Novel Distance Estimation Methods Using “Stochastic Learning on the Line” Strategies

JESSICA HAVELOCK1, B. JOHN OOMMEN2, (Fellow, IEEE), AND OLE-CHRISTOFFER GRANMO3

1School of Computer Science, Carleton University, Ottawa, ON K1S 5B6, Canada
2Department of Information and Communication Technology, University of Agder, 4630 Kristiansand, Norway
3Centre for Artificial Intelligence Research, Department of Information and Communication Technology, University of Agder, 4630 Kristiansand, Norway

Corresponding author: B. John Oommen (oommen@scs.carleton.ca)

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. This problem has, typically, been tackled by utilizing parametric functions called the “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 data sets 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.

INDEX TERMS 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 synony-
mously 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,\textsuperscript{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 \textit{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 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 \textit{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 esti-
mation 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.\textsuperscript{3}

\section{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 \textit{Functions} (DEFs). These functions
can take on any form, but the ideal ones are those that are
simultaneously good estimators, and that are also character-
ized by low computational requirements. Love and Morris
first introduced the concept of using simple parametric func-
tions that employ the \textit{x} and \textit{y} co-ordinates for approximat-
ing 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 \textit{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]–[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 conse-
quence of applying more complex principles. For example,
some researchers have proposed using a \textit{weighted} linear com-
bination of two more primitive DEFs [7], [30]. Another novel
idea for a DEF is to use a \textit{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 \textit{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 dis-
tances 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

\textsuperscript{2}Two common vehicle scheduling software packages, ROADNET and
TRUCKSTOP\textsuperscript{2}S 2, use DE methods when determining the distances between
the suppliers and the customers [8].

\textsuperscript{3}In the interest of being focused, we shall concentrate here on the tradi-
tional problem of using DE for estimating “road” distances.
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 parameterization 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 information is actually performed 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 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 \(\pi(P_1, P_2|\Lambda) : R^d \times R^d \rightarrow R\), where \(P_1 = (x_1, x_2, ..., x_d)\) and \(P_2 = (y_1, y_2, ..., y_d)\) are points in \(R^d\), and \(\Lambda\) is a set of parameters whose values characterize \(\pi\), and which must be learnt using a set of training points with known true inter-point distances.

\(^4\)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.
The set of parameters, \( \lambda \), 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]–[18], [26], [30], [35]. Although like the AD, the NAD uses the Relative Absolute-value Difference (RAD) function given in Eq. (2) is often recommended [4]:

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

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]–[18], [26], [30], and [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]:

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

The next and third most-common GoF function is the Normalized Absolute-value Difference (NAD), given by the sum of the absolute values of the differences between the estimated distances and the actual distances, normalized by the sum of the actual distances. Several GoF functions have been consistently utilized in the literature pertaining to the field of DE. 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:

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

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]–[18], [26], [30], and [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]:

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

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:

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

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]–[18], [26], [35].

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

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, \( \lambda \). This allows the user to employ a gradient decent search scheme to determine the “best” parameter \( \lambda \) [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 of \( L^p \) norms, traditionally used for computing distances:

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

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 in Table 1.

---

**TABLE 1. List of DEFs related to various \( L^p \) norms and their corresponding parameters.**

<table>
<thead>
<tr>
<th>DEF</th>
<th>Expression</th>
<th>Name of DEF</th>
<th>Parameters</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_1 )</td>
<td>( \left( \sum_{i=1}^{n}</td>
<td>x_i - x_0</td>
<td>^p \right)^{1/p} )</td>
<td>( L^p )</td>
</tr>
<tr>
<td>( P_2 )</td>
<td>( k \left( \sum_{i=1}^{n}</td>
<td>x_i - x_0</td>
<td>^p \right)^{1/2} )</td>
<td>Weighted Euclidean</td>
</tr>
<tr>
<td>( P_3 )</td>
<td>( k \left( \sum_{i=1}^{n}</td>
<td>x_i - x_0</td>
<td>^p \right)^{1/p} )</td>
<td>Weighted ( L^p )</td>
</tr>
<tr>
<td>( P_4 )</td>
<td>( k \left( \sum_{i=1}^{n}</td>
<td>x_i - x_0</td>
<td>^p \right)^{1/s} )</td>
<td>( kL_p )</td>
</tr>
<tr>
<td>( P_5 )</td>
<td>( \left( \sum_{i=1}^{n} (k +</td>
<td>x_i - x_0</td>
<td>^p \right)^{1/p} )</td>
<td>Generalized Weighted ( L^p )</td>
</tr>
<tr>
<td>( P_6 )</td>
<td>( \left(</td>
<td>x_1 - x_2</td>
<td>^p / M (x_1 - x_2) \right)^{1/2} )</td>
<td>Elliptical DEF</td>
</tr>
</tbody>
</table>

---

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.

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

---
In particular, Eq. (6) performed well 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) out-performed 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 $d_1$” (where $d_1$ 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 out-performed by the $k_{L_p}$ 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 1, 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 1, 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 geographical area in question. Of course, as explained later, a multi-regional approach for areas with different geographical features, is always superior.

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), $\lambda$. We assume that there is an optimal choice for $\lambda$ - 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

---

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

8The 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.
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 $\lambda^*$ and the probability of receiving correct feedback were simultaneously learned, allowing TS-SPL to operate in both deceptive and non-stationary Environments.

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 $\lambda^*$, 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 $\lambda^*$ is to the right or to the left of the currently chosen $\lambda(n)$.

To determine $\lambda^*$ within the resolution of accuracy, the original search interval is divided into three equal and disjoint subintervals, $\Delta^i$, where $i = 1...3$. The subintervals are searched using a two-action LA. The LA returns the $\lambda(n)$, the estimated position of $\lambda^*$ from that subinterval, $\lambda^* \in \{\text{Left}, \text{Right}, \text{Inside}\}$. From these outputs, a new search interval is obtained which is based on the decision table given in Table 2. 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 $\lambda^*$. The ATS algorithm can be seen in Algorithm 1.

### Algorithm 1: ATS Algorithm

**Input:** The Resolution, $\rho$

**Output:** Estimate of $\lambda^*$

**Method:**

1. repeat
2. $(\Delta^1, \Delta^2, \Delta^3) \leftarrow \text{Getpartitions}(\Delta)$
3. for $j \leftarrow 1$ to $3$ do
4. Get position of $\lambda^*$ from $LAI_j$
5. end for
6. $\Delta \leftarrow \text{Get new search interval from Table 2}$
7. until Size of Interval $< \rho$
8. $\lambda^* \leftarrow \text{Midpoint}(\Delta)$
9. End Algorithm

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

<table>
<thead>
<tr>
<th><strong>Table 2. Decision table.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
</tr>
<tr>
<td><strong>Inside</strong></td>
</tr>
<tr>
<td><strong>Left</strong></td>
</tr>
<tr>
<td><strong>Right</strong></td>
</tr>
<tr>
<td><strong>Right</strong></td>
</tr>
<tr>
<td><strong>Right</strong></td>
</tr>
</tbody>
</table>

- **Algorithm 2:**

- **Input:** $k, p$

- **Output:** $k^*, p^*$

- **Method:**

1. repeat
2. $(\Delta^1, \Delta^2, \Delta^3) \leftarrow \text{Getpartitions}(\Delta)$
3. for $j \leftarrow 1$ to $3$ do
4. Get position of $k$ and $p$ from $LAI_j$
5. end for
6. $\Delta \leftarrow \text{Get new search intervals from Table 2}$
7. until Size of Interval $< \rho$
8. $k^*, p^* \leftarrow \text{Midpoint}(\Delta)$
9. End Algorithm

- **End Algorithm**
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 its Environment for a response. The LA uses a Linear Reward Inaction (LRi) 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 $N_{\infty}$, 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 $1 - \epsilon$, 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 $1 - \epsilon$, the LA’s final decision is “Left”. If neither of these cases emerge, the LA does not have a belief greater than $1 - \epsilon$ 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 $\theta_R$ ($0 \ll \theta_R < 1$) 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 in Eq. (8) which can be simplified into two equations, Eq. (12) and Eq. (13) as below:

$$ F(k, p) = k \cdot F_1(X_1, X_2, p), \quad \text{and where,} \quad F_1(X_1, X_2, p) = \left( \sum_{i=1}^{d} |x_i - x_2|^p \right)^{1/p}. \quad (12) $$

$$ F_1(X_1, X_2, p) = \left( \sum_{i=1}^{d} |x_i - x_2|^p \right)^{1/p}. \quad (13) $$

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 \( p \).

**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 \( \in \{ \text{Agree}, \text{Disagree} \} \) as far as the parameter \( k \) is concerned.

**Method:**

1. if ((Choice == Left) \( \land \) \( F(k_r, p_{Max}) \geq \) TrueDistance)) then
   2. return Agree
3. else if ((Choice == Right) \( \land \) \( F(k_r, p_{Min}) \leq \) TrueDistance)) then
   4. return Agree
5. else
6. return Disagree
7. end if

**End Algorithm**

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.

**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 \( \in \{ \text{Agree}, \text{Disagree} \} \) as far as the parameter \( p \) is concerned.

**Method:**

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

**End Algorithm**

We now consider the \( \epsilon \)-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 \( \epsilon \)-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 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 \( T_{dist} \) be the true distance provided by the points under consideration in the training data.

We now partition \( \Delta_k \) into three mutually exclusive and exhaustive parts as \( \Delta_k = P_1 \cup P_2 \cup P_3 \), where:

\[
P_1 = [A, k_1] - \text{ such that } \forall k \in [A, k_1], F(k, p_{Max}) < T_{dist} \text{ or } F(k, p_{Min}) > T_{dist}
\]

\[
P_2 = [k_1, k_2] - \text{ such that } \forall k \in [k_1, k_2], F(k, p_{Max}) > T_{dist} \text{ and } F(k, p_{Min}) < T_{dist}
\]

\[
P_3 = [k_2, B] - \text{ such that } \forall k \in [k_2, B], F(k, p_{Max}) < T_{dist} \text{ or } F(k, p_{Min}) > T_{dist}.
\]

These partitions fully divide the search interval due to the monotonicity properties of \( F(\cdot, \cdot) \). Consequently:

\[
Pr(error) = Pr(error|k' \in P_1)Pr(k' \in P_1) + Pr(error|k' \in P_2)Pr(k' \in P_2) + Pr(error|k' \in P_3)Pr(k' \in P_3)
\]

By considering the \( P_2 \) term more clearly, we see that:

\[
Pr(error|k' \in P_2)Pr(k_1 \leq k' \leq k_2) < Pr(error|k' \in P_2, k' < k_{mid})Pr(k_1 \leq k' \leq k_{mid}) + Pr(error|k' \in P_2, k' < k_{mid})Pr(k_{mid} \leq k' \leq k_2)
\]

\[
= (1)Pr(k_1 \leq k' \leq k_{mid}) + (0)Pr(k_{mid} \leq k' \leq k_2)
\]

\[
= Pr(k_1 \leq k' \leq k_{mid}).
\]

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:

\[
Pr(error) < (0.5)x + 0.5 - x + (0.5)y \text{ Let } x > y, \text{ since } k^* \text{ is to the left of the interval}
\]

\[
\Rightarrow Pr(error) < (0.5)x + 0.5 - x + (0.5)x
\]

\[
\Rightarrow Pr(error) < 0.5
\]

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 \( \theta \) which is arbitrarily close to zero, the following is true:

- If \( \lambda^* \) is left of \( \Delta_k \), then \( Pr(O' = \text{Left}) \to 1 \)
- If \( \lambda^* \) is right of \( \Delta_k \), then \( Pr(O' = \text{Right}) \to 1 \)
- If \( \lambda^* \) is inside \( \Delta_k \), then \( Pr(O' = \text{Left, Right or Center}) \to 1 \).

**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 \( \theta \) which is arbitrarily close to zero, the following is true:

- If \( (O' = \text{Left}) \) then \( Pr(\lambda^* \text{ being left or inside of } \Delta_k) \to 1 \)
- If \( (O' = \text{Right}) \) then \( Pr(\lambda^* \text{ being right or inside of } \Delta_k) \to 1 \)
- If \( (O' = \text{Inside}) \) then \( Pr(\lambda^* \text{ being Inside of } \Delta_k) \to 1 \).

**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 \( \theta \) arbitrarily close 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 \( \varepsilon \)-optimality of the LA used in every level of the scheme.

**Theorem 1:** The set of decision rules given in Table 2 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 \( \lambda^* \) 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 \( \lambda^* \), assuming the following conditions:

- The algorithm uses the \( L_{RI} \) same scheme at all levels of the recursion.
- The parameter \( \theta \) is arbitrarily close to zero.
- The parameter \( N_\infty \) 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):

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

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
1) 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 ($\theta_R$) was set to 0.02. We used an $\epsilon$ value of 0.1 and $N_\infty = 2500$. These values were chosen through preliminary testing and lie within the generally accepted range of values use for these types of LA. We emphasize 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 $L_{R_\infty}$ LA converge slowly as the probability approach the absorbing barriers, zero and unity.

2) 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.

3) $L^p$ 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 $p$.

4) 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 0.000001; however, for this ATS the resolutions for $k$ and $p$ were set to 0.00001 and 0.001 respectively. Table 3 shows an example of a typical ATS for the Weighted $L^p$ DEF in a noiseless environment. The errors for this ATS were...
TABLE 3. Example run of the ATS with the Weighted $L^p$ DEF on the noiseless data.

<table>
<thead>
<tr>
<th>Epoch</th>
<th>$k$’s Current Search Interval</th>
<th>$p$’s Current Search Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>[6.0396743, 6.040656]</td>
<td>[6.630701, 6.703157]</td>
</tr>
<tr>
<td>23</td>
<td>[6.0396743, 6.040329]</td>
<td>[6.642777, 6.690949]</td>
</tr>
<tr>
<td>24</td>
<td>[6.0397835, 6.040238]</td>
<td>[6.651163, 6.6861053]</td>
</tr>
<tr>
<td>25</td>
<td>[6.0397835, 6.0408863]</td>
<td>[6.6569867, 6.681252]</td>
</tr>
<tr>
<td>26</td>
<td>[6.0398846, 6.0408863]</td>
<td>[6.661031, 6.6778817]</td>
</tr>
<tr>
<td>27</td>
<td>[6.039952, 6.0408863]</td>
<td>[6.6638393, 6.6754414]</td>
</tr>
<tr>
<td>28</td>
<td>[6.039952, 6.0400414]</td>
<td>[6.6667896, 6.673916]</td>
</tr>
<tr>
<td>29</td>
<td>[6.039982, 6.0400214]</td>
<td>[6.6671544, 6.672787]</td>
</tr>
<tr>
<td>30</td>
<td>[6.039982, 6.0400214]</td>
<td>[6.668084, 6.6720033]</td>
</tr>
<tr>
<td>31</td>
<td>[6.039982, 6.0400048]</td>
<td>[6.6687374, 6.671459]</td>
</tr>
<tr>
<td>32</td>
<td>[6.0399004, 6.0400068]</td>
<td>[6.669191, 6.670181]</td>
</tr>
<tr>
<td>33</td>
<td>[6.039966, 6.0400048]</td>
<td>[6.669506, 6.6701813]</td>
</tr>
<tr>
<td>34</td>
<td>[6.039996, 6.0400043]</td>
<td>[6.669506, 6.670381]</td>
</tr>
</tbody>
</table>

B. RESULTS FOR THE NOISY DATA

1) 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:

$$\text{Noise} = \frac{\text{TrueDistance}}{10} \times X$$

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, $\lambda$) to the $\lambda + \epsilon$ and $\lambda - \epsilon$, and moved to the value that minimized the GoF function. This process was repeated until the current value $\lambda$ 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 100 runs. Observe that for each of these 100 executions of the ATS, the noise and “Actual values” change.

2) WEIGHTED EUCLIDIAN DEF

The errors of the “Benchmark” and “Estimated” values shown in Table 5 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...
TABLE 4. Results for 100 runs of the ATS with the Weighted $L^p$ DEF on the noiseless data sets.

<table>
<thead>
<tr>
<th>Data Set Type</th>
<th>Average Error</th>
<th>Standard Deviation</th>
<th>Average Error</th>
<th>Standard Deviation</th>
<th>Average Error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>2.06 x 10^-7</td>
<td>4.69 x 10^-7</td>
<td>1.04 x 10^-7</td>
<td>1.53 x 10^-7</td>
<td>4.17 x 10^-7</td>
<td>8.77 x 10^-7</td>
</tr>
<tr>
<td>NAD</td>
<td>6.84 x 10^-7</td>
<td>9.41 x 10^-7</td>
<td>6.94 x 10^-7</td>
<td>3.44 x 10^-7</td>
<td>3.23 x 10^-7</td>
<td>3.62 x 10^-7</td>
</tr>
<tr>
<td>RAD</td>
<td>1.66 x 10^-7</td>
<td>2.54 x 10^-7</td>
<td>3.27 x 10^-7</td>
<td>3.34 x 10^-7</td>
<td>3.58 x 10^-7</td>
<td>3.93 x 10^-7</td>
</tr>
<tr>
<td>EP</td>
<td>1.69 x 10^-7</td>
<td>2.59 x 10^-7</td>
<td>3.30 x 10^-7</td>
<td>3.32 x 10^-7</td>
<td>3.58 x 10^-7</td>
<td>3.93 x 10^-7</td>
</tr>
</tbody>
</table>

TABLE 5. Results for the typical run for the case when the dataset was noisy and it used the Weighted DEF.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>For Estimated Value: $k = 1.3528224$</th>
<th>For Benchmark Value: $k = 1.346991$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td>427.73</td>
<td>425.52</td>
</tr>
<tr>
<td>NAD</td>
<td>105.63</td>
<td>105.36</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0581</td>
<td>0.0582</td>
</tr>
<tr>
<td>EP</td>
<td>0.0597</td>
<td>0.0595</td>
</tr>
</tbody>
</table>

TABLE 6. Results for 100 runs of the ATS with the Weighted Euclidian DEF on noisy data sets.

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>N=29</th>
<th>N=75</th>
<th>N=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type</td>
<td>Average Error</td>
<td>Standard Deviation</td>
<td>Average Error</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SD</td>
<td>239.31</td>
<td>82.55</td>
<td>70.73</td>
</tr>
<tr>
<td>NAD</td>
<td>1.99</td>
<td>0.4636</td>
<td>14.68</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0546</td>
<td>0.0131</td>
<td>0.0572</td>
</tr>
<tr>
<td>EP</td>
<td>0.0553</td>
<td>0.0129</td>
<td>0.0580</td>
</tr>
</tbody>
</table>

TABLE 7. Results for 100 runs of the ATS with the $L^p$ DEF on the noisy data sets.

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>N=29</th>
<th>N=75</th>
<th>N=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type</td>
<td>Average Error</td>
<td>Standard Deviation</td>
<td>Average Error</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SD</td>
<td>370.38</td>
<td>89.67</td>
<td>94.39</td>
</tr>
<tr>
<td>NAD</td>
<td>2.68</td>
<td>0.41</td>
<td>19.23</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0724</td>
<td>0.0114</td>
<td>0.0774</td>
</tr>
<tr>
<td>EP</td>
<td>0.0744</td>
<td>0.0113</td>
<td>0.0759</td>
</tr>
</tbody>
</table>

TABLE 8. Results for 100 runs of the ATS with the Weighted $L^p$ DEF on noisy data sets.

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>N=29</th>
<th>N=75</th>
<th>N=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type</td>
<td>Average Error</td>
<td>Standard Deviation</td>
<td>Average Error</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>SD</td>
<td>327.54</td>
<td>88.62</td>
<td>94.01</td>
</tr>
<tr>
<td>NAD</td>
<td>2.66</td>
<td>0.41</td>
<td>19.20</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0720</td>
<td>0.0109</td>
<td>0.0744</td>
</tr>
<tr>
<td>EP</td>
<td>0.0740</td>
<td>0.0113</td>
<td>0.0759</td>
</tr>
</tbody>
</table>

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.

Table 6 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.

3) $L^p$ DEF

We again ran the ATS 100 times on the $L^p$ DEF which incorporated noisy data. The results are shown in Table 7. 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%.

4) 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 8. 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 values for both the RAD and the EP errors. These results
### TABLE 8. Results for 100 runs of the ATS with the Weighted $L^p$ DEF on the noisy data sets.

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>N=29</th>
<th>N=75</th>
<th>N=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type</td>
<td>Average Error</td>
<td>Standard Deviation</td>
<td>Estimated</td>
</tr>
<tr>
<td>SD</td>
<td>99.38</td>
<td>85.67</td>
<td>28.90</td>
</tr>
<tr>
<td>NAD</td>
<td>0.75</td>
<td>0.62</td>
<td>5.60</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0208</td>
<td>0.0171</td>
<td>0.0218</td>
</tr>
<tr>
<td>EP</td>
<td>0.0208</td>
<td>0.0172</td>
<td>0.0221</td>
</tr>
</tbody>
</table>

### TABLE 9. Results for 100 runs of the ATS with the Weighted Euclidian DEF on the real-worlds data sets.

<table>
<thead>
<tr>
<th>Data Set Size</th>
<th>N=29</th>
<th>N=75</th>
<th>N=1561</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Type</td>
<td>Average Error</td>
<td>Standard Deviation</td>
<td>Estimated</td>
</tr>
<tr>
<td>Value</td>
<td>0.2230</td>
<td>1.6259</td>
<td>1.3808</td>
</tr>
<tr>
<td>SD</td>
<td>23.66</td>
<td>0.18</td>
<td>20853.85</td>
</tr>
<tr>
<td>NAD</td>
<td>1.82</td>
<td>0.01</td>
<td>83.62</td>
</tr>
<tr>
<td>RAD</td>
<td>0.0406</td>
<td>0.0001</td>
<td>0.1268</td>
</tr>
<tr>
<td>EP</td>
<td>0.0507</td>
<td>0.0002</td>
<td>0.2060</td>
</tr>
</tbody>
</table>

### FIGURE 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.

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.

### C. RESULTS FOR THE REAL-WORLD DATA SETS

#### 1) 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].

#### 2) WEIGHTED EUCLIDIAN DEF

Table 9 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 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.

3) **LP DEF**
Table 10 shows the results for 100 runs of the ATS search using the LP norm on the real-world data. Overall the errors were extremely large, over 300% error for the RAD and EP errors for the data set size 29 and 561. This can be attributed to two main reasons; first the LP 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 LP 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.

4) **WEIGHTED LP DEF**
When the ATS is used in conjunction with the Weighted LP DEF, the ATS out-performed the hill-climbing search, as shown in results in Table 11. 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 LP 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 LP DEF and the Weighted Euclidean DEF. Finally, the ATS using the Weighted LP 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 Lp DEF. These large errors are due to the predicting abilities of the Lp 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-climbing 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 Tertiary 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.

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.

ACKNOWLEDGMENT

This paper was presented in part at the Proc. of IEA/AIE’18, the 2018 International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Montreal, Canada, June 2018. The author B. J. Oommen gratefully acknowledges the partial support of NSERC, the Natural Sciences and Engineering Council of Canada.

REFERENCES


---

JESSICA HAVELOCK was born in Ottawa, ON, Canada, in 1986. She received the B.Sc. degree in mathematics and engineering–computers and communications from Queen’s University, Kingston, ON, Canada, in 2009, and the M.Sc. degree from the School of Computer Science, Carleton University, Ottawa, ON, Canada, in 2011. During her time at Queen’s University, she focused in statistics and stochastic control theory. She then went on to study stochastic learning algorithms at Carleton University. She is currently a Solution Architect at the private sector.

B. JOHN OOMMEN (F’03) was born in Coimoor, India, in 1953. He received the B.Tech. degree from IIT Madras, India, in 1975, the M.E. degree from the Indian Institute of Science, Bangalore, India, in 1977, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, USA, in 1979 and 1982, respectively. He joined the School of Computer Science, Carleton University, Ottawa, ON, Canada, from 1981 to 1982, where he is currently a Full Professor. Since 2006, he has been a recipient of the honorary rank of Chancellor’s Professor, which is a lifetime award from Carleton University. His research interests include automata learning, adaptive data structures, statistical and syntactic pattern recognition, stochastic algorithms and partitioning algorithms. He has authored over 465 refereed journal and conference publications. He is a Fellow of the IAPR. He has also served on the Editorial Board of the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS AND PATTERN RECOGNITION.

OLE-CRISTOFFER GRANMO was born in Porsgrunn, Norway. He received the M.Sc. and Ph.D. degrees from the University of Oslo, Norway, in 1999 and 2004, respectively. He is currently a Professor with the Department of ICT, University of Agder, Norway, and also the Director of the Center for AI Research. He has authored over 70 refereed journal and conference publications. His research interests include intelligent systems, stochastic modeling and inference, machine learning, pattern recognition, learning automata, distributed computing, and surveillance and monitoring.

---


