

Figure 2.6: The Environment - LA Learning Loop

In the learning loop, the environment responds to the LA just as a stochastic teacher would. It does this by producing a reward or a penalty for the action that the LA has chosen. The environment produces a reward or a penalty for the action that the LA has chosen based on its set penalty probabilities, C.

2.3.4 Qualifying the Performance of a LA

To quantify the performance of a LA, we must first consider how an *optimal* LA would perform in any environment. After learning, an optimal LA would choose the action which yields the best results, or equivalently the action with the least penalty. The definition of an optimal LA is stated below in terms of $M(t)$, the less (penalty) at time "t".

Definition 7. *A LA is considered optimal if* $\lim_{t\to\infty} E[M(t)] = c^*$, *where* $c^* = \min\{c_i\}$.

The task of designing an optimal LA that operates in an unknown environment is, in fact, impossible due to the stochastic nature of the environments that we consider. It is always possible that after an arbitrarily large amount of learning, the LA is misled into choosing a suboptimal output by a sequence of erroneous responses. As a result, we define a marginally sub-optimal criterion for a LA that operates in a stochastic environment, that of it being ε-optimal.

Definition 8. *A LA is considered* ε*-optimal if:* $\lim_{n\to\infty} E[M(t)] < c^* + \varepsilon$, *where* $\varepsilon > 0$, *and can be arbitrarily small, by a suitable choice of some parameter of the LA.*

ε-optimality is the best achievable performance for a LA operating in a stochastic environment. This level of performance is the goal while designing a LA. It is, however, not always the case that a LA is ε-optimal. We must, therefore, also define other levels of performance.

Let us consider a LA that does not learn but chooses the output action based on a uniform probability distribution. Let $P(t) = [p_1(t), p_2(t), ..p_r(t),]^T$ be the action probability vector, with $\sum_{i=1}^{r} p_i(t) = 1 \; \forall \; t$. In this case $p_1(t) = p_2(t) = .. = p_r(t) = 1/r$.

The average penalty, M(t), for a LA at time "t" is:

$$\begin{aligned} M(t) &= E[\beta(t)|P(t)] \\ &= P[\beta(t) = 1|P(t)] \\ &= \sum_{i=1}^{r} P[\beta(t) = 1]p_i(t) \\ &= \sum_{i=1}^{r} c_i p_i(t). \\ \text{When } p_1(t) &= p_2(t) = ... = p_r(t) = 1/r, \\ M_0 &= 1/r \sum_{i=1}^{r} c_i. \end{aligned}$$

To reiterate, the expected penalty for a LA that does not learn is M_0. If a LA does learn to some degree, then as $t \to \infty, M(t) < M_0$. A LA that achieves this is said to be Expedient, and this is formally defined below.

Definition 9. *A LA is considered expedient if:* $lim_{t\to\infty} E[M(t)] < M_0$.

The condition of expediency is a very weak condition. It means that the LA is better than one which randomly chooses an output action. While it is an important measure, it is not a criterion that one can be satisfied with. Rather, we define an LA that displays a stronger condition referred to as Absolute Expediency.

Definition 10. *A LA is said to be Absolutely Expedient if* $E[M(t+1)|P(t)] < M(t)$, *implying that* $E[M(t+1)] < E[M(t)]$.

The condition of Absolute Expediency implies that the LA is able to learn, in the expected sense, at every time step. As a result the expected penalty is monotonically decreasing over time. The condition of absolute expediency is much stronger than the condition of expediency, and can be shown to be equivalent to ε-optimality. To find Absolute Expedient or ε-optimal LA is the goal when designing LA for a stochastic environment.

2.3.5 Classification of Learning Automata

While all LA have the same structure, $\langle A, B, Q, F(.,.), G(.)\rangle$, the properties of the LA that are based on these quantities can differ dramatically. These differences effect the performance of the LA, and the types of environments that they can ideally operate in. In this situation, we will describe some types of LA that are relevant to this study. The intent of this overview is to provide a better understanding of the *types* of LA so as to enable us to have detailed discussions in subsequent chapters.

Deterministic Learning Automata

Deterministic LA were first designed in 1961 by Tsetlin in modeling biological systems [25, 38, 39]. A deterministic LA has both a deterministic output function (G) and transition function (F) [38]. This means that given the current state of the LA, the output of the LA is well defined. Similarly, the transition function of a deterministic LA is fully determined by the state of the LA and the feedback from the environment.

Deterministic LA have been used in many areas from network traffic control to solving the nonlinear fractional knapsack problem. While deterministic LA are able to provide solutions to these problems, a stochastic approach is often necessary for obtaining improved solutions. Indeed, in the 1950s, before that field LA was formally proposed, Mosteller suggested that a *stochastic* method of learning would be beneficial over a deterministic model [40].

Stochastic Learning Automata

A stochastic process is one that is determined by both the process's predictable actions and by a random component that is innate to the machine. This means that the resultant dynamic process contains some amount of randomness. Formally, the

stochastic LA was first introduced by Tsetlin and Krylov [41]. Such a machine has either a stochastic output function (G) or a stochastic transition function (F) [38]. To ensure that the axioms of probabilities are satisfied, the output function (G) and transition function (F) are defined in terms of stochastic matrices given by Definition 11 [42].

Definition 11. *A stochastic transition matrix is a matrix that is used to define the transitions of the LA and is specified in terms of a Markov Chain, which is a real-value $N \times N$ matrix where:*

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{ns} \end{bmatrix}.$$

and has the following properties:

(i) $a_{ij} \geq 0$ *for* $1 \leq i, j \leq s$;

(ii) $\sum_{i=1}^{s} a_{ij} = 1$ *for* $1 \leq i \leq s$.

These transition matrices contain the probability of the underlying stochastic process moving from one state to another. The matrix that is used to define F, the transition function of the LA, consists of the conditional probability of moving to the next state given the previous state of the LA Thus, the entry in the i^{th} row and j^{th} column of the transition function is:

$$F_{ij}(t) = P[q_j(t+1)|q_i(t), \beta(t)]$$

Similarly the matrix that is used to define G, the output function, consists of the conditional probability of producing the current output given the current state of the LA. Hence, the entry in the j^{th} column of the i^{th} row of the output function is:

$$G_{ij}(t) = P[\alpha_j(t)|q_i(t)]$$

Fixed Structure Learning Automata

Fixed Structure LA have transition probabilities, $\{f_{ij}\}$, and output probabilities, $\{g_{ij}\}$, that remain constant over time ($f_{ij}(t) = f_{ij}$ and $g_{ij}(t) = g_{ij}, \forall t$). Note that all deterministic LA are fixed structure LA but the converse is not true. LA that are both stochastic and which have a fixed structure are called Fixed Structure Stochastic Automata (FSSA).

FSSA were the first type of LA to be thoroughly studied. The most common examples of these FSSA were presented by Tsetlin [25, 39], Krylov [43], and Krinsky [44], and have been used in applications such as game playing, and modeling biological learning systems [25, 39, 45, 46]. All of these FSSA are shown to be ε-optimal in some or all random environment [38].

Variable Structure Learning Automata

Vorontsova and Varshavskii introduced the concept of a variable structure LA [38], i.e., those that do not possess a fixed structure. In other words, the state transition probabilities or the action probabilities can change over time. Such a machine can be seen to be based on the input from the environment and an internal probability updating scheme. These factors, indeed, influence the action probability distribution

vector $P(t)$, which is the determining factor for the machine choosing a specific action. A LA that possesses both a variable structure and which is stochastic is said to be a Variable Structure Stochastic Automata (VSSA), as defined in Definition 12.

Definition 12. *A VSSA is a triple* $\langle A, B, T \rangle$, *where* A *is the set of actions,* B *is the set of responses from the environment to the LA, and* T *is the action probability updating scheme* $T : [0,1]^r XAXB \longrightarrow [0,1]^r$, *such that:*

$P(t+1) = T(P(t).\alpha(t), \beta(t))$, *where* $P(t)$ *is the machine's action probability vector.*

A VSSA is considered continuous if the components of $P(t)$ can assume any values in $[0, 1]$. If the range of the components of $P(t)$ is restricted to certain values in $[0, 1]$. we have a *Discrete* VSSA.

The sequence of states defined by the transition matrices of both FSSA and VSSA constitute a Markov process. The two main types of Markov chains that we encounter (with respect to learning algorithms) are defined in Definition 13 and Definition 14, i.e., absorbing and ergodic Markov chains respectively. In particular, we will consider the properties of the action probability updating scheme, T, so as to classify the VSSA-base learning algorithms. The classification of these learning algorithms is shown in Table 2.3.

Definition 13. *An Absorbing Markov Chain is one that contains at least one absorbing state, where if state* i *is an absorbing state, then* $P[q(t+1) = q_i | q(t) = q_i] = 1$.

Definition 14. *An Ergodic Markov Chain is a chain in which all the states are aperiodic and positive recurrent, where any state after learning it, is said to be positive recurrent if the expected time of returning to that state is finite.*

Table 2.3: Classification of Learning Algorithms

Property of T	Type of Learning Algorithm
Contains an absorbing state	Absorbing learning algorithms
T is an ergodic Markov chain	Ergodic learning algorithms
Linear function of $P(t)$	Linear learning algorithms
Nonlinear function of $P(t)$	Nonlinear learning algorithms

As mentioned, LA can be characterized as either absorbing, or an ergodic [2]. Absorbing learning algorithms are not ideal for non-stationary environments, because of the "possessive" nature of their absorbing states [2]. This is because the LA will become "stuck" in one of *these* states regardless of how the environment changes. Absorbing learning algorithms are typically ideal for stationary environments, because the LA can come to a permanent conclusion of what action it believes is best. As opposed to these, ergodic learning algorithms are better suited for non-stationary environments [2] because the LA can always move from one state to another given enough time. Also, the steady-state probability distribution of ergodic learning algorithms are independent of the initial state of the LA. This can be a big advantage when operating in complex environments or in environments where little information is known.

2.3.6 Action Probability Updating Scheme

The action probability updating scheme, T, is updated based on the input from the environment. In our formulation, we consider families of two-input LA, since they are the types used in our implementation. The two possible inputs are: $\beta \in \{0, 1\}$, where 0 is a positive response and 1 is a negative response. Using these inputs, the LA can reward or penalize an action. For example, if the LA chooses action a_i and receives a positive response 0, it may increase the likelihood of choosing that action

and consequently decrease the remaining action probabilities. As opposed to this, for a negative response, the probability of choosing the chosen action is decreased and the remaining probabilities are increased.

Consider the following action probability updating scheme, where there are two possible inputs and the current action is $\alpha(t) = \alpha_i$:

If $\beta = 0$:

$$p_j(t+1) = p_j(t) - g_j(P(t)) \quad \forall j \neq i \tag{2.35}$$

$$p_i(t+1) = p_i(t) + \sum_{k=1, k\neq i}^{r} g_k(P(t)) \tag{2.36}$$

$$\tag{2.37}$$

Else If $\beta = 1$:

$$p_j(t+1) = p_j(t) + h_j(P(t)) \quad \forall j \neq i \tag{2.38}$$

$$p_i(t+1) = p_i(t) - \sum_{k=1, k\neq i}^{r} h_k(P(t)) \tag{2.39}$$

$$\tag{2.40}$$

In the above both g and h must have the following properties:

- Have the domain $[0, 1]$,
- Are continuous over $[0, 1]$, and
- $0 < g_i(P(t)) < p_i(t) \leq 1$.

The updating scheme ensures that the action probability vectors, $P(t)$, remains a valid probability vector. There are three possible ways of updating T:

- Reward-Penalty (RP): In this case, the action probabilities are updated when the automaton is rewarded as well as penalized.
- Inaction-Penalty (IP): When the automaton is penalized, the action probability vector is updated, and when the automaton is rewarded, the probabilities are neither increased nor decreased.
- Reward-Inaction (RI): The action probability vector is updated whenever the automaton is rewarded, and is unchanged whenever the automaton is penalized.

Most VSSA in the literature use linear learning algorithms. We also, in the subsequent chapters, propose using linear learning algorithms to resolve the DE problem. There are three types of linear learning algorithms: Linear Reward-Penalty (L_{RP}), Linear Reward-Inaction (L_{RI}), and Linear Inaction-Penalty (L_{IP}). All of theses updating schemes use the following functions:

$$g_j(P(t)) = ap_j(t) \tag{2.41}$$

$$h_j(P(t)) = b(1 - p_j(t)) \tag{2.42}$$

While all linear update schemes use the same functions, they all have different characteristics, which differ depending on the environment in which they operate, and on the values of 'a' and 'b'.

Linear Reward-Penalty Scheme (L_{RP})

The L_{RP} scheme uses both rewards and penalties to update the action probabilities. More specifically, the parameters a and b are non-zero in $g(P(t))$ and $h(P(t))$ respectively. The L_{RP} scheme is ergodic, due to the fact that changes are made for both

rewards and penalties [47]. Because the L_{RP} is ergodic it is ideal for non-stationary environments. The long term expected penalty of the L_{RP} scheme (in which the updates are symmetric, i.e., $a = b$) is given by:

$$lin_{t->\infty} E[M(t)] = \frac{2c_1c_2}{c_1+c_2} < \frac{c_1c_2}{2} = M_o$$

This implies that the *symmetric* L_{RP} LA is expedient regardless of the initial conditions. In order to improve the performance of the L_{RP} scheme, the parameters a and b must be chosen appropriately. If $a = b$, the update scheme is called a symmetric L_{RP} scheme. This type of update scheme is only expedient. However if, $b > 0$ and $a >> b$ the L_{RP} scheme becomes Linear Reward- ϵ Penalty scheme ($L_{R-\epsilon P}$) which is ϵ - optimal in all random environments [48].

Linear Reward-Inaction Scheme (L_{RI})

A linear learning scheme is said to be L_{RI} when $b = 0$ and $a < 1$. Observe that the L_{RI} scheme changes the probabilities only when it receives a positive response from the environment. If a negative response is received, the LA performs no update to the action probabilities and continues to choose the next action with the same probability vector.

The L_{RI} scheme is an absorbing learning algorithm. It is therefore not useful for non-stationary environments because it will not respond properly to a changing environment. However, it is ϵ - optimal in all stationary environments [48].

Linear Inaction-Penalty Scheme (L_{IP})

If $a = 0$ and $b > 0$, the linear learning scheme is a L_{IP} scheme, which is an ergodic learning algorithm. It is more appropriate for non-stationary environments, because

the action probability vector can never become an unit vector. The long term expected penalty approaches $\frac{1}{r-1}$ [47], and thus the L_{IP} scheme is only expedient [47]. Although such a scheme may appear to be weak, it can be useful in a fast changing environment in which the penalty probabilities are very large.

The performance of these different types of linear learning algorithms are well reported and summarized in Table 2.4 [2].

Table 2.4: Properties of Continuous Learning Schemes from [2]

Learning Scheme	Learning Parameters	Usefulness	Optimality	Ergodic / Absorbing (Environment)
L_{RI}	$a > 0$ $b = 0$	Good	ϵ - optimal as $a \rightarrow 0$	Absorbing (Stationary)
L_{IP}	$a = 0$ $b > 0$	Very Bad	Not even expedient	Ergodic (Non-stationary) (Non-stationary)
L_{RP} (Symmetric)	$a > 0$ $a = b$	Bad	Never ϵ - optimal	Ergodic (Non-stationary)
$L_{R-\epsilon P}$	$a > 0$ $b << a$	Good	ϵ - optimal as $a \longrightarrow 0$	Ergodic (Non-stationary)

Chapter 3

The ATS for Distance Estimation

In this chapter, we present the ATS for DE. This is done by first outlining the general ATS for any environment. Then, we introduce the adaptation required for using it in the DE domain. These adaptations are required for the ATS to function within the constraints of the corresponding environments associated with DE, but do not change the underlying structure and strategy of the ATS itself.

3.1 The Adaptive Tertiary Search

The ATS, from Oommen and Raghunath, was presented in the context of the Stochastic Point Location problem in [17]. The advantage of the ATS is that it is not a hill climbing search, and therefore overcomes the problems of being dependent on a starting point and a step size. In [17], the ATS was applied to a stochastic environment (where a stochastic environment is one which can inherently provide an incorrect response). The ability of the ATS to function in environments of this type makes it ideal for the DE problem.

Oommen and Raghunath used ATS to determine a point, θ^*, in a bounded interval and a resolution of accuracy, where the oracle or environment is modeled as a "Stochastic Teacher" [17]. For the environment to be considered a "Teacher", it must provide a correct response with a probability greater than 0.5 [18]. The environment for The Stochastic Point Location problem gives feedback about the location of the point in question.

To determine θ^* within the resolution of accuracy, the original search interval is divided into three equal and disjoint subintervals, Δ^i, where $i = 1...3$. The subintervals are searched using a two-action LA. The LA returns the position of θ^* from that subinterval, $O^i \in \{Left, Right, Inside\}$. From these outputs, a new search interval is obtained which is based on the decision table given in Table 3.1. This is repeated until the search interval is smaller than the resolution of accuracy. The search interval will be reduced to yield the required resolution within a finite number of epochs because the size of search interval is non-increasing [17]. After the search interval has been sufficiently reduced, the midpoint of the find search interval is returned as the estimate for θ^*. The ATS algorithm can be seen in Algorithm 3.

O^1	O^2	O^3	New Sub-Interval
Inside	Left	Left	Δ^1
Left	Left	Left	Δ^1
Right	Inside	Left	Δ^2
Right	Left	Left	$\Delta^1 \cup \Delta^2$
Right	Right	Inside	Δ^3
Right	Right	Left	$\Delta^2 \cup \Delta^3$
Right	Right	Right	Δ^3

Table 3.1: Decision Table

An example of how the ATS works is shown below.

3.1.1 Example of ATS

In this example we show four epochs of the ATS. This example is taken from [17] and has the initial search interval of $[0, 1]$. Here, the value of $\theta = 0.9123$.

Step 1: $\triangle = [0, 1]$

Partitions:

$\triangle^1 = [0, 0.333]$

$\triangle^2 = [0.333, 0.667]$

$\triangle^3 = [0.667, 1]$

Automata Results:

$O^1 = Right$

$O^2 = Right$

$O^3 = Right$

New Search Interval:

$\triangle^3 = [0.667, 1]$

Step 2: $\triangle = [0.667, 1]$

Partitions:

$\triangle^1 = [0.667, 0.778]$

$\triangle^2 = [0.778, 0.889]$

$\triangle^3 = [0.889, 1]$

Automata Results:

$O^1 = Right$

$O^2 = Right$

$O^3 = Inside$

New Search Interval:

$\triangle^3 = [0.889, 1]$

Step 3: $\triangle = [0.889, 1]$

Partitions:

$\triangle^1 = [0.889, 0.926]$

$\triangle^2 = [0.926, 0.963]$

$\triangle^3 = [0.963, 1]$

Automata Results:

$O^1 = Right$

$O^2 = Left$

$O^3 = Left$

New Search Interval:

$\triangle^1 \cup \triangle^2 = [0.889, 0.963]$

Step 4: $\triangle = [0.889, 963]$

Partitions:

$\triangle^1 = [0.889, 0.914]$

$\triangle^2 = [0.914, 0.938]$

$\triangle^3 = [0.938, 0.963]$

Automata Results:

$O^1 = Right$

$O^2 = Left$

$O^3 = Left$

New Search Interval:

$\triangle^1 \cup \triangle^2 = [0.889, 0.938]$

This yields the current estimate to be $\theta = 0.9135$.

Algorithm 3 ATS Algorithm

Input: The Resolution

Output: Estimate of θ^*

Method:

1: **repeat**
2: $(\Delta^1, \Delta^2, \Delta^3) \leftarrow Getpartitions(\Delta)$
3: **for** $j \leftarrow 1$ to 3 **do**
4: Get position of θ^* from LA_j
5: **end for**
6: $\Delta \leftarrow$ Get new search interval from Table 3.1
7: **until** Size of Interval $<$ resolution

End Algorithm

3.2 ATS for Distance Estimation

The ATS proposed by Oommen and Raghunath [17] was used to solve the Stochastic Point Location problem, and also later for parameter learning from a stochastic teacher/compulsive liars [17, 18]. For both of these problems, one had to determine only a single unknown parameter. However, the distance estimation problem may require many parameters to be found simultaneously. In order to adapt the ATS to find more than a single parameter, we must specify both the process of updating multiple search intervals and the issue of how the set of LA interact with the new environment.

3.2.1 Updating Search Intervals

The *order* of updating the search intervals must be considered when finding multiple unknown parameters. If this is not done correctly, it may result in the premature reduction of a search interval. In the stochastic point location problem, the subintervals were searched using the LA, after which the search interval was updated. This

order of executing the searching and pruning must also be maintained while searching for two parameters, k and p, simultaneously. In other words, all the subintervals must be searched before any interval is updated. Each search interval undergoes the same process as in the example for the single parameter ATS. The only difference is that the search intervals are updated simultaneously. This order is shown below in Algorithm 4.

Algorithm 4 Updating the Search Intervals
Method:

1: **repeat**
2: **for** $j \leftarrow 1$ to 3 **do**
3: execute LA^j for k
4: execute LA^j for p
5: **end for**
6: getNewInterval for k - From Table 3.1
7: getNewInterval for p - From Table 3.1
8: **until** Size of Interval $<$ resolution
End Algorithm

The set of LA functions in the same manner as in [17], except for how it deals with additional parameters. When the LA is learning information about k, it uses values of p from within its *current* search interval. As a result, each LA must know the *current* search interval of *all* the other parameters.

This process of searching for multiple parameters can be done in parallel by assuming that for each learning loop, the other parameter's value is either the maximum or the minimum of its current search interval. This is a consequence of the monotonicity of the DEFs, as discussed in Section 3.2.3.

3.2.2 The Corresponding LA

Each LA is given two inputs, the parameter that it is searching for and all the search intervals. Each LA is required to output the relative location of the parameter in question. It does this by producing a decision (Left, Right or Inside) according to *its* final belief after communicating with its specific environment.

The LA starts out with a uniform belief, 50% for both "Left" and "Right". It then makes a decision based on its current belief. If the decision is "Left", then the LA picks a point in the left half of the interval at random; otherwise, the decision is "Right" and the point is chosen from the right half of the interval. Once the decision is made, the LA asks the environment for a response. The LA uses a Linear Reward Inaction (L_{RI}) update scheme, and so the current belief is only updated if the environment provides a positive response.

The LA and the environment repeat this loop of making a decision and updating the current belief for a large number of iterations. After the LA and the environment are done communicating, the LA produces its output as per Algorithm 5. If the LA's belief of "Right" is greater than $1 - \epsilon$, the parameter in question is to the right side of the current search interval, and so its output is "Right". If the belief of "Left" is greater than $1 - \epsilon$, the parameter is to the left, and the LA's final decision is "Left". Otherwise the LA does not have a belief greater than $1 - \epsilon$ that the parameter is to the "Right" or the "Left", in which case the LA's output is "Inside". The entire LA algorithm is formally given in Algorithm 5.

3.2.3 The Corresponding Environment

Each LA requires feedback from a specific environment. This feedback informs the LA if it has made the correct decision, i.e., choosing the right or left half of the subinterval.

Algorithm 5 LA algorithm

Input: Parameter to be found, Search interval, θ_R, N_∞

Output: Decision= { left, right, or inside }

Method:

1: **for** $i = 1$ to N_∞ **do**
2: currentAction $\leftarrow$ chooseAction
3: Feedback $\leftarrow$ getEnvironmentsResponse(currentAction) - From Algorithm 6
4: **if** Feedback==Agree **then**
5: **if** currentAction==Left **then**
6: $pRight = pRight * (1 - \theta_R)$
7: $pLeft = 1 - pRight$
8: **end if**
9: **if** currentAction==Right **then**
10: $pLeft = pLeft * (1 - \theta_R)$
11: $pRight = 1 - pLeft$
12: **end if**
13: **end if**
14: **end for**
15: **if** $pRight > 1 - \epsilon$ **then**
16: return Right
17: **else if** $pLeft > 1 - \epsilon$ **then**
18: return Left
19: **else**
20: return Inside
21: **end if**

End Algorithm

It is easy to obtain this answer because it only involves a single parameter at a time.

Consider the DEF in Equation (2.11). This DEF can be simplified into two equations, Equation (3.1) and Equation (3.2).

$$F(k) = kX \tag{3.1}$$

$$F(p) = X \sum_{i=1}^{i=n} (|x_0^i - x_1^i|^p)^{1/p}. \tag{3.2}$$

In Equation (3.1) and Equation (3.2), X is always positive, and consequently, both $F(k)$ and $F(p)$ are both monotonic for all k and for all $p > 1$. This property allows the oracle to respond according to Algorithm 6 when finding k.

Algorithm 6 Environments Response
Method:

```
1: if Choose = Left and Estimated distance >= True distance then
2:     return Agree
3: else if Choose = Right and Estimated distance <= True distance then
4:     return Agree
5: else
6:     return Disagree
7: end if
```

End Algorithm

The estimated distance uses either the max or min functions from within the search interval of the other parameters. Whether the LA uses the max or min itself depends on the choice that the LA has made. If the parameter, k, is chosen from the left half of the search interval and the actual distance is larger than or equal to the estimated distance, the environment agrees. In this case, the unknown parameter p would be $p \in \{p_{min}, p_{max} | min\{F(p)\}\}$. In other words, the p value becomes the value which causes the function, $F(p)$, to be minimized. Alternatively, if the parameter, k, is

chosen from the right half of the search interval and the actual distance is smaller than or equal to the estimated distance, then the environment gives a positive response, where p is determined by: $p = \{p_{min}, p_{max} | max\{F(p)\}\}$. Otherwise the environment gives a negative response.

The same is also true for the parameter p. The environment gives a positive response, if the LA chooses the left half of the search interval and the actual distance is smaller than or equal to the estimated distance, where, $k = \{k_{min}, k_{max} | min\{F(k)\}\}$, or if the LA chooses the right half of the search interval and the actual distance is larger than or equal to the estimated distance where $k = \{k_{min}, k_{max} | max\{F(k)\}\}$. Otherwise the environment gives a negative response.

3.3 Summary

In this chapter, we have outlined the ATS for DE. For this problem, the ATS maintains its underlying structure. The only differences lies with the process of updating multiple search intervals and how the LA interacts with the new environment. The problem of updating multiple search intervals is solved by maintaining the order by searching the intervals first and then pruning them. Both the LA and the environment have been fully defined in Sections 3.2.2 and 3.2.3.

Chapter 4

2D Distance Estimation Using the ATS

4.1 Introduction

In this chapter, we present the results of the ATS on DE for 2-dimensional data. These results were obtained from using three types of data. These data sets consisted of a list of co-ordinates for a set of points, and a table of inter-point road distances. The first type of data that the ATS was tested on was artificially generated and noiseless. This was done so as to verify the implementation of the ATS in the new environment. The second type of data was artificially generated but noisy. This was used to observe the performance of the ATS in such a noisy setup. The final type of data was taken from real-world road distances. This data was used to confirm the practical use of the ATS.

Each type of data was tested using three DEFs: the weighted euclidean DEF, the L^p DEF, and the weighed L^p DEF. The first two DEFs had only a single parameter;

therefore the ATS was analogous to that of the stochastic search on a line. The last DEF was a combination of the previous DEFs and had two parameters. This DEF was thus more complex and further extends the applications of the ATS.

4.2 Testing and Results

In this section, we present the results for the 2D distance estimation using the ATS. We show that this method of estimation works for three different DEFs where, as mentioned, the first two DEFs, Equation (2.9) and Equation (2.10) each contained only a single parameter that must be found, k and p respectively. The last DEF, Equation (2.11), contained two parameters, k and p. To compare the results we used four GoF measures. The first three, RAD, NAD, and SD were presented in Section 2.1.2, in Equations (2.2),(2.3) and (2.4) respectively. The last GoF measure was the Expected Percent (EP) error for each distance in the region under consideration. The EP error was given by Equation (4.1):

$$EP = \frac{1}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{|A(P_i,P_j) - \pi(P_i,P_j|\Theta)|}{A(P_i,P_j)}. \tag{4.1}$$

The SD and NAD GoF functions were useful for comparing the results against the methods reported in the literature as they are some of the most commonly-used GoF functions; however, the RAD and the EP were useful when looking at the values by themselves. The *RAD* is, in fact, the percentage error for the entire region. The EP also has a useful physical meaning: it is the expected error for an estimated distance in the region in question.

4.2.1 Results for the Noiseless Data

Experimental Setup

The first type of data was noiseless. The noiseless data was constructed by randomly generating points in the region and it employed *known* values of k and p to generate the "true" distances from the DEF being tested. These known values which are used to create the data sets will be called the "Actual Values". The consequence of creating the noiseless data in this manner is that the inter-city distances perfectly fit the DEF. The primary purpose of this data set was to show that under ideal conditions, the ATS can always determine the optimal parameter.

Each DEF was tested on three noiseless data sets. The first set had 29 points, the second had 75, and the third had 100 points. We show below examples of runs for the weighted Euclidean DEF and the weighted L^p DEF. We reported the accuracy over 100 runs of each data set. Each LA's reward parameter (θ_R) was set to 0.02. We used an ϵ value of 0.1 and $N_\infty = 2,500$. These values were chosen through preliminary testing and lie within the generally excepted range of values use for these types of LA. It should also be noted that the algorithm in not overly sensitive to these values because the believes of the LAs are only used to update the search interval and not the final parameter. As a result, only requiring a belief of 90% can help convergence times, as the L_{RI} LA converge slowly as it nears 1.

Weighted Euclidean DEF

In this case we examined the weighted Euclidean DEF (Equation (2.10)), which has only a single parameter, k. The reason for considering such a simple DEF was to demonstrate how the ATS functions in the new environment pertaining to DE. Here we will show that the ATS always finds the optimal parameter for a DEF that contains

only a single parameter in a noiseless environment. Figure 4.1 shows a pictorial representation of the ATS in this environment. Table 4.1 shows a typical ATS in a noiseless environment and the errors associated with that particular run.

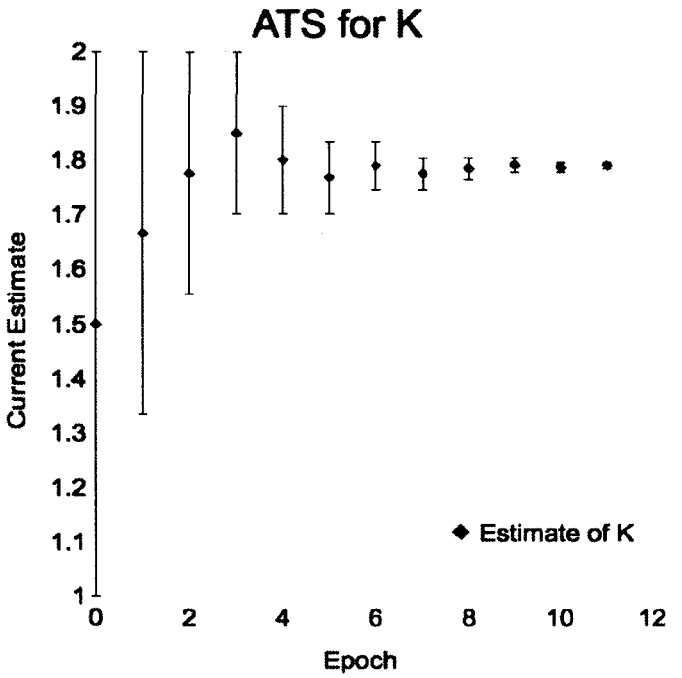

Figure 4.1: Blue lines represent the current search interval and the blue diamonds represent the current estimate of k.

The ATS in Table 4.1 converged to the actual values used to create the noiseless data set in only 36 epochs. After 4 epochs, the ATS's search interval was approximately 0.1, and after 10, the search interval was smaller than 0.01. Since the ATS converged to the actual values, the errors were all exactly zero. We emphasize that this is typical for the ATS search for this DEF in this setting. Table 4.2 shows the

Epoch	Current Search Interval	Epoch	Current Search Interval
1	[1.6666667, 2.0]	19	[1.8498871, 1.8501127]
2	[1.7777778, 2.0]	20	[1.8499624, 1.8500626]
3	[1.7777778, 1.925926]	21	[1.8499624, 1.8500292]
4	[1.7777778, 1.8765433]	22	[1.8499846, 1.8500292]
5	[1.8106996, 1.8765433]	23	[1.8499846, 1.8500143]
6	[1.8326474, 1.8765433]	24	[1.8499945, 1.8500143]
7	[1.8326474, 1.8619113]	25	[1.8499945, 1.8500078]
8	[1.8424021, 1.8619113]	26	[1.8499945, 1.8500034]
9	[1.8424021, 1.8554082]	27	[1.8499975, 1.8500034]
10	[1.8467375, 1.8554082]	28	[1.8499975, 1.8500015]
11	[1.8467375, 1.852518]	29	[1.8499988, 1.8500015]
12	[1.8486643, 1.852518]	30	[1.8499988, 1.8500006]
13	[1.8486643, 1.8512334]	31	[1.8499994, 1.8500006]
14	[1.8495207, 1.8512334]	32	[1.8499998, 1.8500004]
15	[1.8495207, 1.8506625]	33	[1.8499998, 1.8500001]
16	[1.8495207, 1.8502818]	34	[1.8499999, 1.8500001]
17	[1.8497744, 1.8502818]	35	[1.85, 1.8500001]
18	[1.8497744, 1.8501127]	36	[1.85, 1.85]
Estimated k		$k = 1.85$	
Actual k		$k = 1.85$	
AD		0.0	
NAD		0.0	
RAD		0.0	
EP		0.0	

Table 4.1: Example run of the ATS with the weighted Euclidean DEF on the noiseless data.

average error for 100 ATSs for the weighted Euclidian DEF in such noiseless environments. Again, the reader should note that all of the errors are zero because the ATS finds the actual value *every time*.

Data Set Size	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
AD	0.0	0.0	0.0	0.0	0.0	0.0
NAD	0.0	0.0	0.0	0.0	0.0	0.0
RAD	0.0	0.0	0.0	0.0	0.0	0.0
EP	0.0	0.0	0.0	0.0	0.0	0.0

Table 4.2: Results for 100 runs of the ATS with the weighted Euclidean DEF on the noiseless data sets.

L^p DEF

We now consider the L_p DEF from Equation (2.9). This DEF also has a single parameter, and consequently, we believe that there is no need to display a typical ATS since it is very similar to the one shown in Table 4.1. Table 4.3 shows the average errors and standard deviation for 100 ATS for each data set. The result is again the same as for the weighted Euclidian DEF. All the errors are exactly equal to zero, because the ATS always converged to the actual value of p.

Data Set Size	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
AD	0.0	0.0	0.0	0.0	0.0	0.0
NAD	0.0	0.0	0.0	0.0	0.0	0.0
RAD	0.0	0.0	0.0	0.0	0.0	0.0
EP	0.0	0.0	0.0	0.0	0.0	0.0

Table 4.3: Results for 100 runs of the ATS with the L^p DEF on the noiseless data sets.

Weighted L^p DEF

The final DEF which we studied was the weighted L^p DEF from Equation (2.11). This DEF has two parameters, k and p. As a result, the ATS had to search for both the parameters in the joint space simultaneously. While this can be done, it does affect the minimum resolution. This is because the reduction of the search interval for each parameter is depended on the resolution of the search interval of the other parameter. In the example, ATS for the weighted Euclidian DEF, the resolution was set to 0.0000001; however, for this ATS the resolutions for k and p were set to 0.00001 and 0.001 respectively. Table 4.4 shows an example of a typical ATS for the weighted L^p DEF in a noiseless environment. The errors for this ATS were almost (although not exactly) zero, which is due to the larger resolutions that we had employed.

Figure 4.2 shows a pictorial representation of an ATS for multiple parameters in a noiseless environment. In the example shown in Table 4.4, the search interval for parameter k converged faster than the search interval for p. It took 7 epochs for the magnitude of k's search interval to be smaller than unity. Alternatively, it took 17 epochs for p's search interval to be smaller than unity. This may be due to the fact that p's search interval was first pruned on the 8^{th} epoch.

We again emphasize that the errors in the example run are typical for the ATS for the weighted L^p DEF in a noiseless environment. This is confirmed in Table 4.5, where we report the average errors and standard deviation of 100 runs of the ATS. It should be noted the "Actual" values are always contained in the final search intervals. We also mention that if the resolution is set too small, the ATS will not be able to reduce the search interval to the required size causing it to continue, until the process is manually terminated. While we had set the resolution to be the convergence requirement, one could alternately use the number of epochs as the

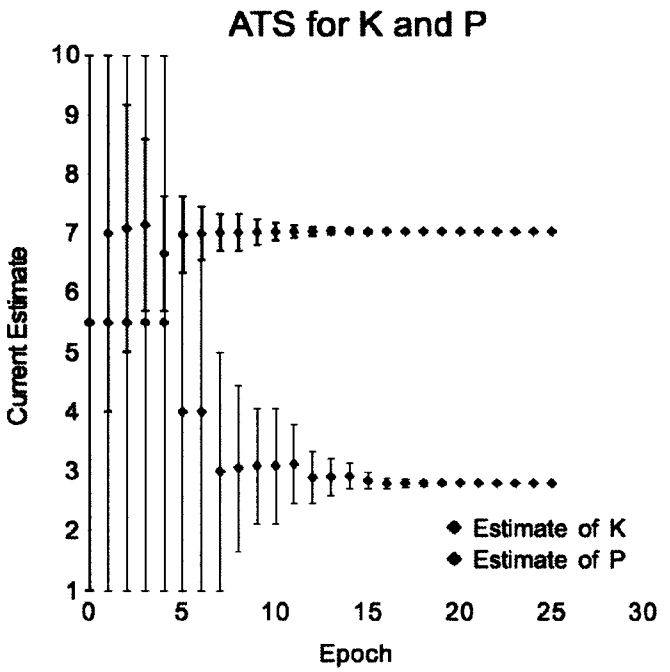

Figure 4.2: Blue and red lines represent the current search interval for the respective parameters and the diamonds represent the current estimate of the parameters.

convergence requirement. Under these circumstances the resolution would vary from run to run.

Epoch	$k's$Current Search Interval	$p's$Current Search Interval
1	[2.5, 8.75]	[1.0, 10.0]
2	[3.5416667, 7.881944]	[1.0, 10.0]
3	[4.265046, 7.279128]	[1.0, 10.0]
4	[4.767393, 6.8605056]	[1.0, 10.0]
5	[5.1162453, 6.5697956]	[1.0, 10.0]
6	[5.358504, 6.3679137]	[1.0, 10.0]
7	[5.5267386, 6.227718]	[1.0, 10.0]
8	[5.6435685, 6.1303596]	[1.0, 10.0]
9	[5.7247005, 6.06275]	[2.5, 8.75]
10	[5.9500666, 6.06275]	[3.5416667, 7.881944]
11	[5.9876275, 6.06275]	[4.265046, 7.279128]
12	[6.012668, 6.06275]	[5.26974, 7.279128]
13	[6.029362, 6.06275]	[5.26974, 7.279128]
14	[6.029362, 6.0516205]	[5.939536, 7.279128]
15	[6.033072, 6.048529]	[6.1628013, 7.093074]
16	[6.033072, 6.043377]	[6.317847, 6.963869]
17	[6.036507, 6.043377]	[6.425517, 6.8741436]
18	[6.037652, 6.042423]	[6.500288, 6.8118343]
19	[6.0384474, 6.0417604]	[6.552212, 6.768564]
20	[6.0384474, 6.040656]	[6.588271, 6.7385154]
21	[6.0391836, 6.040656]	[6.613312, 6.717648]
22	[6.0396743, 6.040656]	[6.630701, 6.703157]
23	[6.0396743, 6.040329]	[6.642777, 6.693094]
24	[6.0397835, 6.040238]	[6.651163, 6.6861053]
25	[6.0397835, 6.0400863]	[6.6569867, 6.681252]
26	[6.0398846, 6.0400863]	[6.661031, 6.6778817]
27	[6.039952, 6.0400863]	[6.6638393, 6.6755414]
28	[6.039952, 6.0400414]	[6.6657896, 6.673916]
29	[6.039982, 6.0400414]	[6.667144, 6.672787]
30	[6.039982, 6.0400214]	[6.668084, 6.6720033]
31	[6.039982, 6.040008]	[6.6687374, 6.671459]
32	[6.0399904, 6.040008]	[6.669191, 6.671081]
33	[6.039996, 6.040008]	[6.669506, 6.6708183]
34	[6.039996, 6.0400043]	[6.669506, 6.670381]
Estimated Values	$k = 6.04$	$p = 6.669944$
Known Values	$k^* = 6.04$	$p^* = 6.67$
SD	$6.775008639660095E-8$	
NAD	$6.862434467784457E-4$	
RAD	$2.3117311334956413E-7$	
EP	$2.407871743082266E-7$	

Table 4.4: Example run of the ATS with the *weighted* L^p DEF on the noiseless data.

Data Set	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
SD	2.06×10^{-7}	4.69×10^{-7}	1.04×10^{-7}	1.57×10^{-7}	4.17×10^{-7}	8.77×10^{-7}
NAD	6.84×10^{-5}	1.05×10^{-4}	9.41×10^{-4}	9.47×10^{-4}	0.0017	0.0019
RAD	1.66×10^{-7}	2.54×10^{-7}	3.27×10^{-7}	3.34×10^{-7}	3.23×10^{-7}	3.62×10^{-7}
EP	1.69×10^{-7}	2.59×10^{-7}	3.30×10^{-7}	3.32×10^{-7}	3.58×10^{-7}	3.93×10^{-7}

Table 4.5: Results for 100 runs of the ATS with the weighted L^p DEF on the noiseless data sets.

4.2.2 Results for the Noisy Data

Experimental Setup

We now consider the more realistic case of testing the ATS on noisy data sets. The sets were constructed in the same manner as the noiseless data sets, except noise was added to the true distances. Thus, to create the noisy data, an additional term was added to each distance. This additional term was proportional to the magnitude of the distance, where the proportion was based on a Gaussian distribution.

$$Noise = \frac{TrueDistance}{10} * X \tag{4.2}$$

where, X is a random gausian variable, N(0,1)

Due to the noise that was added to the distances, the "Actual" values, that are used to construct the data, was not necessarily the "Benchmark" values to predict the distances for the data set. For these noisy data sets, the "Benchmark" or optimal parameter was then found using a simple hill-climbing search that minimized the SD error. This parameter was then compared to the "Estimated" values, in order to verify the accuracy of the ATS's estimate.

The Hill-climbing search we applied started at the values found by the ATS search. It should be noted that other starting points were tested and produced the same results; however, the search took longer to converge. Starting from the values found by the ATS, the Hill-climbing search compared its current value (say, θ) to the $\theta + \epsilon$ and $\theta - \epsilon$, and moved to the value that minimized GoF function. This process is repeated until the current value θ minimizes the GoF function. The Hill-climbing search had to calculate the value of the GoF at every time step, and as a result, this search was computationally, very expensive.

Each of the DEFs was trained on 70% of the data set, and testing was conducted on the remaining 30%. The points in the training set were randomly chosen from the whole data set. While we did not follow a rigorous cross-validation process, we believe that the error obtained is a good representation of the performance of the corresponding scheme, and the only major difference can be seen in the larger standard deviation. This was done for all three data sets, where the first set had 29 points, the second had 75 and the third had 100 points. One example of the ATS is presented for the weighted Euclidian DEF and the weighted L_p DEF for the data set of size 75. The overall characteristics of all three DEFs were examined for the three different sets of noisy data. These characteristics were determined by examining the accuracy of each ATS over 100 runs. Observe that for each of these 100 executions of the ATS, the noise and "Actual values" change.

Weighted Euclidian DEF

The errors of the "Benchmark" and "Estimated" values shown in Table 4.7 were almost equal. While the "Benchmark" value performed better on the training data than the "Estimated" value, the "Estimated" value performed marginally better than

the "Benchmark" value when applied to the testing data. The difference between the errors in both cases are "small"; however, it is interesting that the ATS performed better on the testing data than the optimally trained parameter.

Epoch	Current Search Interval	Epoch	Current Search Interval
0	[1.0, 2.0]	35	[1.3524342, 1.3528624]
1	[1.0, 1.6666667]	43	[1.3527197, 1.3528624]
2	[1.2222222, 1.6666667]	47	[1.3527672, 1.3528306]
3	[1.2222222, 1.5185186]	62	[1.3528095, 1.3528306]
4	[1.2222222, 1.4197531]	82	[1.3528095, 1.3528236]
5	[1.2880658, 1.4197531]	93	[1.352819, 1.3528236]
6	[1.3319615, 1.3904892]	101	[1.3528205, 1.3528225]
7	[1.3514707, 1.377483]	132	[1.3528218, 1.3528225]
8	[1.3514707, 1.3601415]	146	[1.3528223, 1.3528225]
14	[1.3514707, 1.3543609]	149	[1.3528224, 1.3528225]
34	[1.3524342, 1.3537186]	199	[1.3528224, 1.3528224]
Estimated k		1.3528224	
Actual k		1.359	
Benchmark k		1.346091	

Table 4.6: Example run of the ATS with the weighted Euclidian DEF on the noisy data. The number of points in the set was 75.

Table 4.6 is an example run for the ATS for the weighted Euclidian DEF. For clarity, we have only displayed the epochs in which the search interval was changed. After 34 epochs, the search interval had a magnitude of about 0.01 and the ATS converged in 199 epochs. This is longer than in the noiseless case.

The "Actual" value of k that was used to create the data set for this run was 1.359. This is quite close to both the "Estimated" value and the "Benchmark" value, which are k, 1.3539926 and 1.344969 respectively. The errors associated with both the "Estimated" and the "Benchmark" values for the training data and the testing data are shown in Table 4.7.

Table 4.8 shows the average errors of 100 ATS for the weighted Euclidian DEF on each of the noisy data sets. The errors for the "Benchmark" value and "Estimated" values of k produce very similar errors; there was less than 0.1% difference between

Error Type	For Estimated Value: $k = 1.3528224$	For Benchmark Value: $k = 1.346091$
	Training	Training
SD	427.73	425.52
NAD	105.63	105.36
RAD	0.0581	0.0582
EP	0.0597	0.0595
	Testing	Testing
SD	24.16	24.19
NAD	6.60	6.65
RAD	0.0537	0.0545
EP	0.0550	0.0554

Table 4.7: Results for the typical run in Table 4.6.

the "Benchmark" and "Estimated" values for both the RAD and the EP errors. These results show the success of the ATS for DE in a noisy environment.

Data Set Size	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
SD	239.31	82.55	70.71	16.78	76.57	21.16
NAD	1.99	0.4636	14.68	3.29	10.71	2.30
RAD	0.0546	0.0131	0.0572	0.0123	0.0564	0.0123
EP	0.0553	0.0129	0.0580	0.0130	0.0564	0.0121
	Benchmark		Benchmark		Benchmark	
SD	240.80	83.54	70.55	16.64	76.37	20.81
NAD	1.99	0.4596	14.66	3.27	10.69	2.29
RAD	0.0548	0.0132	0.0574	0.0124	0.0565	0.0124
EP	0.0553	0.0128	0.0579	0.0129	0.0563	0.0121

Table 4.8: Results for 100 runs of the ATS with the weighted Euclidian DEF on the noisy data sets.

L^p DEF

We again ran the ATS 100 times with the L_p DEF on the noisy data. The results are shown in Table 4.9. These results are similar to the weighted Euclidian DEFs,

since both the "Benchmark" and "Estimated" values of k produced almost identical testing errors. The difference between the "Benchmark" and "Estimated" values for both the RAD and the EP errors were less than 0.1%.

Data Set Size	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
SD	330.38	89.67	94.39	11.35	220.46	27.79
NAD	2.68	0.41	19.23	1.40	32.15	2.58
RAD	0.0724	0.0110	0.0744	0.0058	0.0726	0.0063
EP	0.0744	0.0114	0.0760	0.0056	0.0739	0.0059
	Benchmark		Benchmark		Benchmark	
SD	327.54	88.62	94.01	11.11	220.03	27.92
NAD	2.66	0.41	19.20	1.40	32.11	2.58
RAD	0.0720	0.0109	0.0744	0.0057	0.0727	0.0063
EP	0.0740	0.0113	0.0759	0.0055	0.0738	0.0059

Table 4.9: Results for 100 runs of the ATS with the L^p DEF on the noisy data sets.

Weighted L^p DEF

The ATS for the weighted L^p DEF was performed on the noisy data. Figure 4.3 shows a pictorial representation of an ATS for multiple parameters in a noisy environment. Table 4.4 shows an example of the ATS, as well as the "Known", "Estimated" and "Benchmark" values. In the case of the weighted L^p DEF, there were two parameters that had to be determined. This is still a useful benchmark for our results since the "Benchmark" values were still as good or better than the "Estimated" values on the training data, as the "Estimated" values were used as the starting point for the hill-climbing search. At this point, it should again be emphasized that the "Known" value produce larger errors than both the "Estimated" and "Benchmark" values.

In the example run shown in Table 4.4, the parameter p converged faster than k.

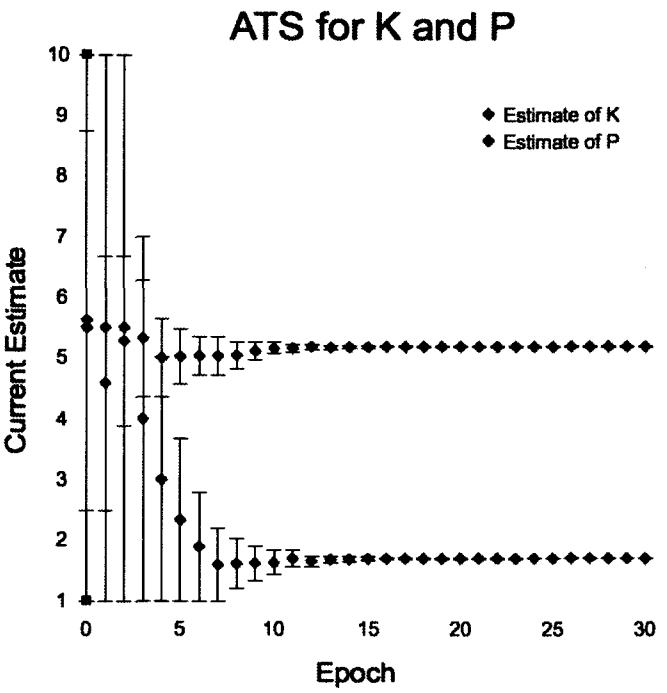

Figure 4.3: Blue and red lines represent the current search interval for the respective parameters and the diamonds represent the current estimate of the parameters.

The parameter p converged after 87 epochs while it took k 104 epochs to converge to the desired accuracy. After only 16 epochs both search intervals had a magnitude of less than 0.01. The errors for this example run are shown in Table 4.11, from which we can see that both the "Estimated" values and "Benchmark" values produced similar errors.

The average testing errors for 100 runs of the ATS using the weighted L^p DEF is shown in Table 4.12. The errors for this norm are smaller than the errors for the other two norms. This is an anticipated result because the weighted L^p DEF had two parameters, and this concurs with the results found in the literature [3, 11, 12,

Epoch	$k's$Current Search Interval	$p's$Current Search Interval
1	[7.0, 10.0]	[1.0, 1.6666667]
2	[8.0, 10.0]	[1.0, 1.4444445]
3	[8.333333, 9.722222]	[1.0, 1.4444445]
4	[8.564815, 9.529321]	[1.0, 1.4444445]
5	[8.564815, 9.529321]	[1.0, 1.2962964]
6	[8.725566, 9.395362]	[1.0, 1.2962964]
7	[8.725566, 9.395362]	[1.0, 1.1975309]
8	[8.837198, 9.302335]	[1.0329218, 1.170096]
9	[8.992244, 9.302335]	[1.0329218, 1.170096]
10	[9.043925, 9.259267]	[1.0557842, 1.1510441]
11	[9.079816, 9.229359]	[1.0716609, 1.1378136]
12	[9.10474, 9.208589]	[1.0826863, 1.1286256]
13	[9.139357, 9.208589]	[1.0826863, 1.1133125]
14	[9.139357, 9.162434]	[1.092895, 1.1133125]
15	[9.154741, 9.162434]	[1.092895, 1.1065067]
16	[9.154741, 9.157306]	[1.0951636, 1.1046162]
17	[9.154741, 9.155596]	[1.0983144, 1.1046162]
18	[9.154741, 9.155596]	[1.0993648, 1.1037409]
19	[9.154741, 9.155596]	[1.1000941, 1.1031331]
23	[9.154741, 9.155596]	[1.1006006, 1.102711]
33	[9.154741, 9.155596]	[1.101304, 1.102711]
42	[9.154741, 9.155596]	[1.1015385, 1.1025156]
50	[9.154741, 9.155026]	[1.1015385, 1.1018642]
58	[9.154741, 9.155026]	[1.1015928, 1.101819]
61	[9.154741, 9.154931]	[1.1015928, 1.1017436]
66	[9.154741, 9.154931]	[1.1016179, 1.1017226]
67	[9.154741, 9.154868]	[1.1016179, 1.1017226]
69	[9.154741, 9.154826]	[1.1016179, 1.1017226]
70	[9.154741, 9.154826]	[1.1016179, 1.1016877]
76	[9.154741, 9.154798]	[1.1016179, 1.1016644]
79	[9.154741, 9.1547785]	[1.1016179, 1.1016644]
81	[9.154741, 9.1547785]	[1.1016257, 1.101658]
84	[9.154741, 9.154754]	[1.1016257, 1.101658]
85	[9.154741, 9.154754]	[1.1016257, 1.1016473]
87	[9.154741, 9.154754]	[1.1016328, 1.1016473]
97	[9.154741, 9.15475]	[1.1016328, 1.1016473]
98	[9.154741, 9.154744]	[1.1016328, 1.1016473]
100	[9.154741, 9.154743]	[1.1016328, 1.1016473]
104	[9.154741, 9.154742]	[1.1016328, 1.1016473]
Known Values	$k^* = 9.208$	$p^* = 1.117$
Estimated Values	$k = 9.154741$	$p = 1.10164$
Benchmark Values	$k = 9.187748$	$p = 1.11164$

Table 4.10: Example run of the ATS with the weighted L^p DEF on the noisy data.

Error Type	For Estimated Value: $k = 9.15, p = 1.10$	For Benchmark Value: $k =, 9.19p = 1.11$
	Training	Training
SD	169.48	158.42
NAD	1.44	1.40
RAD	0.0074	0.0071
EP	0.0076	0.0074
	Testing	Testing
SD	17.66	16.76
NAD	0.18	0.18
RAD	0.0045	0.0047
EP	0.0049	0.0049

Table 4.11: Results for the typical run in Table 4.10.

15]. The errors for the "Estimated" values are are very close to the errors for the "Benchmark" values, where the RAD and EP errors are within 0.2% of each other.

Data Set Size	N=29		N=75		N=100	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
SD	99.38	85.67	28.90	24.50	57.95	44.12
NAD	0.75	0.62	5.60	4.35	8.13	6.11
RAD	0.0208	0.0171	0.0218	0.0167	0.0186	0.0138
EP	0.0208	0.0172	0.0221	0.0172	0.0187	0.0141
	Benchmark		Benchmark		Benchmark	
SD	85.36	71.04	24.85	18.30	51.73	37.37
NAD	0.70	0.56	5.20	3.76	7.67	5.68
RAD	0.0194	0.0155	0.0203	0.0147	0.0176	0.0129
EP	0.0193	0.0156	0.0205	0.0149	0.0176	0.0130

Table 4.12: Results for 100 runs of the ATS with the weighted L^p DEF on the noisy data sets.

4.2.3 Results for the real-world data sets

Experimental Setup

The final test for the ATS was done on real-world data sets. This data consisted of three sets, which in turn involved 29, 97, and 561 cities each. The data sets with 29 and 561 cities were obtained from the MP-TESTDATA (the TSPLIB Symmetric Traveling Salesman Problem Instances) [49]. The data set with 29 points is titled "bays29.tsp". This data was collected from cities in Bavaria, and records the inter-street distances and the locations of the cities. The data set with 561 points is titled "pa561.tsp" and also reports the inter-street distance and the co-ordinates of the cities. The data set with 97 cities was obtained using Turkish cities. The co-ordinates were obtained from [1, 16], and the distances were calculated using Google maps [50].

Observe that for data of this type, there are no "Known" values of k and p. This is because the data was not *created* and therefore did not depend on any "Known" values. The "Benchmark" values were again used for comparison, and the same hill-climbing search was used to determine the "Benchmark" values.

Weighted Euclidian DEF

Table 4.13 shows the result for the three data sets when the ATS used the weighted Euclidian DEF. The standard deviation for the "Benchmark" values was always zero. This is because we were not changing the data set for each run, as in the previous two types of data. The reason for doing multiple runs on the same data set was to see how the ATS behaved.

The ATS out-performed the hill-climbing for the data sets of size 29, and 97. For the "bays29" data set, the ATS had less than a 1% advantage for both the RAD

Data Set Size	N=29		N=97		N=561	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
Value	0.2230	1.6259	1.3808	0.0100	0.1635	0.0010
SD	23.66	0.18	20053.85	129.84	17597.54	788.93
NAD	1.82	0.01	83.62	0.38	2237.76	51.05
RAD	0.0406	0.0001	0.1268	0.0015	0.1448	0.0043
EP	0.0507	0.0002	0.2060	0.0009	0.1576	0.0036
	Benchmark		Benchmark		Benchmark	
Value	0.2260	0.00	0.9960	0.00	0.1660	0.00
SD	28.32	0.00	35489.89	0.00	15880.77	0.00
NAD	2.02	0.00	141.93	0.00	2123.57	0.00
RAD	0.0438	0.00	0.3082	0.00	0.1350	0.00
EP	0.0562	0.00	0.3496	0.00	0.1496	0.00

Table 4.13: Results for 100 runs of the ATS with the weighted Euclidian DEF on the real-worlds data sets.

and EP. When tested on the data set from Turkey, the ATS out-performed the hill-climbing scheme by over 17% and 5% for the RAD and EP respectively. For the larger data set, "pa561", the hill-climbing did out-perform the ATS, but by less than 1% for both the RAD and EP errors. The ATS was able to out-perform the hill climbing because the errors reported here are the testing error, and the hill-climbing was trained using the errors from the training set.

The results of this test are encouraging because the ATS was able to compete with the hill-climbing scheme, that had only a single optimum. To better comprehend the performance of the ATS, one could also possibly compare the results of the hill-climbing scheme with the maximum and minimum errors that the ATS yielded.

L^p DEF

Table 4.14 shows the results for 100 runs of the ATS search using the L^p norm on the real-world data. Overall the errors were extremely large, over 300% error for the RAD

Data Set Size	N=29		N=97		N=561	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
Value	25.0	0.00	0.8044	0.00	25.0	0.00
SD	71326.52	0.00	27650.57	0.00	8610472.43	0.00
NAD	111.04	0.00	103.20	0.00	54193.61	0.00
RAD	3.0184	0.00	0.1731	0.00	3.7626	0.00
EP	3.0844	0.00	0.2542	0.00	3.8175	0.00
	Benchmark		Benchmark		Benchmark	
Value	25.0	0.00	1.4565	0.00	25.0	0.00
SD	71326.52	0.00	29687.46	0.00	8610472.43	0.00
NAD	111.04	0.00	127.73	0.00	54193.61	0.00
RAD	3.0184	0.00	0.2661	0.00	3.7626	0.00
EP	3.0844	0.00	0.3146	0.00	3.8175	0.00

Table 4.14: Results for 100 runs of the ATS with the L^p DEF on the real-world data sets.

and EP errors for the data set of size 29 and 561. This can be attributed to two main reasons; first the L^p norm had very limited predicting power, and second, both the "Estimated" values and "Benchmark" values were actually at the maximum value of the ATS search interval. Regardless of how this maximum value was changed, both the ATS and the hill-climbing converged to the largest value. It should be noted that the change in the DEF decreased for larger values of p.

The L^p norm did a much better job of estimating the distances for the Turkey data. This may be due to the type of network and region under study. The errors are still high but both the ATS and the hill-climbing search converge to values within the search interval. The ATS out-performed the simple hill-climbing search by about 5% for both the RAD and the EP errors on the Turkish data.

Data Set Size	N=29		N=97		N=561	
Error Type	Average Error	Standard Deviation	Average Error	Standard Deviation	Average Error	Standard Deviation
	Estimated		Estimated		Estimated	
K Value	0.2220	$9.4600x10^{-4}$	1.3517	0.0164	0.1410	0.0022
P Value	1.9935	0.0353	1.8022	0.0932	1.1517	0.0437
SD	22.93	0.28	20118.81	226.04	17756.97	364.02
NAD	1.79	0.01	83.28	0.22	2227.20	19.61
RAD	0.0402	0.0003	0.1253	0.0004	0.1423	0.0016
EP	0.0496	0.0004	0.2051	0.0005	0.1569	0.0014
	Benchmark		Benchmark		Benchmark	
K Value	0.2203	$8.1873x10^{-8}$	1.2326	0.0053	0.1550	$4.9035x10^{-8}$
P Value	1.9200	$9.0190x10^{-8}$	1.5071	0.0221	1.7400	$6.0445x10^{-8}$
SD	23.71	0.00	19922.21	62.01	20522.75	0.05
NAD	1.83	0.00	86.27	0.16	2418.51	0.00
RAD	0.0412	0.0000	0.1381	0.0005	0.1598	0.0000
EP	0.0508	0.0000	0.2125	0.0004	0.1704	0.0000

Table 4.15: Results for 100 runs of the ATS with the weighted L^p DEF on the real-world data sets.

Weighted L^p DEF

When the ATS is used in conjunction with the weighted L^p DEF, the ATS out-performed the hill-climbing search, as shown in results in Table 4.15. While the ATS and the hill-climbing search perform very similarly, the ATS had a slight improvement over the hill-climbing search.

Both the data set with 29 points and the data set with 97 points have a p value that is close to 2.0. As a result, the weighted L^p DEF had a similar performance to the weighted Euclidean DEF. For the data set with 561 points, the ATS produced an average p value of about 1.2, whereas the hill-climbing search's p value is 1.74. This change in p value resulted in a larger difference in the accuracy of the estimation of the distances between the ATS using the weighted L^p DEF and the weighted Euclidean DEF. Finally, the ATS using the weighted L^p DEF, out-performed the previous two

DEFs.

4.3 Discussion

The ATS *always* converged close to the "actual" values for all three DEF when interacting with noiseless data sets. The errors were either exactly zero or smaller then $1x10^{-9}$ %. In addition to these small errors, the "actual" values were always contained in the final search interval. This indicated that the ATS was well adapted to finding multiple parameters in the ideal DE domain, and serves as an important baseline for more realistic data sets. Another observation is that the ATS converged very quickly, at every time step a search interval was reduced. Overall, the ATS was able to *always* accurately find the optimal parameters for noiseless data sets.

For the noisy data sets the benchmark values and estimates values were very close, and resulted in similar errors within 0.1%. The ATS did not reduce its search interval at each iteration; however, it was able to reduce the search interval to the desired accuracy with additional epochs. If the number of learning loops per epochs (N_∞) were increased, it would have been more likely to reduce the search interval at each epoch. The ATS was able to accurately determine the parameters of all three DEF in a noisy environment.

In the real world setting, the ATS was competitive with the hill-climbing search. While the hill-climbing search always found the same values, the ATS had small variance of the values. Both the ATS and the hill-climb search produced similar but large errors for the data sets of size 29 and 561 using the L^p DEF. These large errors are due to the predicting abilities of the L^p DEF. The similarity between the ATS and the hill-climbing search shows that the ATS is still a competitive search method.

Over all the ATS found values that were competitive with the standard hill clime method.

4.4 Summary of 2D Results

In this chapter, we applied the ATS search to three types of data: noiseless, noisy, and real-world. The noiseless data was used as a base line to establish that the ATS can function in an optimal DE domain. By applying the ATS to the noiseless data, we were able to determine the parameters to within a very small interval, i.e., less the 10^{-6}. Indeed, the actual values were *always* contained in this interval.

After establishing that the ATS performed as expected on noiseless data, it was then applied to noisy data. Because this type of data had noise added to it, the actual values were not necessarily ideal for use when estimating the distances. To create a better comparison, we used a hill-climbing search to find the "Benchmark" values for estimating the distances. The estimated values that the ATS found were similar to those obtained using the hill-climbing search. Besides, the ATS's accuracy was competitive with the latter. The reader must observe that the hill-climbing search had to calculate the accuracy using a GoF function at each time-step, and the ATS did not use a GoF, except when comparing its accuracy with that of the other methods.

The final 2-dimensional test for the ATS was on real-world data sets. These data sets used real road distances and co-ordinates, creating a practical setting to test the estimating power of the ATS. Again, we used a hill-climbing search to compare the accuracy of the ATS for different DEFs. The ATS was competitive with the hill-climbing scheme, and even exceeded it in the testing error. Since both the hill-climbing and the ATS were trained using a training set, it was possible for the ATS

to out-perform the hill-climbing method when using the testing portion of the data. This was, indeed, the case for the real-world data. The ATS yielded better testing results than the optimally trained parameter for almost all of the data sets and the DEF combinations.

When testing the ATS using the L^p DEF, the expected error for the estimated distances for two of the data sets was over 300%. This could be due to the units of the co-ordinates and distance; however, the hill-climbing search did not produce any better values. So, we are able to conclude that this large error is due to the *predicting* abilities of the DEF for these particular data sets. Indeed, the L^P DEF is a very poor predicting function as discussed in Chapter 2. We implement it hear as a stepping stone for the weighted L^P DEF. The fact that the ATS has the same performance as the hill-climbing is the best that can be hoped for in such a situation.

When using the noiseless data, the ATS was able to determine the parameters to a very small accuracy. For the real-world sets, the accuracy depended on the data set. If the accuracy was set to be too small, the ATS would stop converging before it reached the desired accuracy. The limit of the accuracy depends on the DEF, the size of the data set, and the how the data "fits" the DEF.

Overall, this chapter has outlined the ATS for the 2-dimensional DE, and applied it to different types of regions. We have also discussed its properties and compared it against a well-established hill-climbing search in order to gauge its performance.

Chapter 5

3D Distance Estimation Using the ATS

5.1 Introduction

In this chapter, we consider the problem of DE in three dimensions, referred to as 3DDE. To permit us to do preliminary testing of 3DDE, we have resorted to utilizing artificially-constructed data so as to test it with the ATS. Data of this type will give us a better understanding of how the DE method behaves. Using the artificially-constructed data, we have compared the results from the 3DDE using the two dimensional method presented in Chapter 4. From these comparisons, we are able to show which method is superior in these situations. These examples are "simple" because they are not real-world examples, and so require some assumptions regarding how the data sets were constructed. Thus, for the sake of completeness, the process for creating the data sets is also outlined in Section 5.3.

DE has been well studied in the two-dimensional domain and has been used in

various applications, as discussed in Chapter 2. The use of a third dimension could be especially beneficial if the region in question has a predominant hilly or mountainous terrain. For example, estimating distances in regions like Perugia, Italy, or Zermatt, Switzerland may require the altitude of the points to more accurately estimate the inter-point distances.

3DDE is almost identical to its two dimensional counterpart. It essentially uses DEFs to estimate the true distances between points. The only difference between the two are that the points require a third dimension and, as a result, the DEFs must be generalized for such a higher-dimensional space. However, specifying a third dimension for the points is not overly complicated. Any GPS can provide the elevation of a point on the earth's surface if one knows the longitude and latitude of that point. The DEFs are also easily modified for three dimensions, since the norms that they are based on, are generalized for n dimensions. The generalized DEFs will be briefly discussed in Section 5.2.

5.2 3D DEFs

DEFs in three dimensions are similar to those defined for two. From Chapter 2, we know that a DEF is a function $\pi(P_1, P_2|\Theta) : R^n X R^n \longrightarrow R$, where $P_1 = (x_1, x_2, ..., x_n)$ and $P_2 = (y_1, y_2, ..., y_n)$. The three dimensional case occurs when $n = 3$.

Since the primary focus of DEFs has been in terms of the L^p DEFs, this is the family of functions that will be used throughout this chapter. Just as in the two dimensional case, the three dimensional L^p DEFs are based on the L^p norm given in Equation (2.7). The three dimensional weighted L^p DEF can be seen in Equation (5.1).

$$L_p(X) = \left(k \sum_{i=1}^{3} (|x_i|^p \right)^{1/p}. \tag{5.1}$$

Indeed, this is the DEF that will be used for comparing the two dimensional and three dimensional DE methods that invokes the ATS.

5.3 Creating 3D Data

In this section, we outline the method used for creating the data utilized for the testing of the algorithms. This involves a straightforward strategy that can be used to first create points on a three dimensional surface, and to, thereafter, find the inter-point distances which one would have to travel.

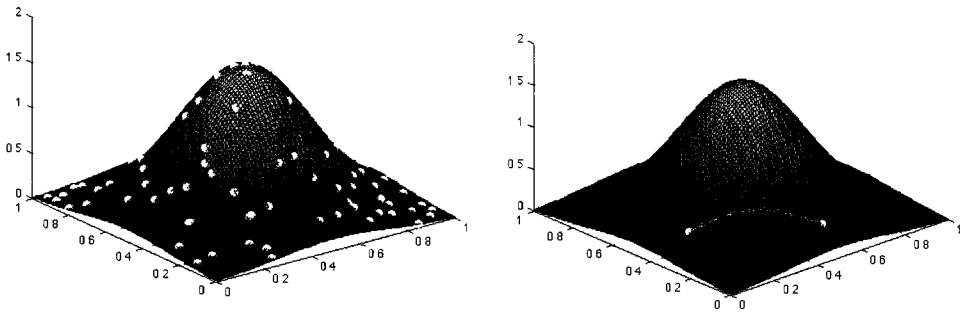

Figure 5.1: The process of creating data. In Figure 5.1a (left) we compute the points. In Figure 5.1b (right) we determine the inter-point distance.

The first step is to get the x and y co-ordinates for each point. Obviously, for testing and training purposes, these points are randomly generated. Once the x and y co-ordinates have been found, the z co-ordinate can be calculated by projecting the

points vertically onto the surface. As a surface can be defined as a function of the x and y co-ordinates, this projection to find the z co-ordinate is straightforward. An example of the projection of the points onto a surface can be seen in Figure 5.1a.

Once this is achieved, we will have our set of points, and the task of determining the inter-point distances has to be tackled. This can be resolved in several ways. The path that we chose to use is as "the crow flies" in the two dimensional space and to then project *that* path "upwards" onto the respective surface.

To achieve this, we first determine intermediate the points, C_i, along the line connecting the two points, A and B. To do this, we utilize Equation (5.2), where $0 \leq \lambda \leq 1$. As λ is increased from 0 to 1, the intermediate point will travel from point B towards point A. The distance between intermediate points, D, is used to determine the change in λ, $\Delta\lambda = \frac{D}{\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}}$. The processes of finding the intermediate points is shown in Algorithm 7.

$$C_n = (\lambda)A + (1 - \lambda)B. \tag{5.2}$$

By way of example, when the intermediate points are projected on the surface, they may form the line shown in Figure 5.1b. Now that we have the intermediate points, the distance between the original two points can be determined by incorporating the individual distances between the intermediate points. This is done by summing the three dimensional Euclidian distances between the sequential intermediate points along the the path.

This method is an approximation of the path integral over the surface. We chose to use this discrete method because it simplifies computation and is easy to generalize for different surfaces and paths. For example. this method can easily be applied to both piecewise surfaces and paths. Indeed, our assumption of a straight line between

Algorithm 7 Finding Intermediate Points

Input: A, B, and D

Output: The set of intermediate points

Method:

1: $\delta\lambda = \frac{D}{\sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}}$;
2: $\lambda = 0$;
3: $i = 0$;
4: **repeat**
5: $x = A_x(\lambda) + B_x(1 - \lambda)$;
6: $y = A_y(\lambda) + B_y(1 - \lambda)$;
7: $z = getZbyProjection(x, y)$;
8: $Intermediate(i) = (x, y, z)$;
9: $i++$;
10: $\lambda += \delta\lambda$;
11: **until** $\lambda >= 1$

points may by changed to more complex paths in future research.

To obtain the training and testing sets, this process should be repeated in such a way that each point has a path to all other points in the set. An example of these paths can be seen in Figure 5.2a. Once all the calculations have been completed. the final network will be fully represented by the original set of points and their corresponding inter-point distances. Thereafter, we observe that there is no more need for the intermediate points. Figure 5.2b shows an example of a three dimensional network.

5.4 Experimental Setup

To compare the 3DDE with the typical two dimensional DE (2DDE), we used the ATS in combination with the weighted L^p DEF. Equation (2.11) was used as the DEF for the 2DDE and Equation (5.1) was used for the 3DDE. The weighted L^p DEF was chosen for four main reasons. This DEF is commonly used for DE and has been well studied. It is also very versatile. It can out-perform many other DEFs in networks

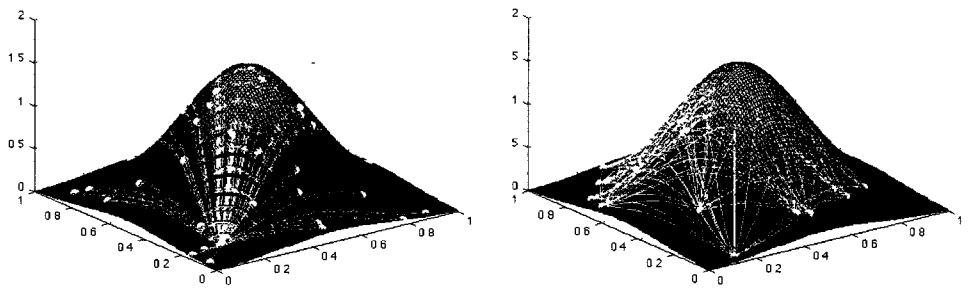

Figure 5.2: The process of creating data. In Figure 5.2a (left) we compute all the distance for a single point, and Figure 5.2b (right) shows the final network of all the corresponding paths

of differing structures. The final reason, is its simplicity as it generally performs well for a DEF with only two variables.

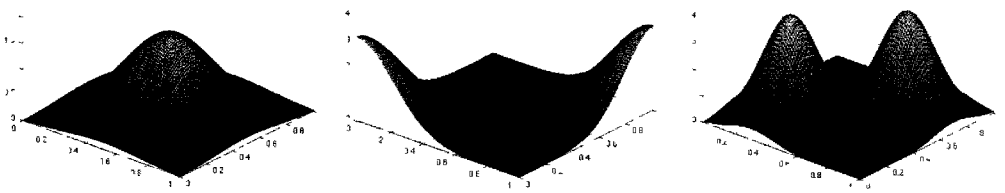

Figure 5.3: Example of 3D Surfaces

The testing was done on two types of data sets. The first type was constructed using the method outlined in Section 5.3, and the second was the noisy data set. The noisy data set was constructed in the same manner as the first noiseless set, except that a noise component was added to the inter-point distances. This additional term

was proportional to the magnitude of the noiseless distance, where the proportion was based on a Gaussian distribution as shown in Equation (5.3).

$$Noise = \frac{TrueDistance}{10} * X, \tag{5.3}$$

where, X is a random Gaussian variable, N(0,1).

All of the data sets consisted of 100 points (or cities). This value was chosen based on the common data set sizes in the literature. The distance between the intermediate points, D, was another parameter that had to be defined. We used the value $D = 0.05$, which was based on the largest rate of change of the terrain and the desired accuracy for the inter-point distances. This inter-point distance was small enough to account for the features on all the surfaces considered in this chapter. Further, both the 2D and 3DDE methods used the same data sets. The only difference for the 2DDE method was that it did not incorporate the third dimension.

For both the noiseless and the noisy data, there were three types of surfaces used to construct the data sets. The first type was a single hill, an example of which is shown in Figure 5.3(left). Five different single Gaussian hill surfaces, $N(0.5, \sigma)$, were constructed to determine the accuracy of the DE of both "easy" and "hard" terrains. The parameters for these surfaces are shown in Table 5.1 . The second type of surface used was a valley. This consisted of two Gaussian hills, $N(0.\sigma_1) + N(1, \sigma_2)$ located at opposite corners of the terrain, which had a 1x1 unit base. An example of this is shown in Figure 5.3(center). Again, the steepness of the valley was varied according to the values in Table 5.1 . The last surface consists of two Gaussian hills side by side, $N(\mu_1, \sigma_1) + N(\mu_2, \sigma_2)$. This is, clearly, the most complete surface and is shown in Figure 5.3(right). The differences between the two hill surfaces are outlined in Table 5.1. For this scenario, the steepness was also changed.

Surface:	Single Hill		Valley		Two Hills	
Number	Mean (μ)	Variance (σ^2)	Means (μ_1, μ_2)	Variances (σ_1^2, σ_2^2)	Means (μ_1, μ_2)	Variances (σ_1^2, σ_2^2)
1	(0.5)	(0.3)	(0.0, 1.0)	(0.02, 0.02)	(0.2, 0.8)	(0.09, 0.09)
2	(0.5)	(0.2)	(0.0, 1.0)	(0.05, 0.05)	(0.2, 0.8)	(0.09, 0.07)
3	(0.5)	(0.1)	(0.0, 1.0)	(0.09, 0.09)	(0.2, 0.8)	(0.07, 0.07)
4	(0.5)	(0.09)	(0.0, 1.0)	(0.12, 0.12)	(0.2, 0.8)	(0.07, 0.05)
5	(0.5)	(0.08)	(0.0, 1.0)	(0.15, 0.15)	(0.2, 0.8)	(0.05, 0.05)

Table 5.1: The 3D testing surfaces.

5.5 Noiseless Results

Using the above described data sets, the 3DDE and the 2DDE were compared over 20 runs to ensure a fair comparison.

5.5.1 Single Hill Surfaces

The results for the single hill surfaces are shown in Table 5.2. Both the 2DDE results and the 3DDE results are shown for the five different hills described in Table 5.1 of the previous section.

Name σ^2	Hill-1 0.3	Hill-2 0.2	Hill-3 0.1	Hill-4 0.09	Hill-5 0.08
	3D	3D	3D	3D	3D
SD	0.0103	0.2052	5.95	10.7081	23.20
NAD	1.8370	8.4747	41.09	55.7849	66.78
RAD	0.0053	0.0229	0.1175	0.1627	0.2071
EP	0.0042	0.0195	0.0945	0.1282	0.1535
	2D	2D	2D	2D	2D
SD	0.0113	0.1956	26.11	31.9760	81.13
NAD	3.2779	12.5711	131.92	151.9378	214.73
RAD	0.0059	0.0213	0.2457	0.2663	0.3692
EP	0.0075	0.0289	0.3033	0.3493	0.4936

Table 5.2: Results for 20 runs of 2D ATS and 3D ATS on single hill surfaces

The first hill with $\sigma^2 = 0.3$ had an error of less that 1% for both 2DDE and

3DDE. When $\sigma^2 = 0.2$, the error increased. For both the 2DDE and 3DDE, the errors were about 2%, well within reason. For $\sigma^2 = 0.1$, the error is much larger. The expected error for an estimated distance and the overall error were less then 12% for the 3DDE, while the 2DDE errors were both over 24%. The error continues to increase for a surface with $\sigma^2 = 0.09$. For the 3DDE, the expected error for the estimated distance and the overall error were 16% and 13% respectively, while for the 2DDE these were 27% and 35% respectively. The final hill that was considered, had a variance of 0.08. This surface produced errors larger than 15% for both the 3DDE and the 2DDE.

5.5.2 Valley Surfaces

The valley surfaces which are constructed using the parameters listed in Table 5.1, and produced the results in Table 5.3.

Name σ^2	Valley-1 0.02	Valley-2 0.05	Valley-3 0.09	Valley-4 0.12	Valley-5 0.15
	3D	3D	3D	3D	3D
SD	16.96	4.01	4.99	1.0654	0.72
NAD	28.01	23.17	23.98	16.94	18.27
RAD	0.1093	0.0762	0.0768	0.0438	0.0436
EP	0.0644	0.0533	0.0551	0.0389	0.0420
	2D	2D	2D	2D	2D
SD	14.22	28.38	59.52	21.98	7.73
NAD	42.77	145.32	189.08	128.97	70.97
RAD	0.1069	0.3101	0.3980	0.2833	0.1537
EP	0.0983	0.3341	0.4347	0.2965	0.1631

Table 5.3: Results for 20 runs of 2D ATS and 3D ATS on valley surfaces

The 3DDE performed well on the valley surfaces. All of the errors were less then 11%. For the 3DDE, the highest error was 10.7% for the valley with $\sigma^2 = 0.02$. The next two valleys with σ^2 values of 0.05 and 0.09 had similar results for the 3DDE with

errors of about 7%. The last two valleys with $\sigma^2 = 0.12$ and $\sigma^2 = 0.15$ had about 4.3% errors. The main trend that we observed is that the less steep the valley is, the better the 3DDE performs. The 2DDE does not follow the same trend. The 2DDE performed best on the valley with the steepest sides. Indeed, the RAD and the SD error of the 2DDE method was less than those of the 3DDE. All other 2DDE errors were larger than the corresponding 3DDE errors. The second smallest 2DDE error was for the valley with the flattest sidewalls. The errors were 15.4% and 16% for the RAD and the EP respectively. All other errors for the 2DDE were greater then 28%.

5.5.3 Two Hill Surfaces

The results for the five two hill surfaces as described in Table 5.1 are shown in Table 5.4. Each of these surfaces consisted of two hills where the steepness of the hills were varied.

Name (σ_1^2, σ_2^2)	Two Hills-1 (0.09, 0.09)	Two Hills-2 (0.09, 0.07)	Two Hills-3 (0.07, 0.07)	Two Hills-4 (0.07, 0.05)	Two Hills-5 (0.05, 0.05)
	3D	3D	3D	3D	3D
SD	10.6310	13.0030	12.2464	26.5862	45.7997
NAD	56.6280	58.9983	52.2928	79.4759	92.8988
RAD	0.1527	0.1483	0.1380	0.2080	0.2472
EP	0.1302	0.1356	0.1202	0.1827	0.2136
	2D	2D	2D	2D	2D
SD	54.1130	84.3325	223.2262	325.5372	500.5970
NAD	177.8009	205.7288	347.3660	397.9892	468.6166
RAD	0.3361	0.3937	0.7274	0.7667	0.8396
EP	0.4087	0.4729	0.7985	0.9149	1.0773

Table 5.4: Results for 20 runs of 2D ATS and 3D ATS on surfaces with two hills

The 2DDE errors were at least twice as large as the 3DDE errors for the respective surfaces. The smallest 2DDE error of 33.6% and of 40.9% was for surface with the two flattest hills, which had $\sigma^2 = 0.09$. When the variance of one of the hills was

decreased to 0.07, the error increased by about 6%. When both hills associated with the surfaces had $\sigma^2 \leq 0.07$, the error increased drastically, and all of these errors were greater than 72%. The 3DDE well out-performed the 2DDE. It has a maximum error of 24.7% and a minimum error of 12.0%. The smallest error was for the surface where both hills had a σ^2 value of 0.07. The three surfaces had $\sigma^2 \geq 0.07$, displayed similar errors, within 1.5%. For the other two surfaces where $\sigma_1^2 < 0.07$ and $\sigma_2^2 < 0.07$, the errors were greater then 18%.

5.6 Noisy Results

The comparison of the 3DDE on noisy data sets were conducted in the same manner as in Section 5.5. The 3DDE was compared to the 2DDE over 20 runs. Further, the level of noise used was different for each run.

5.6.1 Single Hill Surfaces

Table 5.5 shows the errors for the surfaces containing a single hill. These surfaces are in order from the flattest to the steepest.

The first single hill surface had a value of $\sigma^2 = 0.3$. This is a flat hill, and as a result the 2DDE marginally out-performs the 3DDE. The 2DDE and the 3DDE produced an 8% and slightly over 9% error respectively. The 3DDE method had a slight improvement to about 8.8% when $\sigma^2 = 0.2$; while the 2DDE error increases to over 9% error. When σ^2 was increased to values greater than or equal to 0.1, the 2DDE produced an error greater than 26%. For the 3DDE, all the errors were less than 21%. One observes that the steeper the hill, the larger the error, as is understandable.

Name	Hill-1	Hill-2	Hill-3	Hill-4	Hill-5
σ^2	0.3	0.2	0.1	0.09	0.08
	3D	3D	3D	3D	3D
SD	3.05	3.13	8.01	12.26	23.16
NAD	40.09	38.46	56.36	63.70	77.81
RAD	0.0950	0.0871	0.1369	0.1726	0.2135
EP	0.0922	0.0884	0.1296	0.1464	0.1789
	2D	2D	2D	2D	2D
SD	2.24	3.78	32.89	30.91	79.61
NAD	36.05	44.28	145.42	147.94	209.79
RAD	0.0787	0.0933	0.2722	0.2608	0.3627
EP	0.0829	0.1018	0.3343	0.3401	0.4823

Table 5.5: Results for 20 runs of 2D ATS and 3D ATS on single hill surfaces

5.6.2 Valley Surfaces

The results of the 2DDE and 3DDE on the valley surfaces with noisy distances are summarized in Table 5.6.

Name	Valley-1	Valley-2	Valley-3	Valley-4	Valley-5
σ^2	0.02	0.05	0.09	0.12	0.15
	3D	3D	3D	3D	3D
SD	21.63	5.93	7.39	4.29	3.56
NAD	49.40	46.24	48.27	45.99	42.40
RAD	0.1469	0.1128	0.1196	0.1012	0.0915
EP	0.1136	0.1063	0.1110	0.1057	0.0975
	2D	2D	2D	2D	2D
SD	17.23	31.42	53.60	26.22	12.80
NAD	57.95	146.90	174.09	136.31	86.96
RAD	0.1318	0.3036	0.3593	0.2863	0.1784
EP	0.1332	0.3377	0.4002	0.3133	0.1999

Table 5.6: Results for 20 runs of 2D ATS and 3D ATS on single hill surfaces

For the valley surfaces, the 2DDE and the 3DDE have the closest performance on the valley with the steepest sidewalls. The 2DDE had errors of 13%, and the 3DDE had an expected error for an estimated distance and the overall error of 11% and 15% respectively. As the valleys flatten out, the RAD and EP errors for the

2DDE increased to 35% and 40% respectively. Thereafter, the 2DDE improved as the valleys continued to flatten. For the flattest valley, the 2DDE had errors of 18% and 20%. The 3DDE decreased to a maximum of 5%. When $\sigma^2 > 0.02$, the errors were all between 12% and 9%.

5.6.3 Two Hill Surfaces

Table 5.7 shows the results for the noisy distances on the surface with two hills. The surfaces are arranged, based on the mean height of the two hills from the shortest to the highest.

Name (σ_1^2, σ_2^2)	Two Hills-1 (0.09, 0.09)	Two Hills-2 (0.09, 0.07)	Two Hills-3 (0.07, 0.07)	Two Hills-4 (0.07, 0.05)	Two Hills-5 (0.05, 0.05)
	3D	3D	3D	3D	3D
SD	13.53	16.18	15.60	31.18	47.14
NAD	67.55	67.53	66.52	85.15	97.39
RAD	0.1728	0.1635	0.1637	0.2238	0.2546
EP	0.1553	0.1552	0.1529	0.1957	0.2239
	2D	2D	2D	2D	2D
SD	54.49	86.05	197.49	302.59	517.72
NAD	175.28	206.11	326.22	380.01	476.78
RAD	0.3236	0.3810	0.6673	0.7219	0.8441
EP	0.4029	0.4738	0.7499	0.8736	1.0960

Table 5.7: Results for 20 runs of 2D ATS and 3D ATS on surfaces with two hills

The 3DDE had RAD and EP errors ranging from 25% to 15%. The terrains with the smallest error contained hills where $\sigma_1^2 = 0.09$, $\sigma_2^2 = 0.07$ and $\sigma_1^2 = 0.07$, $\sigma_2^2 = 0.07$. The largest error of 25% for the 3DDE was caused by estimating distances for the surface with the two steepest hills, $\sigma_1^2 = 0.05$ and $\sigma_2^2 = 0.05$. This surface also forced the 2DDE to produce its largest error of 109%. In fact, the smallest error that the 2DDE produced for the surfaces with two hills was 32%., which was for the flattest hills.

5.7 Discussion

5.7.1 Noiseless

To discuses the noiseless results, Figures 5.4 -5.6 shows the errors of each surface type against the overall steepness of the terrain.

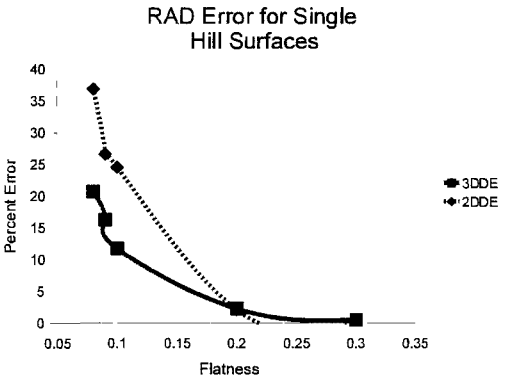

Figure 5.4: Plot of RAD errors versus the steepness of the terrain for the noiseless single hill data sets.

Single Hill Surfaces

The first hill with $\sigma^2 = 0.3$ was easily estimated by both 2DDE and 3DDE. The hill was not pronounced enough to cause the DE methods any trouble. Again, the second hill with $\sigma^2 = 0.2$ had the errors which were still small and we believe, are quite useful for practical purposes. The third hill with $\sigma^2 = 0.1$ was more of a challenge. The 3DDE error increased, although it was still respectable. The 2DDE error for the third hill was very large and for many applications it would be unreasonably large to be of any use. When $\sigma^2 = 0.09$, the 3DDE was larger, but in the absence of a superior

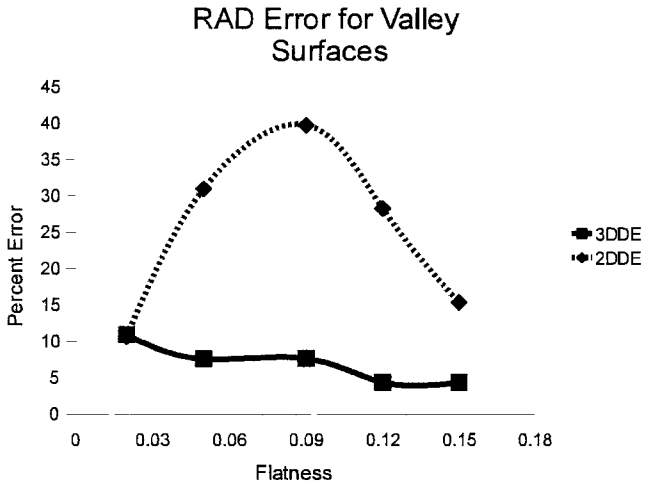

Figure 5.5: Plot of RAD errors versus the steepness of the terrain for the noiseless valley data sets.

method, we argue that it could still be useable for some applications. The 2DDE error was unusually large, and we believe quite useless. The last hill with $\sigma^2 = 0.08$ yielded large errors for both the 2DDE and the 3DDE. While the 3DDE errors are less than two thirds of the 2DDE errors, they are still too large to be used in practice.

The trend in Figure 5.4 shows that as σ^2 decreases, the errors become larger. In fact, we have shown that both the 3DDE and the 2DDE (that have been discussed here) have a limit to the terrains in which they can be usefully applied. Indeed, we see that the error for a simple Gaussian hill increases at a much faster rate when $\sigma^2 < 0.1$. Overall, the 3DDE significantly out-performs the 2DDE on the noiseless single hill terrains that have been examined.

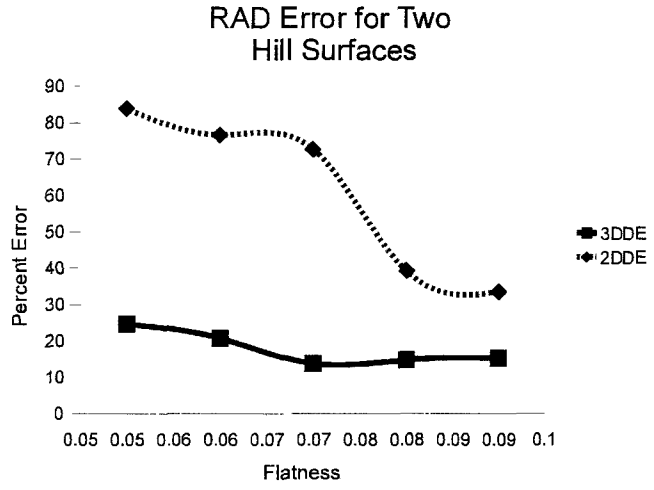

Figure 5.6: Plot of RAD errors versus the steepness of the terrain for the noiseless two hill data sets.

Valley Surfaces

For each of the valley surfaces, the 3DDE performed better then the 2DDE method. Theses two different methods have very different trends. The 3DDEs estimates improved as the sides of the valley became less steep. It performed well on all of these surfaces, and all but one error were less that 10%. The 2DDE performed best on the valley with the steepest sides and the flattest sides. The improved performance on the valley with the steepest hill is probably due to the larger flat center of the valley. Figure 5.5 shows the advantage that the 3DDE method has over the 2DDE method when examining valley surfaces.

Two Hill Surfaces

The surfaces with two hills were by far the most challenging for the two methods. For the 3DDE the errors ranged from 12% to 25%. While a 12% error may be acceptable, the 25% error is certainly high for a useful distance estimation strategy. To improve this, we suggest dividing the regions into multiple subregoins that may resemble a single hill or a valley surface. The 2DDE had a minimum error of 33% and a maximum over 100% error. The 2DDE is clearly not a desirable distance estimation method for this type of terrain.

5.7.2 Noisy

The discussion in this section will be focused on the RAD errors. These errors for both the 2DDE and the 3DDE are plotted in against the steepness of the surface in Figure 5.7 - 5.9. The flatter surfaces can be seen on the right of each plot while the steeper surfaces are on the left.

Single Hill Surfaces

Both the 3DDE and the 2DDE have the same trend for the single hill surfaces. They both perform better on the flatter surfaces. Overall, the 3DDE produces a better estimate than the 2DDE. However, for the flattest surface the 2DDE out-performs the 3DDE. The performance of both the noisy 2DDE and the 3DDE are similar to that of those noiseless counterparts for the steeper hills, $\sigma^2 \leq 0.1$. The noise causes the 2DDE and the 3DDE to have greater errors on the flatter surfaces, i.e., for $\sigma^2 \geq 0.2$.

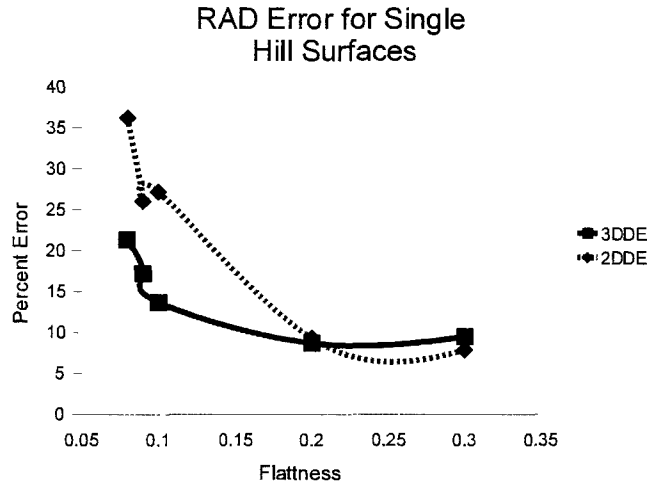

Figure 5.7: Plot of RAD errors versus the steepness of the terrain for the noisy single hill data sets.

Valley Surfaces

The 3DDE has little variance in the errors. They range from 15% to 9%, with the steepest valley having the largest errors, and the flattest valley having the lowest errors. All of the noisy 3DDE errors are larger than their noiseless counterparts. The 2DDE does not follow a linear trend. It performs best on the steepest valley, and the flattest valley. The 2DDE produces the worst result for the middle valley when $\sigma^2 = 0.09$. It is intuitive that the 2DDE would perform best on the flattest surface. The ability for the 2DDE to perform better on the steepest valley than the middle valley is again, possibly due to the flat center of the steepest valley. This is the same trend that was observed for the noiseless data sets.

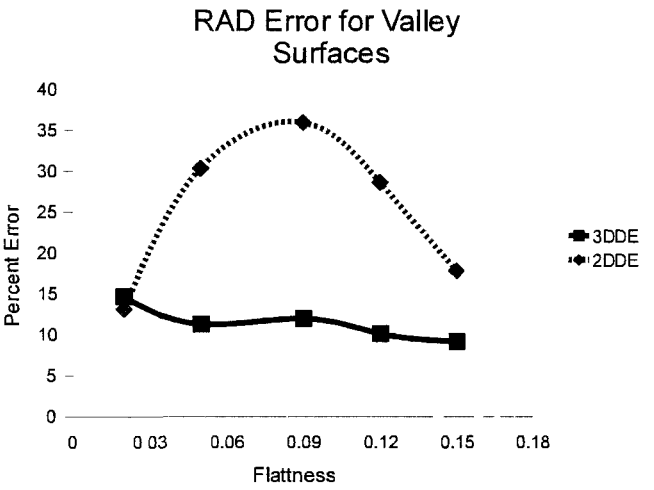

Figure 5.8: Plot of RAD errors versus the steepness of the terrain for the noisy valley data sets.

Two Hill Surfaces

Just as for the noiseless data, the surfaces with two hills were the hardest for both methods to produce accurate estimates. Both performed best on the flatter hills, and worst on the steepest. The 2DDE had a much larger rate of change for its error than the 3DDE. The 3DDE produced less then half the errors of the 2DDE for all but one surface. Both the trends and magnitude of error are similar to the results seen for the noiseless data sets involving surfaces with two hills.

5.7.3 Examining Error Distribution

From these preliminary tests, we see that the 3DDE is superior to the 2DDE when using the weighted L^p DEF on more "hilly" surfaces. While the error on the more challenging surface is higher than one would like, the 3DDE was still consistent and

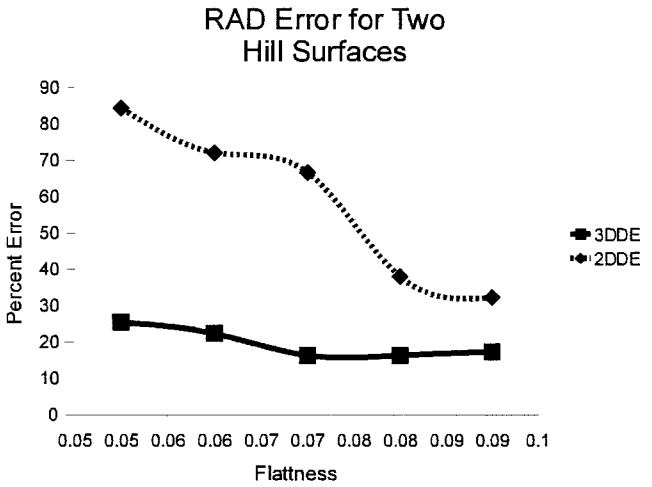

Figure 5.9: Plot of RAD errors versus the steepness of the terrain for the noisy two hill data sets.

superior to the 2DDE. The cases with the larger errors were likely due to the limitations of the weighted L^p DEF on these surfaces. To reduce the errors for both the 2DDE and the 3DDE, we recommend utilizing a more versatile DEF.

To get a closer look at where the errors were coming from on the terrain, Figures 5.10 and 5.11 show the distribution of errors on noiseless and noisy data sets respectively. Each of these figures contain 500 points. The errors are represented by how red each point is. The black points contain the smallest errors while the light red points indicate the highest error.

The trends for both the noiseless and noisy data sets are the same. For surfaces with one and two hills, the ATS favours the tops of the hills. As opposed to this, for the valley surfaces, the ATS produces the best estimates at the bottom of the valley. A more versatile DEF might be useful to make the distribution of error more

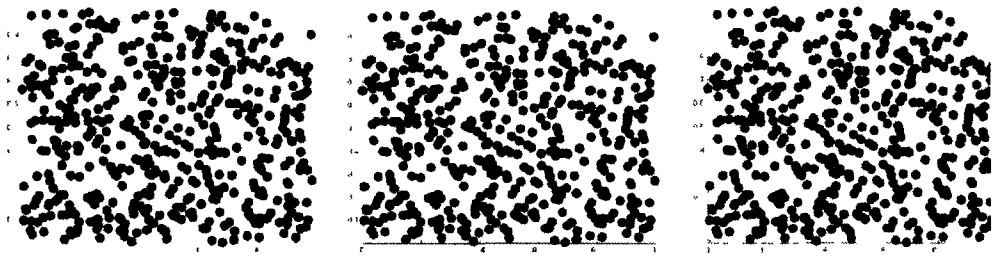

Figure 5.10: The distribution of the errors for the noiseless 3D data sets. In Figure 5.10a (left) the error distribution for a noisy hill surface is shown, Figure 5.10b (center) shows the error distribution for two hills, and Figure 5.10c (right) shows the distribution for a valley surface.

uniform. Another approach to reducing the errors in the problem areas might be to use multi-regional DE in combination with a 3DDEF. For example, the surfaces with two hills have the largest errors. Splitting these surfaces up into two single-hill surfaces may help in reducing the errors. This is an unsolved problem!

5.8 Chapter Summary

In this chapter, we have introduced the idea of DE in three dimensions. 3DDE is identical to 2DDE except that the DEF uses the elevation of the points, as well as the latitude and longitude. The differences between 2DDE and 3DDE were that 3DDE required the DEFs to be adapted for an extra dimension, and to achieve this the 3D weighted L^p DEF was defined in Section 5.2.

To compare the 3D weighted L^p DEF to the 2D weighted L^p DEF, two types of tests were conducted. The first was done on noiseless data sets that were generated using a technique described in Section 5.3. The second set of tests were done on data sets with noise added to the "true" distances. Both of these tests show that

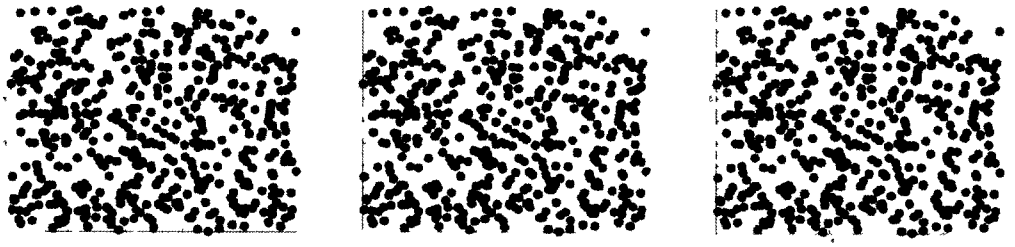

Figure 5.11: The distribution of the errors for the noisy 3D data sets. In Figure 5.11a (left) the error distribution for a noisy hill surface is shown, Figure 5.11b (center) shows the error distribution for two hills, and Figure 5.11c (right) shows the distribution for a valley surface.

the 3DDE method is considerably superior to the 2DDE method for the more "hilly" terrains and for the networks of points.

After examining the overall errors of the two DE methods, we also studied the *trends* of the location of the errors. From these studies (Figures 5.10 and 5.11) we see that there are distinct trends for all of the terrains examined in this chapter. Based on these trends, we have also suggested two possible improvements to the 3DDE method. The first possible improvement is to use a more versatile DEF, such as the one based on the *generalized* weighted L^p DEF in Equation (2.13). The second is to use a multi-regional method of DE for terrains with these topographical trends.

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the contributions of this thesis and suggest some avenues of future work for the ATS, and for the DE problem that has been addressed in this thesis. The contributions are outlined in Section 6.1. The suggested future work is described in Section 6.2. The purpose of these suggestions is to address some gaps and important areas that were not deemed to be within the scope of this thesis.

6.1 Contributions

6.1.1 The ATS

With respect to the ATS, this thesis has made some major contributions. Firstly, it extended the ATS application to the DE problem. We defined both the new environments and the corresponding LA for this problem for three simple DEFs. Using these newly-defined environments and LA, the ATS was shown to produce parameters competitive to those obtained by the hill-climbing search for all of these DEFs.

The second contribution that the thesis made (with regards to the ATS) was

to successfully search for multiple parameters *simultaneously*. To achieve this, we proposed an algorithm in which the ATS could perform a search for multiple parameters, while it still maintained the core foundations of the ATS described in [17] by Oommen and Raghunath. This search has been shown to produce both the optimal parameters in an ideal (non-stochastic) environment, and competitive parameters in a stochastic environment. While we need the algorithm to only find two parameters simultaneously, we believe that it can be extended to the problem of determining more parameters by following the same principles.

6.1.2 Distance Estimation

With regards to DE, the ATS was applied to the problem of DE in order to find the parameters for three different DEFs. The parameters that were determined have been shown to be competitive with parameters computed using a hill-climbing search. The biggest advantage of the ATS over the hill-climbing search is that it does not require a GoF function to determine the parameter. However, we do use these functions to *compare* the qualities of the parameters.

The last major contribution of this thesis was its preliminary use to yield results in the field of 3DDE. The field of DE has predominantly concentrated on two dimensional road networks. Our intention was to use three dimensional data and DEFs to estimate the distances where the road network lies on a particularly hilly terrain. For our preliminary study of 3DDE, we considered mainly terrains with rolling hills or valleys. We then produced a simple road network on top of these terrains that would be used for DE. The results showed that when the terrain possessed a certain amount of "hilliness", the 3DDE outperforms the 2DDE. Our preliminary results for 3DDE demonstrates the advantages of its use under certain conditions and show promise for

practical applications.

6.2 Future Work

6.2.1 The ATS

In this thesis, the algorithm for determining multiple parameters was defined and tested on both an ideal (non-stochastic) environment, and thereafter a stochastic environment. The results of the ideal environment were shown to be optimal while the results for the stochastic environment were compared against another search method to confirm their quality. An important potential direction for this algorithm would be to confirm that the ATS can find the true parameters in an ϵ-optimal manner in certain stochastic environments. The theoretical properties of the stochastic search on the line would be a good foundation by which one can examine these claims. Observe that in this domain, one can fully control both the nature of the environment and the optimal parameters that are to be found.

The application of the ATS in the DE domain shows it can be utilized for practical purposes. The use of the ATS in *other* practical applications holds huge potential. Indeed, we argue that the ATS is useful in other practical domains and that these domains will benefit from involving it because of its useful characteristics.

6.2.2 Distance Estimation

The ATS has been shown to produce competitive results using the three DEFs in Chapter 4; however, the quality of the estimated distance is limited by the DEF that is used. To further take advantage of the ATS search, we recommend that one should consider applying it to a more complex method of distance estimation. One example

of a promising estimator would be the DE method that used stacking (as described in [15]), which uses a combination of distance estimators to improve the resultant accuracy. Applying the ATS to a estimator such as this could produce even better results.

This thesis also produced preliminary results for the 3DDE problem. The results showed the advantages of the 3DDE; however, the errors increased when the terrain became too hilly. There are still many ways to improve the accuracy of 3DDE. Because the 3D weighted L^p DEF had trouble predicting distances on some terrains, we suggest looking into a more adaptive norm like the generalized DEF, or a multi-regional approach. Indeed, any strategy that works for 2DDE can be applied to 3DDE, and should be considered when one wants to improve the accuracy.

Another important avenue for the future research for 3DDE is the possibility of it being applied to a more realistic data set. While our preliminary results are procured from simple networks that were artificially generated, we did this so that many different terrains could easily be examined. We conclude by observing that further research is required on more realistic or real-world road networks, in order that our hypothesis is fully justified.

Bibliography

[1] I. K. Altnel, J. Oommen, and N. Aras, "Vector quantization for arbitrary distance function estimation," *INFOMS Journal on Computing*, vol. 9, no. 4, pp. 439 – 451, 1997.

[2] B. J. Oommen and J. P. R. Christensen, "ϵ-optimal discretized reward-penalty learning automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-18, pp. 451–457, 1988.

[3] R. F. Love and J. G. Morris, "Modelling inter-city road distances by mathematical functions," *Operational Reach Quarterly*, vol. 23, no. 1. pp. 61–71, 1972.

[4] J. H. Walker, *An Empirical Study of Round and Block Norms for Modelling Actual Distances*, Ph.D. thesis, McMaster University, 1991.

[5] R. A. Fildes and J. B. Westwood, "The development of linear distance functions for distribution analysis," *The Journal of the Opertional Research Society*, vol. 29, no. 6, pp. 585–592, 1978.

[6] J. Brimberg, R. F. Love, and J. H. Walker, "The effect of axis rotation on distance estimation," *European Journal of Operational Research*, vol. 80, pp. 357–364, 1995.

[7] H. Erkut and S. Polat, "A simulation model for an urban fire fighting system," *OMEGA - The International Journal of Management Science*, vol. 20, no 4, pp. 535–542, 1992.

[8] H. Uster and R. F. Love, "Application of a weighted sum of order p to distance estimation," *IIE Transactions*, vol. 33, no. 8, pp. 675–684, 2001.

[9] W. Berens, "The suitability of the weighted lp-norm in estimating actual road distances," *European Journal of Operational Research*, vol. 34, no. 1, pp. 39–43, 1988.

[10] R. F. Love and J. G. Morris, "Mathematical models of road travel distances," *Management Science*, vol. 25, no. 2, pp. 130–139, 1979.

[11] R. F. Love and J. G. Morris, "On estimating road distances by mathematical functions," *European Journal of Operational Research*, vol. 36, no. 2, pp. 251–253, 1988.

[12] W. Berens and F. Korling, "On estimating road distances by mathematical functions - a rejoinder," *European Journal of Operational Research*, vol. 36, no. 2, pp. 254–255, 1988.

[13] J. Brimberg and R. F. Love, "A new distance function for modeling travel distances in a transportation network," *Transportation Science*, vol. 26, no. 2, pp. 129–137, 1992.

[14] F. A. Ortega and J. A. Mesa, "A methodology for modelling travel distances by bias estimation," *Sociedad de Estadistica e Investigación Operativa*, vol. 6, no. 2, pp. 287–311, 1998.

[15] E. Alpaydn, K. Altinel, and N. Aras, "Parametric distance metrics vs. nonparametric neural networks for estimating road travel distances," *European Journal of Operational Research*, pp. 230–243, 1996.

[16] J. Oommen, I. K. Altnel, and N. Aras, "Discrete vector quantization for arbitrary distance function estimation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, no. 4, pp. 496 – 510, 1998.

[17] B. J. Oommen and G. Raghunath, "Automata learning and intelligent tertiary searching for stochastic point location," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 28, no. 6, pp. 947 – 954, 1998.

[18] B. J. Oommen and B. Kuipers, "Parameter learning from stochastic teachers and stochastic compulsive liars," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 36, no. 4, pp. 820 – 834, 2006.

[19] R. F. Love. J. H. Walker, and M. L. Tiku, "Confidence intervals for lk,p, distances," *Transportation Science*, vol. 29, no. 1, pp. 93–100, 1995.

[20] J. Perreur and J. Thisse, "Central metrics and optimal location," *Journal of Regional Science*, vol. 14, no. 3, pp. 411 – 421, 1974.

[21] J. B. Wesolowsky and G. O. Wesolowsky, "Probabilistic lp distances in location models," *Annals of Operations Research*, vol. 40, 1992.

[22] Wendell R E. Ward, J E., "Using block norms for location modeling," *Operations Research*. vol. 33, no. 5, pp. 1074–1090, 1985.

[23] A. Yazidi, O. Granmo, and B. J. Oommen, "Service selection in stochastic environments: A learning-automaton based solution," *Applied Intelligence*, 2011.

[24] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: An overview," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 32, no. 6, pp. 711–722, 2002.

[25] M. L. Tsetlin, *Automaton Theory and the Modeling of Biological Systems*, Academic Press, New York, 1973.

[26] M. Agache and B. J. Oommen, "Generalized pursuit learning schemes: New families of continuous and discretized learning automata," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 32, no. 6, pp. 738–749, 2002.

[27] G. I. Papadimitriou and A. S. Pomportsis, "Learning-automata-based TDMA protocols for broadcast communication systems with bursty traffic," *IEEE Communication Letters*, vol. 4, no. 3, pp. 107–109, 2000.

[28] B. J. Oommen and T. D. Roberts, "Continuous learning automata solutions to the capacity assignment problem," *IEEE Transactions on Computers*, vol. C-49, pp. 608–620, 2000.

[29] M. S. Obaidat, G. I. Papadimitriou, A. S. Pomportsis, and H. S. Laskaridis, "Learning automata-based bus arbitration for shared-edium ATM switches," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, vol. 32, pp. 815–820, 2002.

[30] S. Misra and B. J. Oommen, "GPSPA: A new adaptive algorithm for maintaining shortest path routing trees in stochastic networks," *International Journal of Communication Systems*, vol. 17, no. 10, pp. 963 – 984, 2004.

[31] A. F. Atlassis, N. H. Loukas, and A. V. Vasilakos, "The use of learning algorithms in ATM networks call admission control problem: A methodology," *Computer Networks*, vol. 34, no. 3, pp. 341–353, 2000.

[32] A. F. Atlassis and A. V. Vasilakos, "The use of reinforcement learning algorithms in traffic control of high speed networks," *Advances in Computational Intelligence and Learning*, pp. 353–369, 2002.

[33] A. V. Vasilakos, M. P. Saltouros, A. F. Atlassis, and W. Pedrycz, "Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques," *IEEE Transactions on Systems, Man and Cybernetics: Part C*, vol. 33, pp. 297–312, 2003.

[34] F. Seredynski, "Distributed scheduling using simple learning machines," *European Journal of Operational Research*, vol. 107, pp. 401–413, 1998.

[35] J. Kabudian, M. R. Meybodi, and M. M. Homayounpour, "Applying continuous action reinforcement learning automata (CARLA) to global training of hidden markov models," in *Proceedings of the International Conference on Information Technology: Coding and Computing , ITCC'04*, Las Vegas, Nevada, 2004, pp. 638–642.

[36] M. R. Meybodi and H. Beigy, "New learning automata based algorithms for adaptation of backpropagation algorithm pararmeters," *International Journal of Neural Systems*, vol. 12, pp. 45–67, 2002.

[37] C. Unsal, P. Kachroo, and J. S. Bay, "Simulation study of multiple intelligent vehicle control using stochastic learning automata," *Transactions of the Society for Computer Simulation International*, vol. 14, pp. 193–210, 1997.

[38] J. Oommen and S. Misra, "Cybernetics and learning automata," in *Springer Handbook of Automation*, Shimon Y. Nof, Ed., pp. 221–235. Springer Berlin Heidelberg, 2009.

[39] M. L. Tsetlin, "On the behavior of finite automata in random media," *Automation and Remote Control*, vol. 22, pp. 1210–1219, 1962.

[40] R. R. Bush and F. Mosteller, *Stochastic Models for Learning*, John Wiley & Sons, New York, 1958.

[41] V. I. Varshavskii and I. P. Vorontsova, "On the behavior of stochastic automata with a variable structure," *Automation and Remote Control*, vol. 24, pp. 327–333, 1963.

[42] S. M. Ross, *Introduction to Probability Models*, Academic Press, 2003.

[43] V. Krylov, "On the stochastic automaton which is asymptotically optimal in random medium," *Automation and Remote Control*, vol. 24, pp. 1114–1116, 1964.

[44] V. I. Krinsky, "An asymptotically optimal automaton with exponential convergence," *Biofizika*, vol. 9, pp. 484–487, 1964.

[45] S. Lakshmivarahan, "Two person decentralized team with incomplete information," *Appl. Math. and Computation*, vol. 8, pp. 51–78, 1981.

[46] N. Baba, T. Soeda, and Y. Sawaragi, "An application of stochastic automata to the investment game," *Int. J. Syst. Sci.*, vol. 11, no. 12, pp. 1447–1457, 1980.

[47] M.K. Hashem, *Learning Automata Base Intelligent Tutorial-Like Systems*, Ph.D. thesis, Ottawa-Carletn Institute for Computer Science, 2007.

[48] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Prentice-Hall, Inc., 1989.

[49] G. Skorobohatyj, "MP-TESTDATA - The TSPLIB symmetric traveling salesman problem instances," 2011, Available as of September 12, 2011 at http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

[50] "Google maps," 2011, Available as of September 12, 2011 at http://maps.google.com/.