On Using "Stochastic Learning on the Line" to Design Novel Distance Estimation Methods

Jessica Havelock[1], B. John Oommen[1,2], and Ole-Christoffer Granmo[2]

[1] School of Computer Science, Carleton University, Ottawa, Canada
[2] Centre for Artificial Intelligence Research, University of Agder, Grimstad, Norway

Abstract. In this paper[3], we consider the problem of Distance Estimation (DE) when the inputs are the x and y coordinates 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 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 GoF 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, demonstrating that our scheme is novel and pioneering.

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

1 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

[3] The second author gratefully acknowledges the partial support of NSERC, the Natural Sciences and Engineering Council of Canada.

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", which are the *physical* distances to be traveled on the "roads" built in the community, is much more challenging, and far from trivial. 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 four decades, and its solutions have been put to use in many practical applications, such as in developing vehicle scheduling software, vehicle routing, and in the partitioning of districts for firefighters [1, 5]. Indeed, as alluded to earlier, this is a central issue in designing GISs and GPSs.

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 [3]. The first DEFs were based on common norms, most of which are still used. All these DEFs involved parameters whose values are obtained by a "training" phase in order for them to best fit the data of the system being characterized. Consequently, some "true" road 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.

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 [4]. To date, it has been central to two related problems (and their respective applications), namely, the continuous Stochastic Point Location problem, and the problem of parameter learning from a stochastic teacher or a stochastic compulsive liar. 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 the parameter sought for. The most "daring" step that we have taken in DE 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. These concepts will be explained presently.

2 Distance Estimation: Core Concepts

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. Clearly, 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 $\pi(P_1, P_2|\Lambda) : R^d \times R^d \longrightarrow R$, in which $P_1 = \langle x_1, x_2, ..., x_d \rangle$ 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 GoF function, which, in turn, is used to measure how well a network or region is represented by the DEF. Central to the legacy methods of DE is the above-mentioned concept of 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 most commonly-used GoF function is the sum of Square Deviation (SD).

2.1 Distance Estimation Functions (DEFs)

The most common types of DEFs are those based on the family[4] of L^p norms, traditionally used for computing distances:

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

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 [3]. 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 these parameters have been determined, the DEF can thereafter be invoked for estimating distances for other cities whose inter-city distances are unknown.

[4] 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.

3 The Adaptive Tertiary Search and its use in DE

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 this paper, we shall use the ATS [4] 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.

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 1. This is repeated until the search interval is smaller than the resolution of accuracy. The search interval will be reduced to yield the required resolution within a finite number of epochs because the size of the search interval is decreasing [4]. After the search interval has been sufficiently reduced, the midpoint of the final interval is returned as the estimate for λ^*.

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 1. The Decision Table for the ATS scheme.

The ATS proposed by Oommen and Raghunath [4] was initially used to solve the SPL problem, and subsequently for parameter learning when interacting with a stochastic teacher or a stochastic compulsive liar. 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 interact with it.

3.1 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 1.

Algorithm 1 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 1
  GetNewInterval for p - From Table 1
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

The set of LA operate in the same manner as in [4], 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 3.3.

3.2 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 the LA algorithm briefly described below. This is omitted here in the interest of space but found in [2]. 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 [2] (omitted here in the interest of space),

3.3 The Corresponding Environment

Each LA requires feedback from a specific Environment. This feedback informs the LA if it has made the correct decision, i.e., choosing the right or left half of the subinterval. It is easy to obtain this answer because it only involves a single parameter at a time. To further explain this, consider the DEFs below:

$$F(k,p) = k \cdot F_1(X_1, X_2, p), \text{and where,} \tag{2}$$

$$F_1(X_1, X_2, p) = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p \right)^{1/p}. \tag{3}$$

Although nothing specific can be said about the monotonicity characteristics of $F(k,p)$, we see from Eq. (2) 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. (3), since $\mathrm{F}_1(X_1, X_2, p)$ 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 accordingly 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 a corresponding algorithm (*EnvironmentResponseP*, found in [2]) 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.

4 Testing and Results: 2-dimensional Environments

In this section[5], 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, the first two DEFs each contained only a single parameter that had to be be determined, k or p respectively. The last DEF contained two parameters, k and p. To compare the results we used four typical GoF measures.

Experimental Setup: Our 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) [6]. 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. "Benchmark" values were therefore used for comparison, determined using a hill-climbing search.

Weighted L^p DEF: When the ATS was used in conjunction with the Weighted L^p DEF, the ATS out-performed the hill-climbing search, as shown in results in Table 2. 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 was 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. The most significant contribution of this work was that the ATS did not require the use of GoF functions, which we believe is pioneering and novel.

5 Conclusions

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. Our

[5] The experimental results that we have obtained are extensive and involve two artificial and two real-life data sets. The results presented here constitute only a small subset; additional details of the experimental results are found in [2].

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

solution differs significantly 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) 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 – without utilizing GoFs. The other contribution that we made (with regards to the ATS) was to successfully search for multiple parameters *simultaneously*.

References

1. Erkut, H., Polat, S.: A simulation model for an urban fire fighting system. OMEGA - The International Journal of Management Science 20(4), 535–542 (1992)
2. Havelock, J., Oommen, B. J. and Granmo, O.-C.. Novel Distance Estimation Methods Using "Stochastic Learning on the Line" Strategies. Unabridged version.
3. Love, R.F., Morris, J.G.: Modelling inter-city road distances by mathematical functions. Operational Reach Quarterly 23(1), 61–71 (1972)
4. Oommen, B.J., Raghunath, G.: Automata learning and intelligent tertiary searching for stochastic point location. IEEE SMC 28(6), 947 – 954 (1998)
5. Oommen, J., Altnel, I.K., Aras, N.: Discrete vector quantization for arbitrary distance function estimation. IEEE SMC 28(4), 496 – 510 (1998)
6. Skorobohatyj, G.: 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