Novel Distance Estimation Methods Using "Stochastic Learning on the Line" Strategies

Jessica Havelock, B. John Oommen, IEEE Fellow, and Ole-Christoffer Granmo

***Abstract*—In this paper, we consider the problem of Distance Estimation (DE) when the inputs are the x and y coordinates (or equivalently, the latitudinal and longitudinal positions) of the points under consideration. The aim of the problem is to yield an accurate value for the real (road) distance between the points specified by the latter coordinates[1]. This problem has, typically, been tackled by utilizing parametric functions called "Distance Estimation Functions" (DEFs). The parameters are learned from the training data (i.e., the true road distances) between a *subset* of the points under consideration. We propose to use Learning Automata (LA)-based strategies to solve the problem. In particular, we resort to the Adaptive Tertiary Search (ATS) strategy, proposed by Oommen *et al.*, to affect the learning. By utilizing the information provided in the coordinates of the nodes and the true distances from this *subset*, we propose a scheme to estimate the inter-nodal distances. In this regard, we use the ATS strategy to calculate the best parameters for the DEF. Traditionally, the parameters of the DEF are determined by minimizing an appropriate "Goodness-of-Fit" (GoF) function. As opposed to this, the ATS uses the current estimate of the distances, the feedback from the Environment, and the set of known distances, to determine the unknown parameters of the DEF. While the goodness-of-fit functions can be used to show that the results are competitive, our research shows that they are rather not *necessary* to compute the parameters themselves. The results that we have obtained using artificial and real-life datasets demonstrate the power of the scheme, and also validate our hypothesis that we can completely move away from the GoF-based paradigm that has been used for four decades.**

Based on the latter results, the paper also suggests a completely novel method by which we can extend the traditionally-studied DE problem where the road distances were, typically, estimated using only the x and y coordinates (or equivalently, the latitudinal and longitudinal positions). In such a generalized model, we hypothesize that one can also provide to the system the additional z coordinate, that represents the height or elevation of the subset of nodes and of the cities whose inter-city distance is to be estimated. The results for this generalized model are currently being compiled into a companion paper.

Keywords : *Road Distance Estimation, Estimating Real-life Distances, Learning Automata, Adaptive Tertiary search, Stochastic Point Location*

I. Introduction

There are many well-studied problems whose solutions depend upon the distances between points in the Cartesian plain or in a geographic region. The traveling salesman problem, and vehicle scheduling problems are common examples of real-life scenarios that rely on distance information. The input to these Distance Estimation (DE) 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 traveled between a pair of known locations, is trivial. However, determining the actual "road distances" (the physical distance to be traveled on the "roads" built in the community) for an area, is more challenging. These road distances (also synonymously known as traveling 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 DE 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[2], vehicle routing, and in the partitioning of districts for firefighters [8, 9, 26]. Indeed, as alluded to earlier, this is a central issue in designing GISs and GPSs.

To initiate discussions, we first informally formulate the DE problem. Consider a table of distances in which the points or cities 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 *small* subset of these distances is actually known to any given precision. 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 access 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

A very preliminary version of this paper appeared in the *Proc. of IEA/AIE'18, the 2018 International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, held in Montreal, Canada, in June 2018. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. e-mail address: jhavelock2436@gmail.com.

Author's status: *Chancellor's Professor*; *Fellow: IEEE* and *Fellow: IAPR*. This author can be contacted at: School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6. This author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. E-mail address: oommen@scs.carleton.ca. This author also holds an *Adjunct Professorship* with the Department of Information and Communication Technology, University of Agder, Grimstad, Norway. He gratefully acknowledges the partial support of NSERC, the Natural Sciences and Engineering Council of Canada.

Author's status: This author can be contacted at: Centre for Artificial Intelligence Research, Department of Information and Communication Technology, University of Agder, Grimstad, Norway. e-mail address: ole.granmo@uia.no.

[1]This is a typical problem encountered in a Geographic Information System (GIS), or in a GPS. However, unlike the traditional systems where *all* the inter-city distances are assumed to be stored, in our setting, the distance between *any pair of cities* is assumed to be *computed* by merely having access to a *small subset* of *known* inter-city distances.

[2]Two common vehicle scheduling software packages, ROADNET and TRUCKSTOPS 2, use DE methods when determining the distances between the suppliers and the customers [8].

and practical purposes. Further, it is unarguably infeasible to store the actual distances when the number of villages/cities, specified by the size of the table, is in the thousands or tens of thousands.

From an abstract perspective, the fact that DE is used for approximating distances in road networks, does not mean that it is limited in its applications. The most obvious application is in modelling travel times. Thus, DE has been used by companies to schedule deliveries [8]. A more "academic" problem, seen in the use of DE in modeling travel times, involves location and transportation problems (the traveling salesman problem) in geographical areas where the distances are *almost entirely* unknown. DE has also been useful in resolving location analysis and optimization problems. Thus, it has found direct applications in urban planning. For example, in Turkey, DE has been used to determine the optimal locations for fire stations [9]. Otherwise, road distance estimation has been applied in road network simulations to take into account network impedance, for tasks such as OEM picking-up routing [22].

In general, whenever a problem requires the knowledge of all the distances between all possible points in the domain, the invocation of DE can be highly advantageous. DE can also be used in in sailing and under-water navigation, where the distance to be traversed has to also consider rocks and islands, and possibly, ocean currents. Indeed, while in the simplest setting these estimated distances are geographical distances, in a more general case, for example, dealing with chemical structures, these estimated distances may be far more abstract "navigational" distances required to change the structure from one configuration to another[3].

A. Legacy Methods: Distance Estimation Functions

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, historically, one typically resorts to Distance Estimating *Functions* (DEFs). These functions can take on any form, but the ideal ones are those that are simultaneously good estimators, and that are also characterized by 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 [16]. 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 being characterized. Consequently, some distances in the system must be known *a priori*, and they are used to "learn" the parameters associated with the DEFs. The accuracy of the estimations depends on the DEF, the system and the available data.

For over four decades, DEFs have been applied in DE and the methods utilized have been extensively tested and modified [4, 5, 16, 17, 18, 35, 36]. Over the years, new DEFs have been proposed, based on other distance measures [7, 30]. 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 [7, 30]. Another novel idea for a DEF is to use a *nonparametric* method [1]. DEFs have further been used in business models, including in the modeling of service systems, and in operational and strategic decision processes [6].

A new strategy by which DE has been improved, has been 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 [8]. This was done so as 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. The latter approaches divide the original region into smaller sub-regions, and each sub-region has its own trained parameters. One example of such a multi-regional approach to DE was presented by Fildes and Westwood [10]. The scheme estimated the distances based on the sub-regions that lay between the two points. The sub-regions themselves determined the weighting of the parameters used in the DEF, which, in turn, was based on the proportion of the Euclidian distance that lay in each sub-region. Another multi-regional approach used Vector Quantization (VQ) [3, 26]. This strategy utilized the points learned by the VQ to represent closed groups of points. The parameters that were used for inter-regional and intra-regional distance estimation themselves were 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 [1]. They compared these two methods with the voting and stacking of typical DEF-based DE methods. The overall leader was the one which used the stacking of simple DEFs.

B. Our Proposed Approach

In this paper, we will contribute to the field of DE by applying a new method for determining the DEF. This method is called the Adaptive Tertiary Search (ATS) which was derived by Oommen and Raghunath [28]. To date, it has been applied to two problems, namely, the continuous Stochastic Point Location problem [28], and the problem of parameter learning from a stochastic teacher or a stochastic compulsive liar [29]. Both of these problems work within a stochastic domain analogous to that of DE. The ability of the ATS to perform ε-optimally 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 determine or locate the parameter sought for. As explained presently, the most "daring" step that we have taken is that we have completely moved away from invoking Goodness-of-Fit (GoF) criteria for the DEFs, thus proposing a marked departure from the methods that have been used for more than four decades.

LA [21] are autonomous or self-operating machines that have the ability to learn in stochastic environments. They are typically used to determine information and make decisions in environments that have incomplete knowledge [38]. They are able to learn in these environments because the search for the

[3]In the interest of being focused, we shall concentrate here on the traditional problem of using DE for estimating "road" distances.

information is actually preformed in a probability space and not the action space itself [34]. 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 is interacting with and which it can observe. For an LA, the mechanism of decision making depends on its structure and the environment it is in.

Put in a nutshell, the strategy we propose for using the ATS to solve DE is as follows: At any given time instant, the system has certain values for the parameters which characterize the DEF in question. With this as a premise, as per the LA's inference philosophy, the ATS searches a bounded interval by comparing the current estimate to the feedback from the environment, where the latter is presented in terms of pairs of points and their corresponding known distances. After comparing the current estimate with the latter, a new smaller search interval for the parameters of the DEF is inferred. The search interval is continually decreased until it reaches a small-enough size.

It is crucial to emphasize the following: Apart from the convergence phenomena, the rationale and goal for using the ATS is to improve the accuracy of the DEFs by *totally* removing any dependencies on the GoF functions.

C. Outline of the Paper

This paper starts with an overview of topics related to the field of DE. The objective of this section is to briefly present the reader with relevant background information and with a collection of existing research, in a well-organized fashion. Section III deals with the ATS and Section IV explains the adaptations required in order for the ATS to operate in a DE domain. Thereafter, in Section IV-D, we explain the analysis of the two-dimensional ATS, and explains its use in DE. The adapted ATS and the environments that are defined are then tested in the subsequent section, Section V. This includes the testing procedure, the test results and a discussion for the results obtained for the two-dimensional ATS. In two dimensions, the intention is to demonstrate that the ATS operates ε-optimally 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 demonstrate that the ATS produces competitive parameters for the DEF, for the purpose of comparison, we have invoked a hill-climbing search. This is quite realistic, because in the literature, the parameters were either found by a linear regression or a hill-climbing search [7, 8, 36]. As a result, the parameters found by means of such a hill-climbing search proved, unequivocally, that it is a natural method of showing that the ATS's parameters are competitive. Section VI concludes the paper.

D. Contributions of the Paper

The following list highlights the three primary contributions of this paper:

1) The first contribution of this paper is the extension of the ATS so as to involve determining multiple parameters simultaneously, and utilizing this strategy in DE.
2) The second contribution involves the improvements to the field of DE itself by virtue of 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.
3) The third contribution is the "proof of the pudding" of these results in DE, where we submitted experimental results for artificial noiseless data, artificial noisy data, and for real-life data. The results that we have obtained are, truly, conclusive.
4) Our final contribution lies in the fact that we have suggested how we can use the same concepts for 3D DE. The details of these results, where we show that the 3D DE problem for estimating distances over certain types of "hilly" terrains is solvable, are currently being compiled for a separate publication.

II. Distance Estimation: Core Concepts

A. Formalizing Distance Estimation

The distance[4] 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. For example, in Figure 1, the blue path represents the true path (the "road" path) and the red line is the direct path (as the bird flies) between Carleton University and Ottawa University in Ottawa, Canada, and the goal of the DE problem is to estimate the length of the blue path as accurately as possible. 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.

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 also result in errors due to the changes caused by different coordinates systems or by the points from which the distances are measured [36]. For example, in 1982, the Province of Ontario published all the distances between large

[4]The entire section describing DE, DEFs and Goodness-of-Fit functions is included in the interest of completeness. It can be abridged or even deleted if recommended by the Referees.

Fig. 1: The blue path represents the true path (the "road" path) and the red line is the direct path (as the bird flies) between Carleton University and Ottawa University in Ottawa, Canada.

intersections throughout the province [36], 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. 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 is a challenge when one needs to 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 the 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 by *merely using* the information that is provided.

The prediction or DE, is typically done by determining or discovering the appropriate DEF. A DEF is a mapping from $R^d \times R^d$ 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 $\pi(P_1,P_2|\Lambda) : R^d \times R^d \longrightarrow R$, where $P_1 = \langle x_1,x_2,...,x_d\rangle$ and $P_2 = \langle y_1,y_2,...,y_d\rangle$ are points in R^d, and Λ 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, Λ, is typically learnt by minimizing a goodness-of-fit function, which, in turn, is used to measure how well a network or region is represented by the DEF.

B. Goodness-of-Fit

Central to the legacy methods of DE is the concept of Goodness-of-Fit (GoF) functions. 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 by Eq. (1) [1, 3, 4, 8, 10, 16, 17, 18, 26, 30, 35], which was originally proposed by Love and Morris. It is given by the sum of the absolute values of the differences between the estimated distances and the actual distances, and is given as:

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

where $A(P_i,P_j)$ is the actual distance between the points P_i and P_j. Observe that this GoF measure, used in[1, 3, 4, 8, 10, 16, 17, 18, 26, 30, 35], leads us to a natural way of minimizing the overall error of the DEF, although larger distances are given more weight than smaller ones [3, 8, 16, 26].

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 Eq. (2) is often recommended [4]:

$$RAD = \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} |A(P_i,P_j) - \pi(P_i,P_j|\Lambda)|}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} A(P_i,P_j)}. \tag{2}$$

The next and third most-common GoF function is the Normalized Absolute-value Difference (NAD), given by Eq. (3) [8, 19, 35]. Although like the AD, the NAD uses the sum of the absolute values of the differences between the distances, unlike the former, it however, normalizes these differences with respect to their respective actual quantities. This allows the errors for both large and small distances to be weighted equally [8, 19]. Thus, while the AD measures the overall error, the NAD quantifies the weighted percentage error for each distance as follows:

$$NAD = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{|A(P_i,P_j) - \pi(P_i,P_j|\Lambda)|}{A(P_i,P_j)}. \quad (3)$$

In spite of the advantage of the above indices, the most commonly-used GoF function is the sum of Square Deviation (SD) given by Eq. (4) [1, 3, 4, 8, 10, 16, 17, 18, 26, 35].

$$SD = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{A(P_i,P_j) - \pi(P_i,P_j|\Lambda)}{\sqrt{A(P_i,P_j))}} \right)^2. \quad (4)$$

This index was also proposed by Love and Morris, and has useful properties such as its statistical significance and continuity. The SD is both continuous and differentiable with respect to the parameter, Λ. This allows the user to employ a gradient decent search scheme to determine the "best" parameter Λ [1, 3, 26, 35]. Although the SD favors larger distances, it does not possess as large a bias as the AD does [1, 8, 16].

C. 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.

1) L^p-based DEFs: The properties of norms 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-understood and established norm/metric. The most common types of DEFs are those based on the family[5] of L^p norms, traditionally used for computing distances:

$$L_p(X) = \left(\sum_{i=1}^{n} (|x_i|^p \right)^{1/p}. \quad (5)$$

The various L^p norms have been used as stepping stones to design DEFs, and some of the most common DEFs have, indeed, been derived from the L^p norms as shown[6] in Table I (Eq. (6) to (11)), also initially suggested by Love and Morris [16, 17]. 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 the subset of the co-ordinates of the cities and *their* known inter-point distances for the specific region under consideration. This training is done so as to obtain the "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 [1, 4, 8, 16, 17, 18, 19, 26, 31, 35, 37]. In particular, Eq. (6) performed well[7] over a small area, usually described by the boundaries of a given city, but did not perform well over larger areas [16]. Eq. (7) produced usable results for both a network of roads from within a city and which extended to other cities [16]. The Weighted Euclidean DEF in Eq. (7) outpreformed the L^p DEF in Eq. (6); however, both of these were out-performed by the Weighted L^p DEF given by Eq. (8) [16].

The DEF described by Eq. (8) has an advantage 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 Eq. (8) over Eq. (7) has a large variation depending on the geographical area being studied. In a study by Berens, the percentage improvement, based on the GoF criteria, ranged from 0% to 11.27% [1, 5]. 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 an AD improvement of 0.12% and a SD improvement of 0.03% in Germany with 117 cities and 6786 distances [18]. Love and Morris reported AD and SD improvements of 2.4% and 8.3% respectively in Germany with 15 cities and 100 distances [18]. In the United States, Love and Morris reported AD and SD improvements of 10.55% and 34.63% respectively when using Eq. (8) over Eq. (7). 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 differences in the road structures. Berens and Korling stated 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 Eq. (8)). 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.

Although the Weighted L^p DEF yields good results, it is outperformed by the kL_{ps} DEF in Eq. (9) [17] characterized by three parameters. The DEF has been shown to be statistically stronger than the Weighted L^p DEF. Of all the DEFs listed in Table I, Eq. (8) performed second-best in all types of regions. It performed well in both rural and urban networks, but was out-performed by Eq. (9) [16]. Eq. (9) was the best estimator in urban settings [17]. However, the best DEF for urban areas was the Elliptical DEF described in Eq. (11), which performed well in networks that were not highly developed but did not perform well otherwise [17].

In 2000, Uster and Love proposed using a Generalized Weighted L^p DEF, given by Eq. (10). They reported that the generalized Weighted L^p norm yields significant improvements for areas with directional "non-linearity". However, in areas with little directional "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 I, Eq. (9) is the best-performing DEF and is the best estimator for the generic situation [16, 17].

The general conclusion offered in the literature is that the specific DEF should be chosen based on the properties of the

[5]The cases for $p = 1$, $p = 2$ and $p = \infty$ represent the Taxi-Cab, Euclidean and Largest Absolute Value norms respectively. The L^p norms for other values of p ($p \in R$) also have significance in DE.

[6]We apologize for the placement of the tables in the manuscript. It was not dictated by us, but by the IEEE LaTex style files.

[7]In performing comparisons, there are essentially two main GoF criteria: the AD given by Eq. (1), and the SD specified by Eq. (4).

Norms Derived from L^p norms				
DEF	Expression	Name of DEF	Parameters	Equation
F_1	$\left(\sum_{i=1}^{d} \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/p}$	L^p	p	(6)
F_2	$k\left(\sum_{i=1}^{d} \lvert x_{1i} - x_{2i} \rvert^2\right)^{1/2}$	Weighted Euclidean	k	(7)
F_3	$k\left(\sum_{i=1}^{d} \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/p}$	Weighted L^p	k and p	(8)
F_4	$k\left(\sum_{i=1}^{d} \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/s}$	kL_{ps}	k, p, and s	(9)
F_5	$\left(\sum_{i=1}^{d} (k_i * \lvert x_{1i} - x_{2i} \rvert^p\right)^{1/p}$	Generalized Weighted L^p	$\{k_i\}$ and p	(10)
F_6	$[(X_1 - X_2)' M (X_1 - X_2)]^{1/2}$	Elliptical DEF	$M = \begin{bmatrix} k & p/2 \\ p/2 & s \end{bmatrix}$	(11)

TABLE I: List of DEFs related to various L^p norms and their corresponding parameters.

geographical area in question. Of course, as explained later, a multi-regional approach for areas with different geographical features, is always superior[8].

We now proceed with the specific contribution of this paper, namely, the use of the the Adaptive Tertiary Search (ATS) in DE.

III. The Adaptive Tertiary Search

The solution that we propose for DE is based on a scheme relevant to the Stochastic Point Location (SPL) problem. To formulate the SPL, we assume that there is a Learning Mechanism (LM) whose task is to determine the optimal value of some variable (or parameter), λ. We assume that there is an optimal choice for λ - an unknown value, say $\lambda^* \in [0,1]$. In the interest of completeness, we list the available solutions to the SPL:

1) The first-reported SPL solution proposed the problem itself, and then pioneered a solution operating in a discretized space [24];
2) The Continuous Point Location with Adaptive Tertiary Search (ATS) solution was a solution in which three LA worked in parallel to resolve it [28];
3) The extension of the latter, namely the Continuous Point Location with Adaptive d-ARY Search (CPL-AdS), used 'd' LA in parallel [28], and these could operate in truth-telling and deceptive Environments;
4) The General CPL-AdS Methodology extended the CPL-AdS to possess all the properties of the latter, but could also operate in non-stationary Environments [15];
5) The Hierarchical Stochastic Search on the Line (HSSL) proposed that the LM moved to distant points in the interval (modelled hierarchically), and specified by a tree [40];
6) The Symmetrical Hierarchical Stochastic Search on the Line (SHSSL) symmetrically enhanced the HSSL to work in deceptive Environments [42];
7) The Adaptive Step Search (ASS) used historical information within the last three steps to determine the current step size [33].
8) The Thompson Sampling (TS)-guided Stochastic Point Location (TS-SPL) scheme introduced the first Bayesian representation of the SPL, that also overviewed the complete search space at every time instant [11, 12]. Based on the so-called Thompson Sampling [14], both the location of λ^* and the probability of receiving correct feedback were simultaneously learned, allowing TS-SPL to operate in both deceptive and non-stationary Environments.

[8]The literature also reports various neural and Vector Quantization schemes suitable for DE. Since we are using ATS and LA-based schemes for learning the parameters of DEFs, these are not surveyed here.

In this paper, we shall use the ATS [28] to solve the DE problem, although any of the other-reported solutions could have been used just as well. 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 [28], the ATS was applied to a stochastic environment, and the ability of the ATS to function in such environments makes it ideal for the DE problem.

As alluded to above, Oommen and Raghunath used the ATS to determine λ^*, in a bounded interval within a resolution of accuracy. In their work, the Oracle or Environment is modeled as a "Stochastic Teacher" [28], implying that it provides a correct response with a probability greater than 0.5 [29]. This Environment (the "Stochastic Teacher") for the SPL problem provides feedback about the location of the point in question, i.e., whether λ^* is to the right or to the left of the currently chosen $\lambda(n)$.

To determine λ^* 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 $\lambda(n)$, the estimated position of λ^* 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 II. 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 [28]. After the search interval has been sufficiently reduced, the midpoint of the final search interval is returned as the estimate for λ^*. The ATS algorithm can be seen in Algorithm 1.

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 II: Decision Table

Algorithm 1 ATS Algorithm

Input: The Resolution, ρ

Output: Estimate of λ^*

Method:

repeat
  $(\Delta^1, \Delta^2, \Delta^3) \leftarrow Get partitions(\Delta)$
  for $j \leftarrow 1$ to 3 **do**
    Get position of λ^* from LA_j
  end for
  $\Delta \leftarrow$ Get new search interval from Table II
until Size of Interval $< \rho$
$\lambda^* \leftarrow$ Midpoint(Δ)

End Algorithm

Algorithm 2 TwoDimensionalATS

Input: The Resolutions ρ_k and ρ_p

Output: Estimates of $k^*$and p^*

Method:

repeat
  for $j \leftarrow 1$ to 3 **do**
    Execute LA^j for k
    Execute LA^j for p
  end for
  GetNewInterval for k - From Table II
  GetNewInterval for p - From Table II
until (Size_of_Interval(k) $< \rho_k$) $\wedge$ (Size_of_Interval(p) $< \rho_p$)
$k^* \leftarrow$ Midpoint(FinalInterval(k))
$p^* \leftarrow$ Midpoint(FinalInterval(p))

End Algorithm

IV. ATS for Distance Estimation

The ATS proposed by Oommen and Raghunath [28] was initially used to solve the SPL problem, and subsequently for parameter learning when interacting with a stochastic teacher/compulsive liar [28, 29]. For both of these problems, one had to determine only a single unknown parameter. Our aim is to utilize these core concepts in DE where one has to learn/estimate many parameters simultaneously. In order to adapt the ATS to find more than a single parameter, we must specify the corresponding "Environment", and also both the process of updating multiple search intervals and the issue of how the set of LA interacts with it.

A. Updating Search Intervals

Let us first consider the case where the DEF has two parameters, say k and p. The strategy for our search will be to use the ATS to determine the best value for k and p, say k^* and p^*, respectively. However, it is crucial that the *order* of updating the search intervals in the k and p spaces is considered when determining these multiple unknown parameters. If this is not done correctly, it may result in the premature reduction of a search interval. In the SPL problem, the subintervals were first searched using the LA, after which the search interval was updated. This order of executing the searching, and the pruning of the intervals must also be maintained while searching for the two parameters, k and p, simultaneously. In other words, all the subintervals must be searched before any interval is updated. Each search interval must undergo the same search process as in the case of the single-parameter ATS. The only difference is that the search intervals must be updated simultaneously. The order (or sequence) for achieving this is shown in Algorithm 2.

The set of LA operate in the same manner as in [28], except for how it deals with the additional parameters. When the LA is learning information about how it should update the value for k, it uses values of p from within its *current* search interval and vice versa. As a result, each LA operates with the knowledge of 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 IV-C.

We shall now describe in greater detail the various modules of the system.

B. The Corresponding LA

Each LA is provided with two inputs, namely the parameter that it is searching for, and all the search intervals. Each LA is required to yield as its output the relative location of the parameter in question. It does this by producing a decision (Left, Right or Inside) based on *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 (i.e., the decision is "Right") 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's response is positive.

The LA and the Environment repeat this loop for a large number, say N_∞, iterations. After they are done communicating, the LA produces its output as per Algorithm 3. If the LA's belief of "Right" is greater than $1-\varepsilon$, the parameter in question is to the right side of the current search interval, and so its output is "Right". Conversely, if the belief of "Left" is greater than $1-\varepsilon$, the LA's final decision is "Left". If neither of these cases emerge, the LA does not have a belief greater than $1-\varepsilon$ that the parameter is to the "Right" or "Left", and in this case, the LA decides that the parameter's optimal value is "Inside" the present interval. The entire algorithm is formally given in Algorithm 3, where θ_R ($0 \ll \theta_R < 1$) is the (L_{RI}) learning coefficient.

C. 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. It is easy to obtain this answer because it only involves a single parameter at a time. To further explain this, consider the DEF

Algorithm 3 The LA algorithm

Input: Parameter to be determined, Search interval, θ_R, N_∞.
Notation: p_L and p_R are the probabilities of choosing the left and right sub-intervals respectively.

Output: Decision $\in$ {Left, Right, or Inside}

Method:

for $i = 1$ to N_∞ **do**
  CurrentAction $\leftarrow$ ChooseAction
  Feedback $\leftarrow$ GetEnvironmentsResponse(CurrentAction) - From either Algorithms 4 or 5
  if Feedback == Agree **then**
    if CurrentAction == Left **then**
      $p_R = p_R * (1 - \theta_R)$
      $p_L = 1 - p_R$
    end if
    if CurrentAction == Right **then**
      $p_L = p_L * (1 - \theta_R)$
      $p_R = 1 - p_L$
    end if
  end if
end for
if $p_R > 1 - \varepsilon$ **then**
  Return Right
else if $p_L > 1 - \varepsilon$ **then**
  Return Left
else
  Return Inside
end if

End Algorithm

in Eq. (8) which can be simplified into two equations, Eq. (12) and Eq. (13) as below:

$$F(k,p) = k \cdot F_1(X_1, X_2, p), \text{and where,} \tag{12}$$

$$F_1(X_1, X_2, p) = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p \right)^{1/p}. \tag{13}$$

Although nothing specific can be said about the monotonicity characteristics of $F(k,p)$, we see from Eq. (12) that by virtue of the fact that it is always positive and that it can be factored, it is monotonically *increasing* with k for any fixed value, p. Similarly, from Eq. (13), since $F_1(X_1, X_2, p)$ is not a function of k, it is monotonically *decreasing* with p for any fixed value of k. These properties allow the Oracle to respond according to Algorithm 4 when finding k, and for the corresponding LA to move in the desired direction (i.e., "Left" or "Right") in the space that only involves the single parameter k. The contrary monotonicity properties allow the Oracle to respond according to Algorithm 5 when determining p, and for the corresponding LA to move in the desired direction (i.e., "Left" or "Right") in the space that involves only p.

We now consider how we can optimally take advantage of the above-mentioned monotonicity properties. This is done by the DEF using either the *Max* or *Min* values of the other parameter while it evaluates the estimated distance between the points under consideration. Whether the evaluation is done with the *Max* or *Min* point itself depends on the choice that the LA is making. The two possible cases for Algorithm 4 are listed below:

1) If k has to be decreased, the value of p must be chosen so as to minimize the decrease in k so that it is achieved in a conservative manner. Thus, if the parameter k is chosen from the left half of the search interval, its value in the DEF is chosen randomly from the left-half of k's region, but with p being at *its* largest value, p_{Max}. If under these settings, the estimated distance based on the DEF is larger than or equal to the true distance, the Environment provides the corresponding LA with a Reward.
2) If the parameter k is chosen from the right half of the search interval, its value in the DEF is chosen randomly from the right-half of k's region but with p being at *its* smallest value, p_{Min}. Again, if under the latter settings, the estimated distance based on the DEF is smaller than or equal to the true distance, and the Environment provides the corresponding LA with a Reward.

The analogous two possible cases for Algorithm 5 are the following:

1) If p has to be decreased, the value of k must be chosen so as to minimize the decrease in p so that it is achieved in a conservative manner. If p is chosen from the left half of the search interval, its value in the DEF is chosen randomly from the left-half of p's region, but with k being at *its* largest value, k_{Max}. If under these settings, the estimated distance based on the DEF is smaller than or equal to the true distance, the Environment provides the corresponding LA with a Reward.
2) If the parameter p is chosen from the right half of the search interval, its value in the DEF is chosen randomly from the right-half of p's region but with k being at *its* smallest value, k_{Min}. Again, if under the latter settings, the estimated distance based on the DEF is larger than or equal to the true distance, the Environment provides the corresponding LA with a Reward.

In both the above cases, the Environment otherwise yields a negative or Penalty response.

Algorithm 4 EnvironmentResponse(k)

Input: Training Distances; Action chosen by the LA, and the current intervals of k and p.
Notation: The evaluation of $F(k,p)$ is done at a point k_r randomly chosen from the interval under consideration, and with p being at either the maximum or minimum value in its region.

Output: Decision $\in$ {Agree, Disagree} as far as the parameter k is concerned.

Method:

if ((Choice == Left) $\wedge$ ($F(k_r, p_{Max}) \geq$ TrueDistance)) **then**
  return Agree
else if ((Choice == Right) $\wedge$ ($F(k_r, p_{Min}) \leq$ TrueDistance)) **then**
  return Agree
else
  return Disagree
end if

End Algorithm

Algorithm 5 EnvironmentResponse(p)

Input: Training Distances; Action chosen by the LA, and the current intervals of k and p.

Notation: The evaluation of $F(k,p)$ is done at a point p_r randomly chosen from the interval under consideration, and with k being at either the maximum or minimum value in its region.

Output: Decision $\in$ {Agree, Disagree} as far as the parameter p is concerned.

Method:

1: **if** ((Choice == Right) $\wedge$ ($F(k_{Min},p_r) \geq$ TrueDistance)) **then**
2: return Agree
3: **else if** ((Choice == Right) $\wedge$ ($F(k_{Max},p_r) \leq$ TrueDistance)) **then**
4: return Agree
5: **else**
6: return Disagree
7: **end if**

End Algorithm

D. The ε-optimality of the Multi-parameter ATS

We now consider the ε-optimality of the Multi-parameter ATS. The proof of the scheme relies heavily on the proof of the corresponding single-parameter ATS which was earlier rigorously proven in [28]. Consequently, to avoid repetition, we shall merely cite the results from [28] wherever they are needed, and not re-iterate the fine details of the proofs of the assertions here.

To prove that the team of LA described above operate ε-optimal, we will first show that the Environment that we have defined above provides the correct response with $p > 0.5$.

Lemma 1: The Environment defined above provides the correct response with $p > 0.5$ whenever the training data has an accuracy greater than 50%.

Proof: We shall first prove the claim when we are dealing with the parameter k and when p is maintained constant. The proof for the alternate case when we are dealing with the parameter p and when k is maintained constant follows almost identical arguments.

Let us start with an arbitrary learning loop for the parameter k. Let k' be the randomly chosen value for k for this loop, and let the current search interval for k be denoted as $\Delta_k = [A,B]$. Note that the DEF, $F(k,p)$ is monotonically increasing with respect to k. Now, with respect to the current interval for p, let p_{Max} be the value of p that minimizes[9] $F(\cdot,\cdot)$, in this region, and let p_{Min} be the value that maximizes it. We specify below the probability of the defined Environment producing an error. To do this, we consider two mutually exclusive and exhaustive cases, namely when k^* is be to the left of the current interval, and when k^* is to the right of the current interval.

Case 1: Let k^* be to the left of the current interval, and $Tdist$ be the true distance provided by the points under consideration in the training data.

We now partition Δ_k into three mutually exclusive and exhaustive parts as $\Delta_k = P1 \cup P2 \cup P3$, where: P1 =$[A,k_1]$ - such that $\forall k \in [A,k_1]$, $F(k,p_{Max}) < Tdist$ or $F(k,p_{Min}) > Tdist$,
P2 =$[k_1,k_2]$ - such that $\forall k \in [k_1,k_2]$, $F(k,p_{Max}) > Tdist$ and $F(k,p_{Min}) < Tdist$,
P3 =$[k_2,B]$ - such that $\forall k \in [k_2,B]$, $F(k,p_{Max}) < Tdist$ or $F(k,p_{Min}) > Tdist$.
These partitions fully divide the search interval due to the monotonicity properties of $F(\cdot,\cdot)$. Consequently:

$$\begin{aligned} &Pr(error) \\ &\quad = \quad Pr(error|k' \in P1)Pr(k' \in P1) \\ &\qquad +Pr(error|k' \in P2)Pr(k' \in P2) \\ &\qquad +Pr(error|k' \in P3)Pr(k' \in P3) \end{aligned}$$

By considering the P2 term more clearly, we see that:

$$\begin{aligned} &Pr(error|k' \in P2)Pr(k_1 \leq k' \leq k_2) \\ &< Pr(error|k' \in P2, k' < k_{mid})Pr(k_1 \leq k' \leq k_{mid}) \\ &+Pr(error|k' \in P2, k' < k_{mid})Pr(k_{mid} \leq k' \leq k_2) \\ &= (1)Pr(k_1 \leq k' \leq k_{mid}) + (0)Pr(k_{mid} \leq k' \leq k_2) \\ &= Pr(k_1 \leq k' \leq k_{mid}) \end{aligned} .$$

We now observe that k^* is to the left and that the Environment would thus always provides a negative response. Since the error in the intervals $[A,k_1]$ and $[k_2,B]$ come entirely from the data:

$$Pr(error|k' \in [A,k_1]) < 0.5, \text{ and}$$
$$Pr(error|k' \in [k_1,B]) < 0.5.$$

This renders the probability of the Environment returning an error to be:

$$Pr(error) < (0.5)Pr(A \leq k' \leq k_1) + Pr(k_1 \leq k' \leq k_{mid}) + (0.5)Pr(k_2 \leq k' \leq B)$$

We now let $x = \frac{k_1 - A}{B-A}$, and $y = \frac{B-k_2}{B-A}$. Then:

$$\begin{aligned} Pr(error) &< (0.5)x + 0.5 - x + (0.5)y \text{ Let } x > y, \text{ since } k^* \text{ is to} \\ \implies Pr(error) &< (0.5)x + 0.5 - x + (0.5)x \\ \implies Pr(error) &< 0.5 \end{aligned}$$

Similarly, if k^* were to the right of the current interval, $Pr(error) < 0.5$.

Combining these assertions, we see that the Environment provides the correct response with $p > 0.5$, if $x > y$. Hence the result! ■

Lemma 2: Using the L_{RI} scheme with a parameter θ which is arbitrarily close to zero, the following is true:

- If λ^* is left of Δ_k^j, then Pr(O^j=Left) $\rightarrow 1$
- If λ^* is right of Δ_k^j, then Pr(O^j=Right) $\rightarrow 1$
- If λ^* is inside Δ_k^j, then Pr(O^j=Left, Right or Center) $\rightarrow 1$.

Proof: The proof is identical to the one found in [28] and is thus omitted. ■

Lemma 3: Using the L_{RI} scheme with a parameter θ which is arbitrarily close to zero, the following is true:

- If (O^j=Left) then Pr(λ^* being left or inside of Δ_k^j) $\rightarrow 1$
- If (O^j=Right) then Pr(λ^* being right or inside of Δ_k^j) $\rightarrow 1$
- If (O^j=Inside) then Pr(λ^* being Inside of Δ_k^j) $\rightarrow 1$.

Proof: The proof is identical to the one found in [28] and is thus omitted. ■

Lemma 4: If the algorithm uses the L_{RI} same scheme at all levels of the recursion and a parameter θ arbitrarily close

[9]The reader must remember that $F(k,p)$ is monotonically *decreasing* with respect to p.

to zero, the LA converge at every level with a probability as close to unity as desired.

Proof: The proof follows from the above, and is due to the ε-optimality of the LA used in every level of the scheme. ■

Theorem 1: The set of decision rules given in Table II is complete.

Proof: Again, the proof is identical to the one found in [28] and is thus omitted. ■

The final theorem about the entire scheme follows.

Theorem 2: The unknown λ^* is always contained in the subinterval encountered in the subsequent invocation of the algorithm and thus the algorithm finally converges to a value arbitrarily close to λ^*, assuming the following conditions:

- The algorithm uses the L_{RI} same scheme at all levels of the recursion.
- The parameter θ is arbitrarily close to zero.
- The parameter N_∞ is sufficiently large.

Proof: The proof of this theorem, which is true for every level of the LA, is identical to the one found in [28] and is thus omitted. This concludes the proofs of the algorithms. ■

V. Testing and Results: 2-dimensional Environments

In this section, we present the results for the 2-dimensional DE using the ATS. We show that this method of estimation works for three different DEFs where, as mentioned, the first two DEFs, Eq. (6) and Eq. (7) each contained only a single parameter that must be determined, either k or p respectively. The last DEF, Eq. (8), 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 II-B, in Eq. (2), Eq. (3) and Eq. (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 Eq. (14):

$$EP = \frac{1}{n}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{|A(P_i,P_j) - \pi(P_i,P_j|\Lambda)|}{A(P_i,P_j)}. \qquad (14)$$

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.

A. Results for the Noiseless Data

a) Experimental Setup : The first type of data was noiseless. This set of 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 were 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 sets of noiseless data. 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, where the accuracy was reported 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. We emphaszie that the algorithm was not overly sensitive to these values because the decisions of the LAs were only used to update the search interval and not the final parameter. As a result, only requiring a "belief" of 90% helped the convergence times, since the L_{RI} LA converge slowly as the probability approach the absorbing barriers, zero and unity.

b) Weighted Euclidean DEF : In this case, we examined the Weighted Euclidean DEF (Eq. (7)), 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 observed that the ATS always succeeded in finding the optimal parameter for a DEF that contained only a single parameter in a noiseless environment. As a result, all the errors were zero with a standard deviation of zero over 100 executions of the search. Figure 2 shows a pictorial representation of the ATS in this environment.

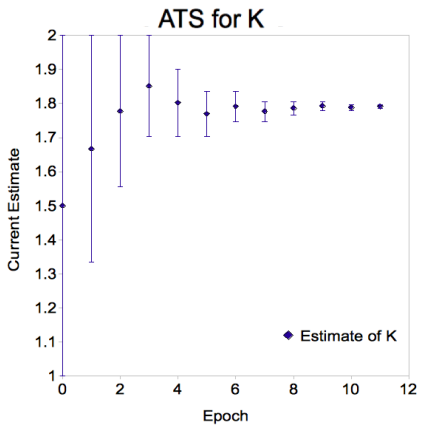

Fig. 2: The progress of the ATS scheme for a typical noiseless environment for the DEF given by (Eq. (7)). Here, the blue lines represent the current search interval, and the blue diamonds represent the current estimate of k.

c) L^p DEF: We now consider the L_p DEF from Eq. (6). This DEF also has a single parameter, and consequently, the typical ATS is very similar to the one shown in Figure 2. The result was identical as for the Weighted Euclidian DEF, and is omitted in the interest of space. All the errors were exactly equal to zero, because the ATS always converged to the actual value of p.

d) Weighted L^p DEF:: The final DEF that we studied was the Weighted L^p DEF from Eq. (8). This DEF has two

parameters, k and p, and as a result, the ATS had to simultaneously search for both the parameters in the joint space. Although this was easily done, it did affect the minimum resolution. This is because the reduction of the search interval for each parameter was depended on the resolution of the search interval of the other parameter. In the example for which the results are submitted, the 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 III 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.

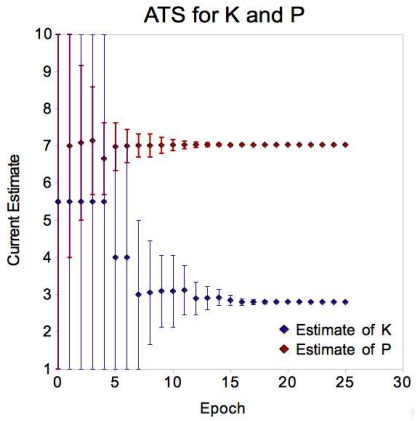

Fig. 3: The progress of the ATS scheme for a typical noiseless environment for the DEF given by (Eq. (8)). The blue and red lines represent the current search intervals for the respective parameters, and the diamonds represent the current estimates of the parameters.

Figure 3 shows a pictorial representation of an ATS for multiple parameters in a noiseless environment.

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 IV, where we report the average errors and standard deviation of 100 runs of the ATS. It should be noted the "Actual" values were always contained in the final search intervals. We also mention that if the resolution was set to be too small, the ATS would not be able to reduce the search interval to the required size causing it to continue, until the process was manually terminated. Thus, while we had set the resolution to be the convergence requirement, one could alternately have used the number of epochs as the convergence requirement. Under these circumstances the resolution would vary from run to run.

B. Results for the Noisy Data

a) 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 that 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 as below:

Epoch	k's Current Search Interval	p's Current Search Interval
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 III: Example run of the ATS with the *Weighted* L^p DEF on the noiseless data.

$$Noise = \frac{TrueDistance}{10} * X \tag{15}$$

where, X is a random Gaussian 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 that 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, λ) to the $\lambda+\varepsilon$ and $\lambda-\varepsilon$, and moved to the value that minimized the GoF function. This process was repeated until the current value λ minimized 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 the 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 deviations. 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

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

100 runs. Observe that for each of these 100 executions of the ATS, the noise and "Actual values" change.

b) Weighted Euclidian DEF:: The errors of the "Benchmark" and "Estimated" values shown in Table V 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.

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 V: Results for the typical run for the case when the dataset was noisy and it used the Weighted DEF.

Table VI shows the average errors of 100 ATSs 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 the "Benchmark" and "Estimated" values for both the RAD and the EP errors. These results clearly demonstrate the success of the ATS for DE in a noisy environment.

c) L^p DEF:: We again ran the ATS 100 times on the L_p DEF which incorporated noisy data. The results are shown in Table VII. These results are similar to those obtained for 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%.

d) Weighted L^p DEF:: The ATS for the Weighted L^p DEF was performed on the noisy data. Figure 4 shows a pictorial representation of an ATS for multiple parameters in a noisy environment.

The average testing errors for 100 runs of the ATS using the Weighted L^p DEF is shown in Table VIII. The errors for this norm were 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 [1, 5, 16, 18]. The errors for the "Estimated" values were are very close to the errors for the "Benchmark" values, where the RAD and EP errors were within 0.2% of each other.

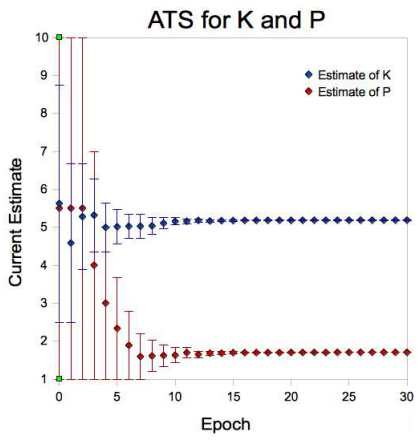

Fig. 4: The progress of the ATS scheme for a typical noiseless environment for the Weighted L^p DEF. The blue and red lines represent the current search interval for the respective parameters, and the diamonds represent the current estimate of the parameters.

C. Results for the Real-World Data Sets

a) Experimental Setup:: The final test for the ATS was done on real-world data sets, since the "proof of the pudding is, indeed, in the eating". This data consisted of three sets, which in turn involved 29, 97, and 561 cities each. The data sets involving 29 and 561 cities were obtained from the MP-TESTDATA (the TSPLIB Symmetric Traveling Salesman Problem Instances) [32]. 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 involving 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 [3, 26], and the distances were calculated using Google maps [13].

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.

b) Weighted Euclidian DEF:: Table IX 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

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 VI: Results for 100 runs of the ATS with the Weighted Euclidian DEF on noisy data sets.

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

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

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 IX: Results for 100 runs of the ATS with the Weighted Euclidian DEF on the real-worlds data sets.

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 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.

c) L^p DEF: Table X shows the results for 100 runs of the ATS search using the L^p norm on the real-world 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	
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 X: Results for 100 runs of the ATS with the L^p DEF on the real-world data sets.

Overall the errors were extremely large, over 300% error for the RAD 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 converged 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.

d) 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 XI. While the ATS and the hill-climbing search performed 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 had 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 was 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.

D. Discussion

The ATS *always* converged close to the "actual" values for all three DEFs when interacting with noiseless data sets. The errors were either exactly zero or smaller than 10^{-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 DEFs 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. Overall the ATS found values that were competitive with the standard hill clime method.

The most significant contribution of this work was that the ATS did not require the use of GoF functions, which we believe is a pioneering and novel contribution.

VI. Conclusions

A. The ATS

In this paper, we considered the Distance Estimation (DE) problem that has been studied for almost four decades. It involves estimating the real-life distances between points in the Cartesian plain or in a geographic region. The input to these DE problems are, typically, the start and end locations in the form of the x and y co-ordinates of the locations in the Cartesian plain, or the latitude and longitude in the geographic region. Our solution departs from the legacy methods in that we depart from the use of so-called "Goodness-of-Fit" (GoF) functions. Rather, we have used the field of Learning Automata (LA) and in particular, the Adaptive Teriary Search (ATS) used to solve the Stochastic Point Location (SPL) problem. This paper has made some major contributions. Firstly, it extended the ATS application to the DE problem. In this regard, 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.

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

The second contribution that we 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. 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.

B. 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 the 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, while using DEFs to *compare* the qualities of the parameters.

REFERENCES

[1] 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.

[2] J. L. Bentley and A. C-C. Yao. An almost optimal algorithm for unbounded searching. *Information Processing Letters*, pp. 82–87, August 1976.

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

[4] 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.

[5] 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.

[6] J. Brimberg and J. H. Walker, *Estimation of Travel Distance*, John Wiley & Sons, Inc., 2004.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] S. Glimsdal and O.-C. Granmo, "Thompson Sampling Guided Stochastic Searching on the Line for Adversarial Learning," in *Artificial Intelligence Applications and Innovations: 11th IFIP WG 12.5 International Conference, AIAI 2015, Bayonne, France, September 14-17, 2015*, pp. 307–317.

[12] S. Glimsdal and O.-C. Granmo, "Thompson Sampling Guided Stochastic Searching on the Line for Non-stationary Adversarial Learning," in *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 687–692.

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

[14] O.-C. Granmo, "Solving two-armed Bernoulli bandit problems using a Bayesian learning automaton," *International Journal of Intelligent Computing and Cybernetics*, vol. 3, no. 2, pp. 207–234, 2010.

[15] D. Huang and W. Jiang. A General CPL-AdS methodology for fixing dynamic parameters in dual environments. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-42:1489– 1500, 2012.

[16] 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.

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

[18] 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.

[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] K. Najim and A. S. Poznyak. *Learning automata: theory and applications*. Elsevier, 2014.

[21] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Courier Corporation, 2012.

[22] A. G.N. Novaes, E.T. Bez, P.J. Burin, and D.P. Aragao, "Dynamic milk-run oem operations in over-congested traffic conditions," *Computers & Industrial Engineering*, vol. 88, no. 12, pp. 326–340, 2015.

[23] M. S. Obaidat, S. Misra, and G. I. Papadimitriou. Guest editorial: Adaptive and learning systems. *IEEE Trans. Sys. Man Cyber. Part B*, vol. 40, pp. 2–5, February 2010.

[24] B. J. Oommen. Stochastic searching on the line and its applications to parameter learning in nonlinear optimization. *IEEE Trans. Sys. Man Cyber.*, vol. 27, pp. 733–739, 1997.

[25] B. J. Oommen and D. Calitoiu. Modeling and simulating a disease outbreak by learning a contagion parameter-based model. *Proc. of Spring Simulation Multiconference*, 547– 555, 2008.

[26] J. Oommen, I. K. Altinel, and N. Aras, "Discrete vector quantization for arbitrary distance function estimation," *IEEE Trans. Sys. Man Cyber.*, vol. 28, no. 4, pp. 496 – 510, 1998.

[27] B. J. Oommen, O.-C. Granmo, and Z. Liang. A novel multidimensional scaling technique for mapping word-of-mouth discussions. *Studies in Comput. Intel.*, Vol. 214:317– 3224, 2009.

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

[29] B. J. Oommen, G. Raghunath, and B. Kuipers, "Parameter learning from stochastic teachers and stochastic compulsive liars," *IEEE Trans. Sys. Man Cyber.*, vol. 36, no. 4, pp. 820 – 834, 2006.

[30] F. A. Ortega and J. A. Mesa, "A methodology for modelling travel distances by bias estimation," *Sociedad de Estadistica e lnvestigaci6n Operativa*, vol. 6, no. 2, pp. 287–311, 1998.

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

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

[33] T. Tao, H. Ge, G. Cai, and S. Li. Adaptive step searching for solving stochastic point location problem. *Intern. Conf. Intel. Comput. Theo*, ICICT-13:192– 198, 2013.

[34] M. A. L. Thathachar and P. S. Sastry, "Varieties of learning automata: An overview," *IEEE Trans. Sys. Man Cyber.*, vol. 32, no. 6, pp. 711–722, 2002.

[35] 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.

[36] J. H. Walker, *An Empirical Study of round and block norms for modelling actual distances*, Ph.D. thesis, McMaster University, 1991.

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

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

[39] A. Yazidi and B. J. Oommen. Discretized Weak Estimators Based on the Principles of the Stochastic Search on the Line Problem. *IEEE Trans. Sys. Man Cyber.*, vol. 46 pp. 2732– 2744, 2016.

[40] A. Yazidi, O.-C. Granmo, B. J. Oommen, and M. Goodwin. A novel strategy for solving the stochastic point location problem using a hierarchical searching scheme. *IEEE Trans. Sys. Man Cyber.*, vol. 44, pp. 2202– 2220, 2014.

[41] A. Yazidi and B. J. Oommen A novel technique for stochastic root-finding: Enhancing the search with adaptive d-ary search. *Information Sciences.*, 31 (393):108–129, 2017.

[42] J. Zhang, Y. Wang, C. Wang, and M. Zhou. Symmetrical hierarchical stochastic searching on the line in informative and deceptive environments. *IEEE Trans. Sys. Man Cyber.*, vol. 47, pp. 626– 635, 2017.

[43] J. Zhang, S. Lu, D. Zang, and M. Zhou. Integrating particle swarm optimization with stochastic point location method in noisy environment. *Proc. of IEEE Int. Conf. Syst., Man, Cybern.*, 145– 150, 2016.