Enhancing RTT Prediction Schemes Using Global Function Minimization

Dragan Milić, Torsten Braun
Institute of Informatics and Applied Mathematics
University of Bern, Neubrückstrasse 10, 3012 Bern, Switzerland
email: {milic|braun}@iam.unibe.ch phone: +41 31 511 {2633|2631}

***Abstract*—Numerous round trip time (RTT) prediction schemes use the least squares method to embed hosts in virtual euclidean spaces. The least squares method minimizes the residuals between measured data (measured RTTs) and their approximation (euclidean distances between the host position and fixed points, to which the distance was measured). This is achieved by minimizing an objective function, which is defined as a sum of square differences between measured distances to fixed points (landmarks) and euclidean distances to those landmarks in a virtual space. Since there is no direct way (closed form) for finding minima of the objective function, numerical function minimization must be used. In this paper we identify the problem of existence of multiple local minima of objective functions and their impact on resulting RTT predictions. To overcome this problem, we propose an algorithm for finding all local minima of the objective function. By finding all minima, we are able to identify the global minimum of the objective function, and thus ensure the optimal embedding of a host in the virtual space. To evaluate our algorithm we compare it with standard methods for function minimization using data collected by the Planet-Lab all-pings experiment.**

I. Introduction

Round-trip-time (RTT) is one of the most important properties of Internet communication, due to its role as a limiting factor for the effective bandwidth of TCP connections [5] and its impact on delay sensitive real time applications such as teleconferencing, massive multiplayer on-line games etc. The idea behind a RTT prediction scheme is not to measure RTT between each pair of hosts in the network, but to to have a more or less precise estimate of RTTs between all hosts in the network based on a small amount of measured RTTs. Such a RTT prediction scheme can be used to optimize overlay network structures, to choose nearest data source for file transfer, or to find an optimal route based on predefined QoS parameters.

Numerous RTT prediction schemes already exist. Most of them have been developed in the last years. The most promising among them are based on embedding hosts in a (low degree) dimensional euclidean space. Almost all of such RTT prediction schemes are based on minimizing a so called "objective function" to embed the host into the euclidean space. Since there are usually no direct (closed form) methods for finding the minimum of such a function, minimization is only possible using numerical (iterative) methods. Besides the non-negligible computational overhead of function minimization, such methods are only able to find one minimum of the function. The problem is, that the objective functions that are used to embed hosts in virtual spaces can have more than one minimum and only one of them is the minimum, which we are interested in: the global minimum.

In this paper, we investigate the magnitude of the problem of local minima existence based on real Internet RTT measurements. We show that the number of local minima of the objective function and the probability of not finding the global minimum affects a considerable percentage of hosts. The consequence of this is that the hosts are not optimally positioned in the virtual space, which reduces the precision of the RTT prediction. We also propose an algorithm, which is in most of the cases able to find the global minimum of the objective function.

This paper is structured as follows. In the next Section we give an overview of related work in the field of RTT prediction. In Section III we motivate the need for finding global minimum of the objective function by showing the results of our research regarding the frequency of local minima occurrence based on data measured in the Planet-Lab all-pings experiment. Section IV describes the principles of different multimodal numerical function minimization techniques. In the same Section we also illustrate the impact of using different minima on RTT prediction error. We also show using a constructed example, the existence of local minima even in an ideal case, where RTTs are originating from euclidean space distances (no violations of the triangle inequality). In Section V we describe our proposed method for finding global minima of the common objective functions used for embedding the hosts in the virtual euclidean space. In Section VI we present the evaluation of our algorithm for finding the global maximum described in Section V and compare it with plain GNP.

II. Related Work

In the last years there have been numerous proposals for RTT prediction schemes based on coordinates [2]–[4], [7], [9]–[11]. Essentially there are two types of coordinates based RTT prediction schemes: landmarks-based and landmarks-less. The landmarks-based RTT prediction schemes determine coordinates of a few fixed reference points (so called landmarks) and use them to determine the positions of all other hosts. Some of them are based on positioning landmarks using function minimization [2], [9], [10]. The others [7], [11] use the principal component analysis (PCA) to reduce

the dimensionality of the landmark RTT measurements and to position the hosts. Landmarks-less RTT prediction schemes such as VIVALDI [3] or S-VIVALDI [4] are based on a distributed simulation of physical systems to iteratively reduce the overall error of embedding a host in an euclidean space.

The first and the most promising landmarks-based RTT prediction scheme is GNP [9], [10]. In GNP, m hosts are chosen as landmarks (denoted by $\mathcal{L}_1 \dots \mathcal{L}_m$). All other Hosts (denoted by $\mathcal{H}$) in the Internet measure RTT distances (denoted by $\hat{d}_{\mathcal{H}\mathcal{L}_1}, \dots, \hat{d}_{\mathcal{H}\mathcal{L}_m}$) to the landmarks and use this information to determine their position in a virtual space. The coordinates of a host in a virtual space (denoted by $\mathcal{C}_\mathcal{H} := (\mathcal{C}^1_\mathcal{H}, \dots, \mathcal{C}^n_\mathcal{H})$) are determined using multilateration. The multilateration itself is based on minimizing the error function f_e. In the case of GNP, the least-squares-error function

$$f_e(\mathcal{C}_\mathcal{H}) := \sum_{i=1}^{m} (d(\mathcal{C}_\mathcal{H}, \mathcal{C}_{\mathcal{L}_i}) - \hat{d}_{\mathcal{H}\mathcal{L}_i})^2 \quad (1)$$

is used. The function minimization is performed using the Downhill Simplex [8] method.

For calculating $d_\mathcal{S}(\mathcal{C}_\mathcal{H}, \mathcal{C}_{\mathcal{L}_i})$, the positions of the landmarks in the virtual space ($\mathcal{C}_{\mathcal{L}_1} \dots \mathcal{C}_{\mathcal{L}_m}$) must be known. In the case of GNP, the coordinates of the landmarks are computed by minimizing the following error function:

$$f_e(\mathcal{C}_{\mathcal{L}_1}, \dots, \mathcal{C}_{\mathcal{L}_m}) := \sum_{i,j \in \{1 \dots m\}, j>i} (d(\mathcal{C}_{\mathcal{L}_i}, \mathcal{C}_{\mathcal{L}_j}) - \hat{d}_{\mathcal{L}_i\mathcal{L}_j})^2$$

III. Motivation

The mere existence of local minima does not necessarily mean that the RTT prediction scheme proposed in GNP produces significant difference in RTT prediction. To show how the magnitude of the problem, we have performed a series of experiments with goal to determine how the portion of the hosts that are affected by the existence of the local minima. Another goal of our experiments is to determine if there is a significant degradation of RTT prediction by using a local instead of the global minimum for the host embedding.

A. How Frequent are Multiple Minima?

To determine how frequent local minima of the objective function really occur, we have performed a so called "monte carlo" minimum search. The idea is to perform large numbers of numeric minimum searches – each time with an other randomly chosen start point. If we would perform an infinite number of such function minimizations, we would be able to find every local minimum for any function. Since we know that the number of minima is relatively low (usually not more than 5), performing a relatively low number of such minimizations would result in a good estimate about how many minima there are and where they are located. In this case we used 400 function minimizations.

Since we are using numerical methods, we still need a non-strict method for deciding, whether if two minima are "equal". For this experiment, we considered two local minima m_1, m_2 as equal if the following inequality holds: $d(\mathcal{C}_{m_1}, \mathcal{C}_{m_2}) < 2$. We have agreed on the value of 2 since the units that are used in the all-pings experiment are milliseconds and a positioning error of two milliseconds is an acceptable compromise between the number of false positives and the prediction error – we are looking for errors of much larger magnitude. Due to this, our method cannot distinguish two local minima if the euclidean distance between them is less than two milliseconds. This is acceptable for us, since we are not interested in an exact number of minima, but only in an estimate of how many local minima there are, that are located "far" away from each other. At the beginning we used the Downhill Simplex method for the function minimization. As the result of this experiment, we have obtained a huge amount of local minima located near each other (but more than 2 ms apart). Further investigation has shown that the main reason for this effect was not the existence of numerous local minima, but the function minimization method used. Often the Downhill Simplex method got stuck in a point that was not a local minimum. The reason for this is most probably due to the collapse of the simplex – a known issue of the Downhill Simplex method, where the used simplex "looses" at least one dimension due to choosing a "better" point, which is near to the plane defined by other points of that simplex. To eliminate this effect, we decided to use a more robust Conjugate Gradient method to position the hosts, which yielded in much better results.

For our experiment, we have taken one snapshot on the RTT information measured by the Planet-Lab all-pings experiment [13]. Since the data of the all-pings experiment is not complete, we removed some hosts from the data set to obtain a complete distance matrix containing measured RTTs between each pair of hosts. The final distance matrix contained the complete distance information for 218 hosts scattered around over the Internet. We used a fixed number of landmarks (12) and calculated for a varying number of dimensions (2 to 9) their coordinates using the method proposed by GNP (Downhill Simplex). For all other hosts, which are not landmarks, we have calculated their positions in the virtual space by minimizing the objective function (1). This method allows us to find the percentage of hosts, for which the objective function has more than one local minimum. Those results are represented in Figure 1. The gathered data also allows us to calculate an estimate on the probability for finding the global minimum by starting the minimization in a random point within the boundaries: by dividing the number of random starting points that "landed" in the global minimum by the total number of starting points (400). The average of those probabilities are represented in Figure 2. Our experiment shows that the number of the hosts, for which the objective function has more than one minimum is dropping with the number of dimensions. The probability of finding the global minimum by starting at a random point within the boundaries is more or less constant at 60% for a number of dimensions between 2 and 5. We can also see, that for more than five dimensions the number of local minima found is practically zero. Unfortunately, since the computational complexity of a function minimization increases rapidly with the number

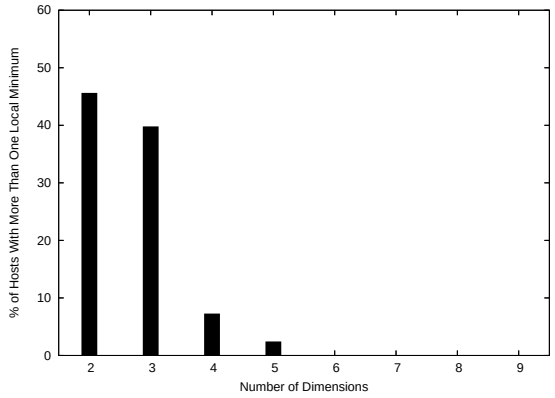

Fig. 1. Percentage of hosts with more than one local minimum of the objective function.

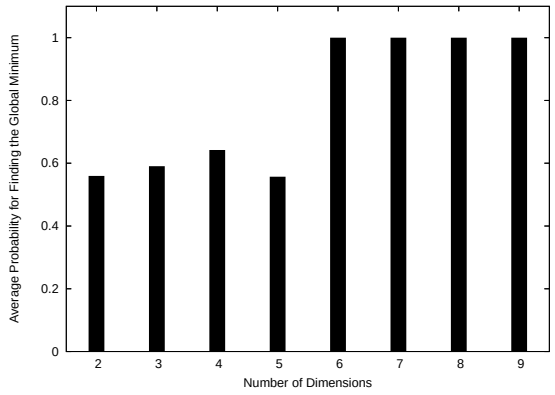

Fig. 2. Average probability for finding the global minimum by starting the function minimization at a random point

of dimensions used, we cannot rely on simply increasing the number of dimensions to reduce the probability of the existence of the local minima.

B. Local Minima of the Objective Function

One might think that the only reason for the existence of local minima are the violations of the triangle inequality of the measured data (caused usually by policy-based routing in the Internet). To show that this is not the case, we have constructed the following example:

We take a two-dimensional euclidean space and four landmarks $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ with the following coordinates: $\mathcal{C}_{\mathcal{L}_1}$: $(0, 180)$, $\mathcal{C}_{\mathcal{L}_2}$: $(0, -180)$, $\mathcal{C}_{\mathcal{L}_3}$: $(30, -30)$, $\mathcal{C}_{\mathcal{L}_4}$: $(30, 30)$ and a host with the coordinates $\mathcal{C}_{\mathcal{H}}$: $(-100, 0)$. If we calculate the euclidean distances between the host and the four landmarks and use them (together with the landmark positions) to construct the usual objective function (1) used for host embedding, we obtain a function, which has two local minima (see Figure 3). As the Figure clearly shows, the objective function obtained has not only one, but two local minima. One of them is located at the real position of the host $(-100, 0)$. The other one is located behind the "pass" located between the "peaks" formed by the two nearest landmarks ($\mathcal{L}_3$ and $\mathcal{L}_4$).

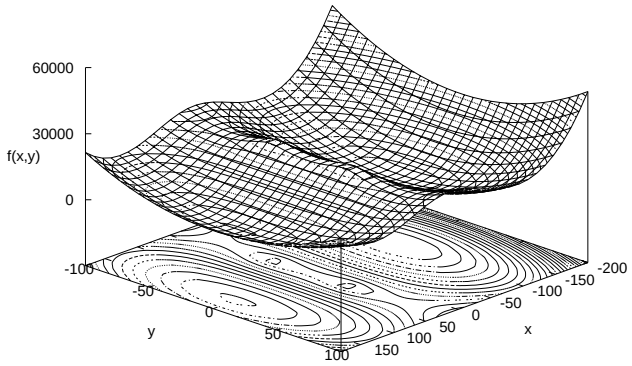

Fig. 3. Objective function demonstrating the existence of local minima in the case where measured distances are "ideal".

The function values at the local minima are different. While the value of the objective function at the global minimum (at the original position of the host: $-100, 0$) is zero, the value at the other global minimum is larger than zero, but surrounded by points with larger values.

There are only few proposals for a generalized approach to finding a global minimum [6]. None of these approaches utilizes the information about the morphology of the function. We feel that for this special case, a better approach to finding the global minimum can be found if we analyze the properties of the objective function that we are minimizing.

C. Impact of Local Minima on RTT Prediction

As an illustration how local minima affects the RTT predictions we show the CDF of the relative error of RTT prediction for one host with three local minima in two dimensions. Figure 4 shows a CDF of relative error $\left|\frac{d(\mathcal{C}_{\mathcal{H}_1}, \mathcal{C}_{\mathcal{H}_i}) - \hat{d}_{\mathcal{H}_1\mathcal{H}_i}}{\hat{d}_{\mathcal{H}_1\mathcal{H}_i}}\right|$. As we can see, the use of the global minimum results in a smaller RTT prediction relative error compared to using the non-global minima.

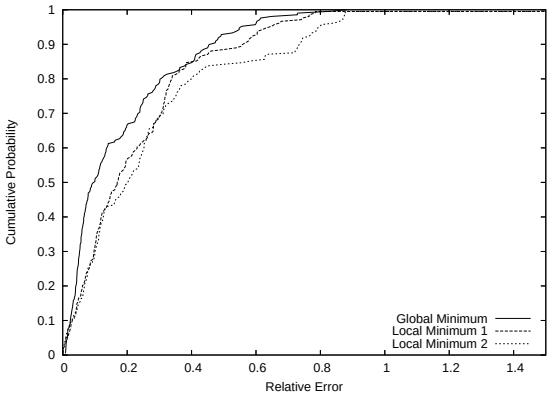

Fig. 4.

IV. Numerical Methods for Finding Local Minima

Finding maxima or minima of a function is a well researched area of mathematical analysis. For continuous functions with a known first derivative (gradient), finding local minima is as simple as finding all points where the gradient is equal to zero and evaluating the function value in those points. In such way, we can find all minima, maxima and saddle points.

The problem arises when there is no closed form for determining all the points, where the gradient of the function is zero. In such cases, numerical methods are used to find local minima (finding a maximum of a function is equal to finding a minimum of the negative of that function). All of those numerical methods start from one or multiple points in the parameter space as initial guess, which is considered to be the current solution. This current solution is then iteratively refined by replacing it with a better one that is obtained by an iterative step. Each of the iterative numerical function minimization strategies also has an terminating condition - the heuristic for telling if further refinements do make sense by considering the computational cost of the last refinement step and the level of refinement reached trough it. Usually, the terminating condition is either the distance of the last two solutions in the parameter space or the function value difference between the last two refinements. The method to obtain a better refinement of the current solution distinguishes the numerical function minimization strategies. In this paper we describe the two most popular of them: Downhill Simplex and conjugate gradient

A. Downhill Simplex

Downhill Simplex [8] function minimization is one of the most general and hence most popular function minimization methods. Its popularity is based mostly on the fact that it needs neither first nor second derivative information of the function to perform the minimization.

The Downhill Simplex algorithm performs as follows: a starting point of the algorithm is a simplex (the simplest geometric figure, which can be constructed in a n-dimensional space using $n + 1$ points) defined in the parameter space of the objective function. The points of this simplex must be affine independent (can be used as a base for a n dimensional space) – otherwise the Downhill Simplex method would not be able to minimize the function value in each dimension. The Downhill Simplex function minimization always tries to find a better simplex than the current one by moving the "high" point (the point of the simplex for which the value of the objective function is maximal) somewhere into the parameter space. To do so, Downhill Simplex tries different transformations of the simplex such as stretching, contracting or mirroring. If none of those moves yields a better point Downhill Simplex contracts the simplex towards the "low" point of the simplex. The price of the flexibility of the Downhill Simplex method is a quite large number of evaluations of the objective function needed to perform the minimization, since it does not make use of additional information about the objective function such as its gradient.

B. Gradient Methods

If the gradient of the function can be computed efficiently, one can use the gradient methods such as Steepest Descent or Conjugate Gradient methods to find a local minimum (for a good introduction to different non-linear function minimization methods see [12]). The procedure for all gradient methods is the same: First, a direction of the search is chosen based on the function gradient at the current solution and eventually based on previous current solutions. After choosing a direction, a one-dimensional minimum search is performed on a line defined by the current solution and the search direction. To perform the one-dimensional function minimization, first the minimum is bracketed by finding a triplet of parameters a, b, c where $a < b < c, f(a) > f(b), f(c) > f(b)$. Then, the bracketed minimum is iteratively reduced, until the difference between a and b is smaller than the given threshold (Brent's golden section search – for details see [1]). The newly found minimum is now the new solution and the procedure for the function minimization is re-iterated until the function value difference of the newly found solution is below the defined threshold.

C. Other Methods

The last two methods discussed are the most commonly used ones for function minimization. In addition to those there are numerous other methods, which we did not mentioned, such as Newton's Iterative, Gauss-Newton, different hybrid methods etc. Those methods are not discussed in this paper, since the two methods mentioned above are sufficient for our purposes. The strategy for the choice of direction is different for every Gradient Method. For example, the Steepest Descent methods decides at each iteration to follow the negative of the gradient (the direction, in which the function has the steepest decrease). It has been shown that this is not an optimal decision in the general case (see [12]) and that other strategies that involve the knowledge of previous local minima, such as Conjugate Gradient are more efficient.

V. Proposed Method to Find the Global Minimum

Most of the general function minimization methods make very few assumptions about the function to be minimized. Generally this is a good idea, since the method can be applied to a wide range of functions. On the other hand, by exploiting additional information about the function known to us, we can create a less general, but more effective method to find the global minimum of the function. In this Section, we describe one such method we developed to find the global minimum of the family of functions described by (1).

A. Limiting the Function

As defined in (1) the objective function is unbounded. This means that the definition range of the objective function is $\mathbf{R}^n$. Still, there must be a range where local minima can be found. This range is important, since we are performing a minimum search in one direction within the parameter space and we need to know at which point we should stop searching for the next

minimum or maximum. The "range" of the objective function we are using is defined as follows: the objective function is "limited" by a hyperball where the center is the center of all landmark positions. The diameter is equal to a maximum of all radii for each landmark. The radius for one landmark is defined as sum of the euclidean distance of the landmark to the center of all landmarks and the double of the measured RTT to that landmark. The rationale behind the choice of such a boundary is that it must be easy to compute and that it includes all possible local minima. Since we define the boundary as a hyperball checking if one point is within the boundary is as simple as computing the euclidean distance from the point to the center of all landmarks and comparing this distance with the hyperball radius. The choice of the hyperball radius makes sure that each measured RTT can reach its double length. We have chosen this, since the positioning of the host (minimizing the objective function) in the case where the embedding error is not zero, is equivalent to changing the distances to the landmarks. Meaning that in the process some distances are getting shorter, some are getting longer. With our choice we make sure that the boundary allows enough room for stretching the measured distances in the worst case.

B. Dissecting the Objective Function

The objective function is usually defined as follows:

$$f(\mathcal{C}_\mathcal{H}) := \sum_{i=1}^{m} (d(\mathcal{C}_{\mathcal{L}_i}, \mathcal{C}_\mathcal{H}) - \hat{d}_{\mathcal{L}_i\mathcal{H}})^2$$

For each landmark ($\mathcal{L}_i$) there is one summand of the form:

$$f := (d(\mathcal{C}_\mathcal{L}, \mathcal{C}_\mathcal{H}) - \hat{d}_{\mathcal{L}_i\mathcal{H}})^2$$

Represented separately in two dimensions, the shape of this summand looks like in Figure 5. As we can see, one summand

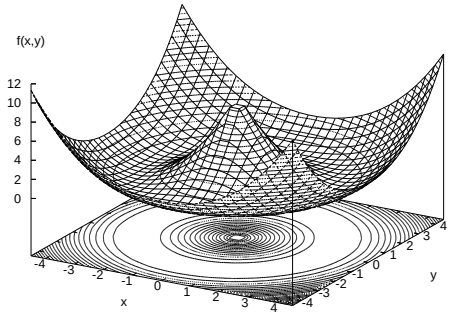

Fig. 5. Single summand of the objective function represented in a two-dimensional space.

of the objective function consists of a "peak" at the position of the landmark. The more we are moving away from the landmark position, the more the function values are decreasing until a circular valley is reached, which is on the exact distance to the landmark, as the distance measured from the host. The valley itself is surrounded by a "wall" (a hill with infinite height), which is getting steeper with increasing distance from the landmark. Since the valley around the peak is not just a single point, but a circle, there is an infinite number of minima (a circle with the radius equal to the measured distance to the host) of this function. Now, if we add two summands (two landmarks) the "landscape" of the function would look like in Figure 6. With two summands we can see that the "landscape"

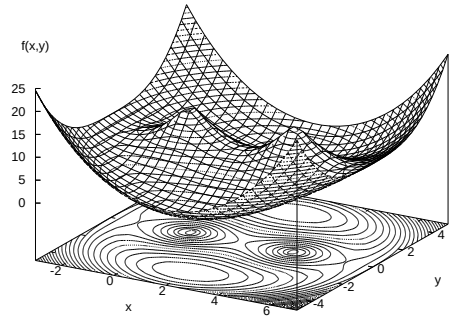

Fig. 6. Objective function with three landmarks represented in a two-dimensional space.

has two peaks at the positions of the corresponding landmarks and two valleys. Each valley of this landscape has its minimum on the same "height" (function value), meaning that there are two possible solutions to position the host. To be able to decide where the host has to be actually positioned we need the measurement to a third landmark.

In Figure 7 we can see the "landscape" of the final objective

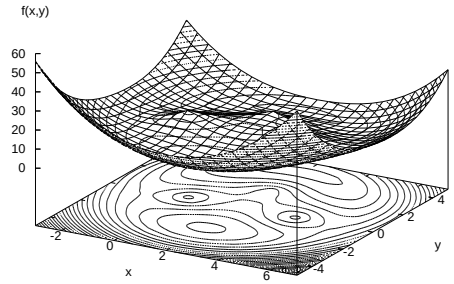

Fig. 7. Objective function with three landmarks represented in a two-dimensional space.

function with three landmarks. As in Figures 5 and 7 we have one "peak" per landmark. Similar to Figure 6 we have two valleys (minima), but this time the function value in one of the minima is lower than in the other. The minimum with the lower function value is the minimum we want to find: the global minimum. If we start a numerical method for function minimization such as Downhill Simplex or some kind of a Gradient method, depending on the starting point for the

function minimization we are either going to land in one or the other minimum. What we are trying to achieve with our approach is to find most of the local minima. If we are able to do so, we can also find the global one, since one of them must be the global minimum.

C. Exploring the Function Landscape

After finding one of the minima, we propose finding others by "climbing" out of the valley. The first question that arises is, how do we know in which direction should we climb? For making this decision we have the following information: we know the positions of the "peaks" (landmark positions) and we know our current position. For making the decision in which direction to climb, we should recall the "landscape" defined by two landmarks (shown in Figure 6). When we have two landmarks in two dimensions, the function "landscape" has one saddle point that is located between those two landmarks. If we were in one of the valleys in Figure 6, the shortest path, for reaching the second valley, would be to climb the hill crossing the saddle point between the peaks. We translate this analogy into mathematical language: we should try moving within the parameter space of the objective function into the direction defined by the position of the current local minimum and the orthogonal to a line drawn trough the positions of the pair of landmarks. Since we do not "know" on which side of the hill we are located, we should try moving in both directions defined by such a vector.

D. Analogy for "Climbing the Hill"

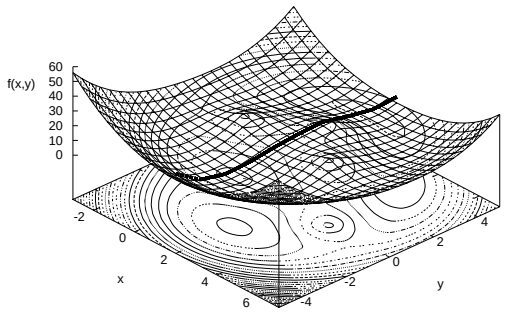

Fig. 8. One direction for leaving the valey of a local minimum.

The next question is: what is the mathematical equivalent to climbing a hill? The answer is: moving into one direction in the parameter space (x-y plane on Figures 5, 6 and 7) until the function stops increasing else we have left the "boundaries" of the function. For moving into one direction in the parameter space we need a fixed point P and a direction vector x. By having that, we can define climbing the hill by finding a parameter $\lambda > 0$, for which the one-dimensional function $f'(\lambda) := f(P + \lambda \cdot x)$ has a maximum, which is equivalent to finding a minimum of $-f'(\lambda)$. Figure 8 shows one such direction. If we are able to find a minimum for $-f'(\lambda)$ we know that we have reached the saddle point (or a point near it) and that the function value of $f'(\lambda)$ is decreasing beyond this point, meaning that we can now start descending into a valley.

E. Descend Into the Next Valley

After finding the maximum of $f'(\lambda)$ we can now start to descend into the next valley. Since we are using a numerical method for finding a maximum of $f'(\lambda)$ we cannot just start a multivariate function minimization in the point $P + \lambda x$ since there is a possibility that we would fall back into the starting point of the climb (old valley). To eliminate this possibility, we propose finding the next minimum of $f''(\mu) := f'(\lambda + \mu) = f(P + (\lambda + \mu)x)$. Now we can start a gradient based multivariate function minimization at that point. The result of this function minimization can be either a new minimum (a new valley) or the old minimum, if the mountain we have just crossed leads back to the previous valley. This procedure can be re-iterated until the "boundary" of the function is reached.

F. Handling Multiple Dimensions

The whole analogy we have been using so far is specific to a two-dimensional parameter space. The third dimension, which is actually defining the shape of the landscape in our examples is the value of the objective function.

Since the RTT prediction schemes based on minimizing the objective function are usually gaining in precision with an increasing number of dimensions, we also need to consider how we could find all valleys in a generalized case of an n-dimensional parameter space.

For leaving the current valley we need a direction vector. For this vector we propose using an orthogonal vector to a hyper-plane defined by n landmarks. In a two dimensional parameter space, it would be a line orthogonal to a line defined by two landmarks. For three dimensions the direction would be a line orthogonal to a plane defined by three landmarks etc. Given the positions of n landmarks $(\mathcal{C}_{\mathcal{L}_1}, \ldots, \mathcal{C}_{\mathcal{L}_n})$ the orthogonal vector $x := (x_1, x_2, \ldots, x_n)$ to a hyper-plane defined by those landmarks can be computed as the following determinant:

$$\begin{vmatrix} (\mathcal{C}^1_{\mathcal{L}_2} - \mathcal{C}^1_{\mathcal{L}_1}) & \cdots & (\mathcal{C}^1_{\mathcal{L}_{n-1}} - \mathcal{C}^1_{\mathcal{L}_1}) & x_1 \\ (\mathcal{C}^2_{\mathcal{L}_2} - \mathcal{C}^2_{\mathcal{L}_1}) & \cdots & (\mathcal{C}^2_{\mathcal{L}_{n-1}} - \mathcal{C}^2_{\mathcal{L}_1}) & x_2 \\ \vdots & \ddots & \vdots & \vdots \\ (\mathcal{C}^{n-1}_{\mathcal{L}_2} - \mathcal{C}^{n-1}_{\mathcal{L}_1}) & \cdots & (\mathcal{C}^{n-1}_{\mathcal{L}_{n-1}} - \mathcal{C}^{n-1}_{\mathcal{L}_1}) & x_{n-1} \\ (\mathcal{C}^n_{\mathcal{L}_2} - \mathcal{C}^n_{\mathcal{L}_1}) & \cdots & (\mathcal{C}^n_{\mathcal{L}_{n-1}} - \mathcal{C}^n_{\mathcal{L}_1}) & x_n \end{vmatrix}$$

G. Algorithm for Exploring the Landscape

In the previous sub-section we have defined everything we need to explore the landscape of the objective function. Now we put all of that together into an algorithm, which is using an "educated guess" about the landscape of the objective function to try to find all valleys of the landscape defined by it.

The algorithm uses two sets: a set of starting points (S), which should be used to perform a numerical function minimization and a set of known local minima (M). At the beginning the set of the known local minima is empty and the set of the

starting points contains one starting point (randomly chosen or obtained in some other manner). The first step of the algorithm is to remove one point from the set of the starting points and to perform a multilateral function minimization (for example using the Conjugate Gradient method) with that starting point. The result of the function minimization is one local minimum. Our algorithm checks whether the found local minimum is in the set of known local minima. If this is not the case, it is added to the set of known local minima and is used to find new starting points.

To find new starting points for the search based on a new local minimum m we use the following procedure: For every subset of landmarks used in the objective function we calculate an orthogonal vector dir (for details see Section V-F). For each such orthogonal vector we try to leave the minimum's valley by finding the next maximum of the one dimensional function $f'(\lambda) := f(P + \lambda \cdot dir)$ (see Section V-D) and the minimum that is behind that maximum by minimizing $f''(\mu) := f'(\lambda + \mu)$. If we are able to find one such pair of maximum/minimum, we take that starting point defined as $m' = m + (\lambda + \mu) \cdot dir$ into the set of the starting points. We continue finding the next maximum/minimum pair in the same direction until we leave the "range" of the objective function. The same procedure for finding maximum/minimum pairs is also done in the opposite direction ($dir' := -dir$), since we cannot be sure in which direction the hill really is. After all starting points relative to a new minimum P are found, the new minimum is taken into a set of known local minima and the whole algorithm is re-iterated. The algorithm terminates when the set of the starting points is empty. A more formal version of this algorithm can be found in Algorithm 1.

Algorithm 1 Algorithm for enlisting all local minima

Require: LMS set of landmark positions
Require: DST vector of distances measured to landmarks
Require: D number of dimensions
  $S \leftarrow \{\text{random point}\}$ // *Start with a random point*
  $M \leftarrow \emptyset$ // *Start with an empty set of local minima*
  for $p \in S$ **do**
    $m \leftarrow find\_min(p, LMS, DST, D)$ // *Find a local minimum*
    if $m \notin M$ **then**
      for $os \in \{x | x \in LMS, \|x\| = D\}$ **do**
        $dir \leftarrow orthogonal(os)$ // *Calculate the orthogonal vector to landmark positions*
        $\lambda \leftarrow 0$
        while $m + \lambda \cdot dir$ within function boundaries **do**
          $\lambda \leftarrow next\_max(\lambda, m, dir)$ // *Find next one-dimensional function maximum relative to λ starting at m along the direction vector dir*
          $\lambda \leftarrow next\_min(\lambda, m, dir)$ // *Find next one-dimensional function minimum relative to λ starting at m along the direction vector dir*
          if still inside function boundaries **then**
            $S \leftarrow S \cup \{m + \lambda \cdot dir\}$
          end if
        end while
        $\lambda \leftarrow 0$ // *Perform search also in the negative direction*
        while $m + \lambda \cdot -dir$ within function boundaries **do**
          $\lambda \leftarrow next\_max(\lambda, m, -dir)$ // *Find next one-dimensional function maximum relative to λ starting at m along the direction vector dir*
          $\lambda \leftarrow next\_min(\lambda, m, -dir)$ // *Find next one-dimensional function minimum relative to λ starting at m along the direction vector dir*
          if still inside function boundaries **then**
            $S \leftarrow S \cup \{m + \lambda \cdot -dir\}$
          end if
        end while
      end for
    end if
  end for
  return S

H. On the Computational Complexity of the Algorithm

The computational complexity of the algorithm we are proposing, depends on the following variables: number of dimensions (denoted by n), number of landmarks (denoted by L) and number of local minima of the function (denoted by M). Since it is very difficult to analytically determine the computational overhead of a numerical method for multimodal function minimization, we will only give the best case and worst case estimations of the number of such local minimizations performed by our algorithm.

Considering the number of local minima M, the best case is the one where the function (1) has only one minimum: $M = 1$. In this case, our algorithm performs one function minimization to find that minimum and afterwards tries to "climb" out of the valley of that minimum using different directions. Each direction is calculated based on positions of n landmarks chosen from the given set of L landmarks. Therefore, the number of possible directions is the number of combinations without repetition: $\binom{L}{n} = \frac{L!}{n!(L-n)!}$. For each such direction we perform two searches: one in the positive and one in the negative direction. Each of these searches results in a certain number of possible starting points for further function minimization. The worst case for the number of possible starting points is obtained if all landmarks are located on one straight line determined by the search direction and the already found local minimum. In such a case, our algorithm will find $L - 1$ new starting points for the local minimum search. As a result, the number of local minimum searches performed by our algorithm is $\frac{2(L-1)L!}{n!(L-n)!}$. This is of course the upper boundary for the worst case. In the best case no new starting points will be found and hence only one local minimization will be performed.

For the general case where the number of local minima is $M > 1$, we can assume that in the worst case each minimum found will yield $\frac{2(L-1)L!}{n!(L-n)!}$ starting points for the

local search. This gives us the upper boundary of the worst case: $\frac{2M(L-1)L!}{n!(L-n)!}$. In the best case each minimum will yield only one new starting point, in which case the number of performed local minimum searches is equal to M.
It is obvious that our algorithm can yield a large number of local function minimizations. Especially in the case where the dimensionality of the virtual space n is low and the number of landmarks L is high, our algorithm will perform very large amount of local searches. On the other hand, with a decreasing number of dimensions, the computational overhead for finding the local minimum is also decreasing. Also, compared with other global minimum search algorithms, our algorithm guarantees to terminate in a finite number of steps, assuming the non-pathological case of (1), where the number of local minima M is finite. This is also an advantage to other random start point methods, which lack a good terminating criterion.

VI. Evaluation

To verify if our proposed algorithm increases the probability of finding the global minimum we used the following experiment. We have compared the results of the standard GNP host positioning system in terms probability to find the global minimum using the same data used to show the existence of the multiple minima of the objective function from Section V. To have comparable results, we have implemented the host positioning from GNP using Downhill Simplex function minimization provided by [12]. We also implemented our algorithm using conjugate gradient, linear minimization (golden section search with hyperbolic extrapolation) and one dimensional minimum bracketing functions provided by the same source [12]. For the experiment we used a fixed number of landmarks (10) and varied the number of dimensions between 2 and 8. To have comparable results and to avoid interference by using different landmark positions, we have based all our objective functions on the same landmark coordinates. The landmark coordinates were calculated only once (per number of dimensions) and used by both our and the GNP algorithm. For each host from our RTT distance matrix we have calculated the host coordinates using the GNP algorithm and our algorithm. We also estimated the total number of local minima of the objective function by performing Conjugate Gradient minimum search with 200 different starting points.
Figure 9 shows the percentage of global minima found by GNP and by our algorithm. As we can see, our algorithm was able to find the global minimum of the objective function almost for each host in our sample. However, there are some cases, where our algorithm is not able to find the global minimum. The reason for this is that our algorithm is using only a heuristic to minimize the probability of not finding the global minimum. it does not guarantee, that it will find *every* minimum there is.

VII. Conclusion

In this paper we identified the problem of the existence of multiple local minima of the objective function used to position hosts in a virtual space. The existence of multiple local minima has an impact on our perspective of minimizing

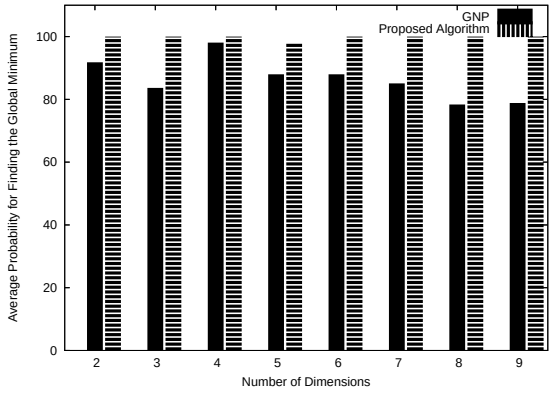

Fig. 9. Comparison of the probability for finding global minima between GNP and our algorithm.

a function. It is not enough to have a method that is fast and precise in finding a local minimum. We also need an algorithm that is able to find the global minimum of the objective function. Our research has shown that the existence of multiple minima of the objective function occurs too frequent that it could be ignored. To overcome this problem we have developed an algorithm, which exploits the knowledge of the objective function's "landscape" shape trying to find all local minima and thus the global minimum. Our evaluation has shown that our algorithm is performing much better than the plain host positioning by function minimization as proposed by GNP.

References

[1] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[2] M. Costa, M. Castro, A. Rowstron, and P. Key. Pic: Practical internet coordinates for distance estimation. In *International Conference on Distributed Systems*, Tokyo, Japan, March 2004.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, New York, NY, USA, 2004. ACM Press.

[4] C. de Launois. A stable and distributed network coordinate systems. Technical report, Université catholique de Louvain, December 2004.

[5] S. Floyd and K. Fall. Router mechanisms to support end-to-end congestion control. Technical report, Lawrence Berkeley National Laboratory, 1997.

[6] F. James. Minuit tutorial: Function minimization. In *CERN Computing and Data Processing School*, Pertisau, Austria, 1972.

[7] H. Lim, J. C. Hou, and C.-H. Choi. Constructing internet coordinate system based on delay measurement. In *Internet Measurement Confgerence 03*, October 2003.

[8] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[9] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordiantes-based approaches. In *IEEE Infocom02*, New York / USA, June 23-27 2002.

[10] T. S. E. Ng and H. Zhang. A network positioning system for the internet. In *USENIX 2004*, pages 141–154, Boston MA, USA, June 2004.

[11] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Internet Measurement Confgerence 03*, October 2003.

[12] W. T. Vetterling, S. A. Teukolsky, W. H. Prea, and B. P. Flannery. *Numerical Recipes in C++, The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition, 2002.

[13] C. Yoshikawa. Planetlab all-sites-pings experiment.