University of Windsor
Scholarship at UWindsor

Electronic Theses and Dissertations

10-19-2015

The point placement problem in an inexact model and its applications

Kishore Kumar Varaharajan Kannan
University of Windsor

Recommended Citation

Varaharajan Kannan, Kishore Kumar, "The point placement problem in an inexact model and its applications" (2015). *Electronic Theses and Dissertations.* Paper 5434.

The point placement problem in an inexact model and its applications

by

Kishore Kumar Varadharajan Kannan

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor
Windsor, Ontario, Canada
2015

The point placement problem in an inexact model and its applications

by

Kishore Kumar Varadharajan Kannan

APPROVED BY:

Tirupati Bolisetti
Department of Civil Engineering

Subir Bandyopadhyay
School of Computer Science

Asish Mukhopadhyay, Advisor
School of Computer Science

August 20,2015

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In the recent years, due to the advancement in computational tools and techniques to analyze the biological data, biologists have been actively engaged in conducting different experiments to study the arrangements of nucleotide sequence in a chromosome. This masters thesis focuses on the area of the computational methods for the genomic map problem.

Though the probe location problem under consideration is known to be NP-complete, it is possible to obtain approximate solutions. The distance geometry approach for achieving efficient and better results is shown here. This also solves the point placement problem when the available distance bounds on some probe pairs, correspond to adversarial responses to distance queries between some pairs of points.

DGPL program has also been implemented to construct a probe map. Finally some chosen results from the experiments and their significance have been discussed. The screenshots of the working of DGPL algorithm have been attached for better understanding.

DEDICATION

To my loving family, who has supported me in every step of my life

ACKNOWLEDGMENTS

I express my sincere gratitude to Dr. Ashish Mukopadhyay, without whose patient guidance and constant supervision, I would not have come so far.

I offer my sincere appreciation to the committee members, Prof. Tirupati Bolisetti and Prof. Subir Bandyopadhyay for their useful critiques and advice.

My special thanks to the love of my life Nitisha, who spent most of her time in active discussions and gave moral support that helped me to finish up my thesis.

My grateful thanks is also extended to my colleagues cum friends Pijus, Roy, Satish, Pramod and Prakash for their invaluable help throughout my Master's degree. Finally to my loving parents for their unmatched encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Distance Geometry [27] is the study of set of points based on the given distances between the pair of points. Nowadays a large community of researchers are actively working in the field of distance geometry because of the several real-life applications within it. Some of them are locating sensors in telecommunication networks, where the position of particular sensors and the distance between some sensors were known, the problem is to identify the position of all sensors in a space. Another interesting application in biology where the experimental techniques are able to estimate the distance between the pair of atoms of a given molecule and the problem would be to identify the three-dimensional conformation of a molecule.

1.1 The point placement problem

The point placement problem on a line is a distance geometry problem which is to determine the location of points uniquely(upto translation and reflection) by making the fewest possible pairwise distance queries of an adversary. The queries can be made in one or more rounds and are modeled as a graph whose nodes represent the points and there is an edge connecting two points, if the distance between the corresponding points is being queried. The simplest of all, the 3-cycle algorithm, has the following query graph.

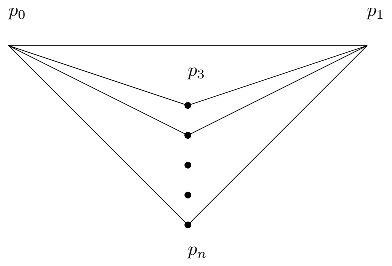


Figure 1.1: Query graph using triangles

If G = *(V,E)* is a query graph, an assignment l of lengths to the edges of G is said to be valid if there is a placement of the nodes V on a line such that the distance between adjacent nodes are consistent with l. Here in this problem, the distance between the pair of points returned by the adversary are exact. The algorithm designer tries to construct a graph over fixed number of rounds to minimize the number of edge queries and also make sure that there is a unique placement of vertices. The construction of such a graph is the heart of different algorithms for this problem.

1.2 Problem Statement

A classical version of the point placement problem is the construction of the coordinates of a set of n points, given inexact distance between some pair of points. This problem could also be termed as point placement problem in an inexact model as the distances provided were not exact. The inexact distances are given in terms of upper and lower bounds. Consider an example of three points with some distance constraints in it:

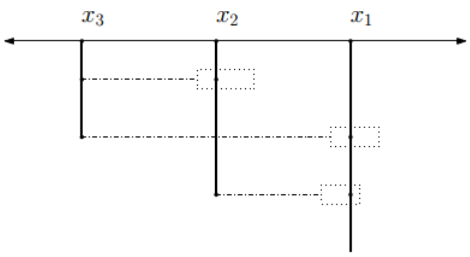


Figure 1.2: Embedding with inexact distances

Here in the figure, x_1, x_2 and x_3 are three points, the placement of those points are represented by a vertical line below the point. The horizontal dotted line represents the distance between each pair of points. The dotted rectangular box represents the distance constraints. If we consider the distance between the point x_2 and x_3, it is constrained by the rectangular box in x_2. For clear understanding we can say that the vertical rod x_2 could be moved either to the right or to the left of the rectangular box and based on that the placement of x_2 will be varied.

Let P = $\{p_0, p_1, ..., p_n\}$ be a set of points. Some pair of points are separated with distance intervals [*l*,*u*], where l and u denotes the lower and upper bound of the distance between the pair of points p_i and p_j. The distance matrix is divided into two matrices: upper and lower distance matrix. Here for the fixed distance between the pair of points the values in upper and lower bound matrix will be same, whereas for the unknown distance between a pair of points the distance interval would be $[-\infty, \infty]$. Ideally the aim is to find the coordinates of all the points in a metric space.

1.3 Motivation

The motivation of the problem comes from the probe location problem in DNA mapping, where many of the research subjects in biology focus on analyzing the arrangement of nucleotide sequences in DNA. A specific set of nucleotide sequence in a DNA is a called a gene. Each gene is responsible for specific functions in an organism. For genomic studies the probes are synthesized in the laboratory by biologists by taking a complementary sequence of a specific strand of DNA and they are injected in to a chromosome to detect the presence of specific sequence of nucleotides. So these probe locations are to be mapped on a chromosome, given estimates of distance between pair of probes that are obtained from Fluorescence in-situ hybridization(FISH) experiments. Probes are the labeled segment of DNA or RNA used to find the specific sequence of nucleotides. A sample probe used in a DNA probe test is shown below: Identifying the location of specific probes on a chromosome and relative distance between different probes on a chromosome helps the scientists to discover the heritable diseases, as well as diseases and other traits that are common to human beings. Different algorithmic approaches include : simulated annealing [21], branch and bound algorithm [22] and Mumey's algorithm [19] have already been implemented to solve this problem. Some of algorithms are limited to 20 probes or fewer and costs much time.

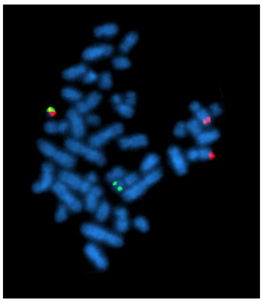


Figure 1.3: DNA probe test [26] showing fluroscently labeled probes

1.4 Thesis Organization

The list below presents the organization of the chapters which makes up this thesis. Also given is a brief description of the topics each chapter deals with.

- Chapter 2 gives a clear background knowledge on the probe location problem in DNA mapping and other extensive approaches to solve the probe location problem. An existing algorithm by Mumey [19] to solve the probe location problem with brief description of each step is explained.
- Chapter 3 we review the existing distance geometry techniques and an algorithm by Crippen and Havel [13] for solving the molecular conformation problem.
- Chapter 4 describes the proposed algorithm and its inner workings giving justification for the chosen approach at each step and also shows the experimental results after applying our algorithm.

- Chapter 5 concludes the work done in this thesis and suggests some possible future research directions.

- Bibliography contains a detailed list of references from which factlets and numbers have been used as a guide for this thesis.

Chapter 2

Probe location problem revisited

One of the chief problem biologists face is to find the position of the probes in a DNA sequence from the distance intervals between each pair of probes. Probes are the small DNA fragments which helps to identify the presence of a gene in a DNA sequence. The distance intervals between a pair of probes are estimated from fluorescence in-situ hybridization(FISH) experiments. This problem is stated as the probe location problem since the distance intervals between some pair of probes are known only with some confidence level and the location of all the probes have to be determined. Identifying the location of specific probes on a chromosome and relative distance between different probes on a chromosome helps the scientists to discover the heritable diseases and other traits that are common to human beings.

2.1 Preliminaries

The following section gives a background details of probes and its synthesis and followed by different algorithmic approaches to solve this probe location problem.

2.1.1 What are probes?

Probes [29] are the fragments of DNA or RNA of variable length(usually 100-1000 bases long) which is radioactively labeled. It can be used in DNA or RNA samples to detect the presence of nucleotide sequences(the DNA target) that are complementary to the sequence in the probe. The probe hybridizes to single-stranded nucleic acid (DNA or RNA) whose base sequence allows probe-target base pairing due to the presence of complementary sequence between the probe and the target. The labeled probe is first denatured(by heating or alkaline conditions such as exposure to sodium hydroxide) into single stranded DNA and then hybridized to the target ssDNA or RNA immobilized on a membrane or in situ.

2.1.2 Probe synthesis

To detect the hybridization of the probe [29] to the target, the probes are labeled with a molecular marker of the fluorescent molecules. Some of the commonly used markers are radioactive isotope of phosphorus or Digoxigenin, which is a non-radioactive, antibody-based marker. DNA sequences or RNA transcripts that have moderate to high sequence similarity to the probe are then detected by visualizing the hybridized probe through different imaging techniques. Normally, either X-ray pictures are taken of the filter, or the filter is placed under UV light.

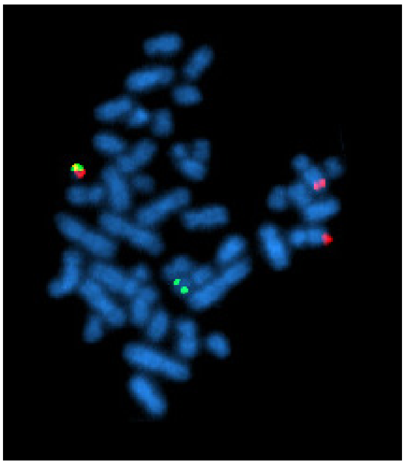


Figure 2.1: A metaphase cell positive for the bcr/abl rearrangement (associated with chronic myelogenous leukemia) using FISH. The chromosomes can be seen in blue and one that is labeled with green and red spots (upper left) is the one where the rearrangement is present found by injecting probes in to the chromosome [28]

Depending on the method, the probe may be synthesized using the phosphoramidite method, or it can be generated and labeled by PCR amplification or cloning (both are older methods). Molecular DNA- or RNA-based probes are used in DNA sequencing which helps scientists to discover the pattern of disease causing genes and other genetic disorders.

2.1.3 FISH experiment

Fluorescence in situ hybridization [28] is a versatile tool that uses fluorescent probes in locating DNA sequences on fixed chromosomes in order to study the structure and function of chromosomes. Non-radioactively labeled fractions of repetitive DNA are used as probes. The fluorescent probes locate and bind with chromosomes with high degree of sequence complementarity. This complementary base pairing allows cells to copy information from one generation to another. It can even find and repair damage to the information stored in the sequences. The information is used in genetic counseling, medicine and species identification. FISH can also be used to detect and localize specific RNA targets (mRNA, lncRNA and miRNA) in cells, circulating tumor cells, and tissue samples. This technique [FISH] allows the analysis of a large series of archival cases much easier to identify the pinpointed chromosome by creating a probe with an artificial chromosomal foundation that will attract similar chromosomes. Consider the following sample DNA sequence: The Fig. 2.2 shows a double stranded DNA. At high temperature or excessive heat double stranded DNA denatures into single strand DNA. Once DNA sequences are denatured, they can be immobilized by using enzymes like nitro cellulose. The details are found in the Fig. 2.3

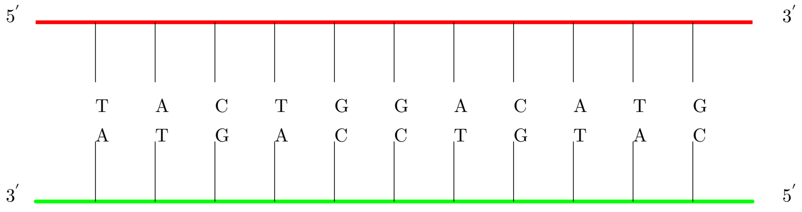


Figure 2.2: A sample double stranded DNA sequence

Later the biologist prepares the fluorescently labeled probes complementary to the

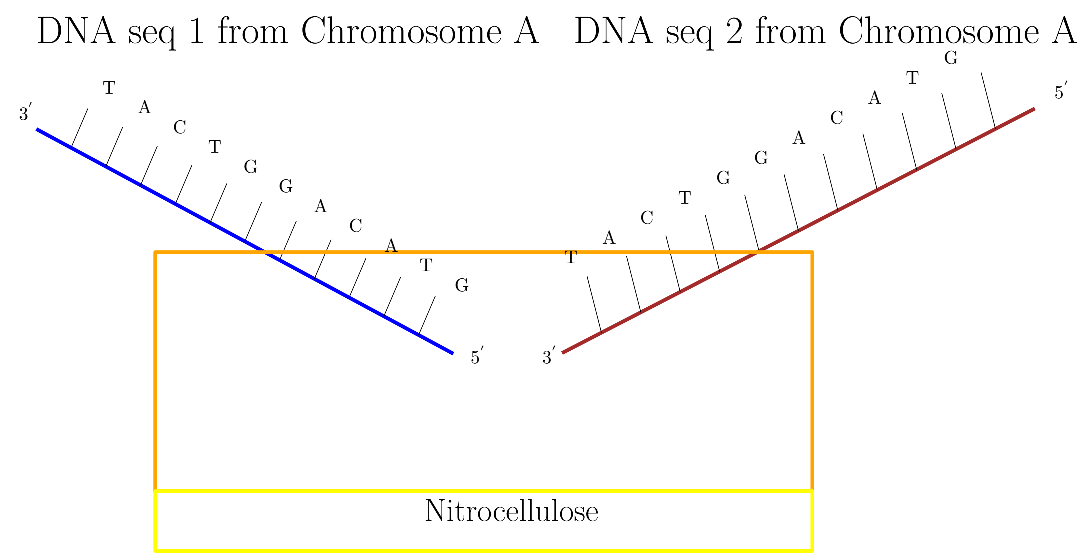

DNA seq 1 from Chromosome A
DNA seq 2 from Chromosome A
3′ T A C T G G A C A T G 5′
3′ T A C T G G A C A T G 5′
Nitrocellulose

Figure 2.3: Denaturing of double stranded DNA into a single stranded DNA sequence

target DNA sequence and they are injected into the chromosome to identify the target DNA sequence.

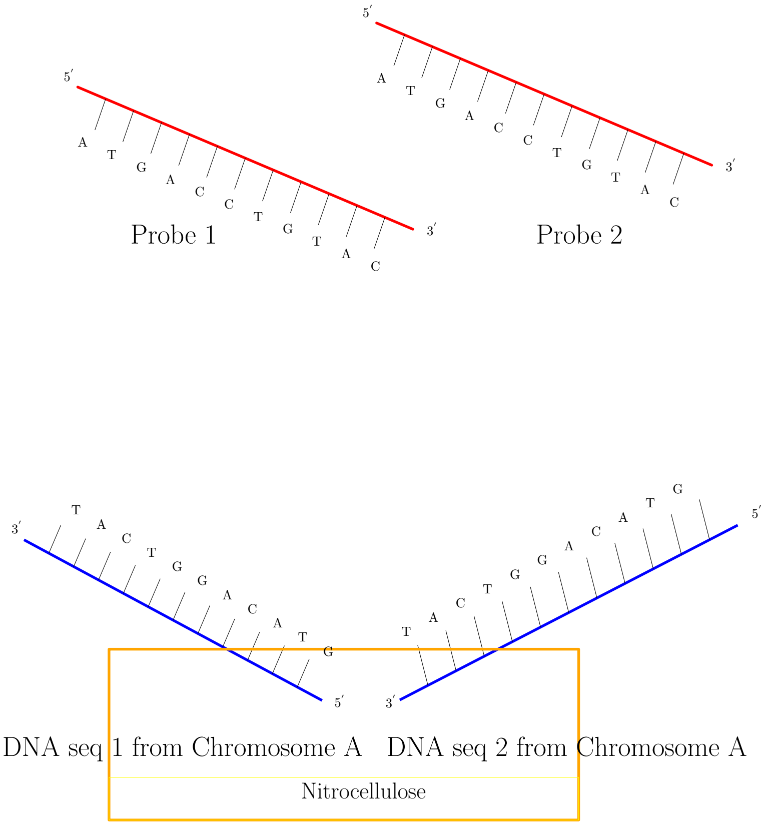

5′ A T G A C C T G T A C 3′
Probe 1
5′ A T G A C C T G T A C 3′
Probe 2
3′ T A C T G G A C A T G 5′
3′ T A C T G G A C A T G 5′
DNA seq 1 from Chromosome A
DNA seq 2 from Chromosome A
Nitrocellulose

Figure 2.4: Probe injection

As soon as the probe is injected, base pairing occurs when the probes find the complimentary strands.

Finally the FISH experiment measures the physical distance (on a microscope slide) between pairs of fluorescently marked probes hybridized to an interphase chromosome [9,8]. For genomic distances of up to about 12 megabases, DNA folding can be described by a random walk model. Statistics can be used to estimate a confidence interval for the genomic distance (in base-pairs) separating two probes given a measured sample of physical distance.

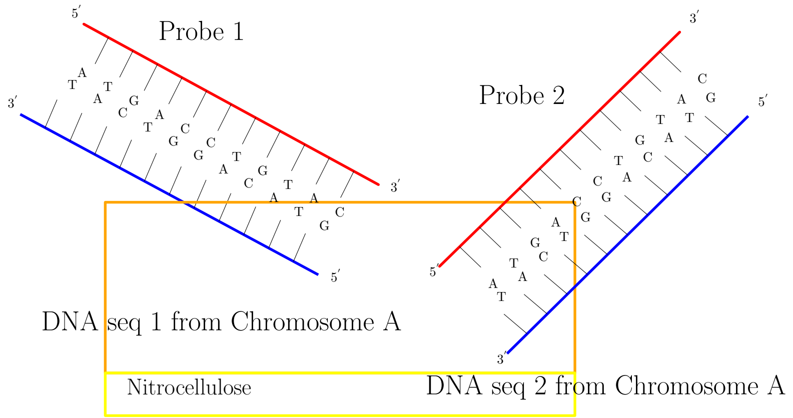


Figure 2.5: DNA Probe Hybridization

2.2 Literature review of approaches to the probe location problem

2.2.1 Seriation Algorithm

Kenneth and Aravindha [4] developed the technique to find the initial and accurate interference of locus(position of gene) order and accurate inter-locus distance and interference obtained using seriation techniques. This analysis requires a matrix of recombination frequency values that can be estimated by pairwise linkage analysis. Recombination frequency is the frequency in which the genetic recombination takes place between two genes in a chromosome. The chief advantage of this method is that the seriation can be performed without the use of computer. In this work a new multi point mapping methodology called seriation algorithm was presented which uses the results of pairwise linkage analysis to determine the locus order and estimate map distances.

Gelfand(1971) presented an algorithm by which collection of n objects could be linearly arranged by knowing the similarity between the pair of objects. The idea of this algorithm is to order the set of points provided. Consider a distance matrix of pairwise recombination values for$n$ where θ_{ij} is the estimated recombination value between the ith and jth locus in the matrix. The pseudo code of the seriation algorithm [4] to find the locus order is mentioned below:

For each locus L_i, i = 1, 2, . . ., n,

1. Write locus L_i.

2. Consider the distance between L_i and the other (n - 1) loci.

Select the locus (L_j) with the smallest distance from L_i and place it to the right of L_i, i.e., L_iL_j.

For the remaining (n - 2) loci in the row referenced by L_i, the following procedure is repeated:

1. Choose the locus L_k from the remaining unplaced loci in that row with the smallest distance to L_i.

2. Compare the distance of L_k with the two loci currently external in the cluster of placed loci, L_I (the locus on the left side) and L_r (the locus on the right side), i.e., $L_I, ..., L_r$.

If $\theta_{kr} > \theta_{kI}$, place L_k to the left of the cluster of currently placed loci, i.e., $L_kL_l, ..., L_r$, or, if $\theta_{kr} < \theta_{kI}$, place L_k to the right of the cluster of currently placed loci, i.e., $L_I, ..., L_rL_k$

Thus by the end of the seriation algorithm we would be able to find the position of the all loci. After obtaining the locus order, the interlocus map distances can be obtained from the pairwise distance measurements by means of least squares.

In this procedure, the ordered distance matrix of recombination frequency values is transformed into map distances by means of mapping functions. From this transformed matrix, estimates of interlocus distances between adjacent loci *(d_i)* can be obtained.

Seriation offers lot of practical advantages. Such as the this method is applicable to an arbitrarily large number of loci. Second the algorithm doesn't use computer rather if desired it can be performed by hand. Simplicity in computation is one main advantage compared to other methods. Some limitations to seriation algorithm is that it requires all possible pairs of distances between loci be available which is difficult as it requires complete set of pairwise comparisons.

2.2.2 Redstone's approach

Redstone [22] chose sum of squares cost function to evaluate different probe orderings and positions. From the developed model they examined the effectiveness of a branch and bound and a local search technique. The branch and bound algorithm searches through a tree of all possible probe orderings. For each probe ordering, the optimal (in the least-squares sense) positions of the probes are determined. This branch and bound approach finds exact solutions upto 18 probes and it will take time when the number of points increases.

Initially the data returned by the FISH process is in terms of physical distance between the probes measured in micrometers. They evaluated the probe placement based on the cost function. The cost measure is to use the sum of squares of the difference between the measured distance between two probes and the distance between the probes in the estimated linear placement of the probes. Considering N be the number of probes, x_i be the position of the probe i, and d_{ij} be the measure distance between the probe i and j. We can write the sum of squares of difference(errors) as

$$Cost(x_1, ..., x_n) = \sum_{i<j}(|x_i - x_j| - d_{ij})^2$$

where d_{ij} is measured. The major advantage of this approach is that they can develop a branch and bound pruning heuristic based on solving for the minimum of this cost function. To construct a branch and bound search the chosen tree representation would be,

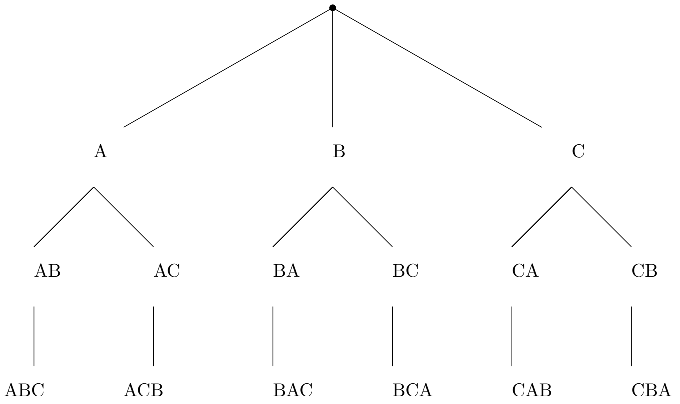


Figure 2.6: At a node, the children are orderings in which each of the unordered probes have been placed to the right of the rightmost ordered probe.

For this approach, in Fig. 2.6. , the ordering of a child of an interior node P will be the ordering of P augmented by a probe placed adjacent to the rightmost ordered probe in P. Later, the Branch and bound algorithm searches through nodes in a tree, pruning a node if its cost is greater than the lowest cost found in a leaf node so far. The basic idea behind this approach is that if the cost for a particular ordering of a probe is minimum then it corresponds to feasible solution.

Due to the exponential nature of the branch and bound algorithm, it doesn't work for large number of probes. However experiments conducted provided good performance on 18 probes or less.

2.3 Mumey's approach to solve the probe location problem

Mumey [19] considered the problem of mapping probes along the genome with the given pairwise distance intervals as input. He called this problem as the probe location problem because the distance intervals are known only with some confidence level, some may be error-prone and it must be identified to find a consistent map. His work was motivated by the goal of mapping probes along a chromosome based on separation intervals estimated from *fluorescence in-situ hybridization (FISH)* experiment. Since the problem is big to solve some previous algorithmic approaches like: a seriation algorithm [4], a simulated annealing approach [21] and a branch and bound algorithm [22] due to their exhaustive nature they are limited to 20 or few number of probes. Here the Mumey's algorithm can solve upto 100 probes at several minutes in a work station. An overview of the Mumey's approach and detailed step by step explanation were discussed in the upcoming sections.

2.3.1 Problem statement

Let P = $\{p_1, p_2, ..., p_n\}$ be the list of probes in a chromosome separated with some distance intervals [*l*,*u*], where l and u denotes the lower and upper bound of the distance between the pair of probes p_i and p_j. The distance matrix is divided into two matrices: upper and lower distance matrix. Here for the fixed distance between the pair of probes the values in upper and lower bound matrix will be same, where as for the unknown distance between a pair of probes the distance interval would be $[-\infty, \infty]$ or other distance intervals returned from *FISH* experiments. The probe location problem is to identify the location of the probes $\{x_1, x_2, ..., x_n\}$ from the given distance intervals such that $\mid x_i - x_j \mid \in [l, u]$.

2.3.2 Overview of Mumey's approach

Given the constraints on a distance measure, $x_i - x_j \in [l, u]$, we have to choose between $x_i - x_j \in [l, u]$ or $x_j - x_i \in [l, u]$. Either choice puts an orientation on an edge connecting the vertices x_i and x_j in a graph whose vertices are the variables, $x_1, x_2, ..., x_n$ and m edges corresponding to the m given constraints. The orientation is from x_i to x_j if x_i is to the left of x_j and from x_j to x_i otherwise. If edge orientations are correctly set then a set of feasible solutions corresponds to the solution of a linear program whose constraints are of the form $x_j - x_i \leq u$ and $x_i - x_j \leq -l$, subject to minimizing the sum $\Sigma_{1 \leq i \leq n} x_i$. In fact, the linear program can be solved by running Bellman-Ford's algorithm for finding shortest paths from a given source vertex in a weighted graph. The weight of an edge is set to be the upper bound u if it is traversed in the same direction as its orientation, else to -l if it is traversed in the opposite direction. There is no feasible solution if the Bellman-Ford algorithm detects a negative cycle.

To set the orientation of the edges correctly a branch-and-bound approach is adopted. A binary orientation tree is constructed where each level of the tree corresponds to an edge and the edges going out of a node corresponds to the left and right orientation. Thus a path from the root to a leaf node gives the orientation of all the edges in the graph. To bound the search, when we reach a given node in the tree, we run Bellman-Ford with the currently available orientations to check for a negative cycle. If there is one, we terminate this branch. The detailed description of each step in Mumey's approach is mentioned in the upcoming sections.

2.3.3 Construction of an Edge Orientation Graph

First step in construction of an edge orientation graph is to set orientation of each edge by choosing one placement. Let x_i and x_j are the position of the probes then:

$$x_j - x_i \in [l, u] \qquad (1)$$

$$x_i - x_j \in [l, u] \qquad (2)$$

where x_i and x_j are the probe positions between the probes i and j. If (1) holds then x_i is to the left of x_j and if (2) holds, then x_j is to the left of x_i. Now the second step would be to assign weights to the edges. For each placement there exists two edges, if x_i is to the left of x_j then there exists two inequalities,

$$l \leq x_j - x_i \leq u$$

$$x_j - x_i \leq u \Longrightarrow (1) \ and \ \ x_i - x_j \leq -l \Longrightarrow (2)$$

Siimilarly, if x_j is to the left of x_i then there exists again two inequalities,

$$l \leq x_i - x_j \leq u$$

$$x_i - x_j \leq u \Longrightarrow (3) \ and \ \ x_j - x_i \leq -l \Longrightarrow (4)$$

Each inequalities mentioned above are represented by an edge, weights will be assigned either upper bound or negative of the lower bound based on the direction of the edge. Similarly we fix edge weights for all pair of probes whose distances are not exact. For the fixed distance between pair of probes the upper and lower bound will be the same and based on the placement of the probe, there will be just one edge connecting the respective probes. Finally an edge orientation graph is constructed.

2.3.4 Finding feasible probe positions

Once all the edge weights are fixed, we plug-in to Bellman-Ford algorithm which finds the shortest path between the source and target vertex. Bellman-Ford algorithm is also used to check for the existence of negative weight cycles. The pseudo-code for Bellman-Ford algorithm [6] as follows:

```
function BellmanFord(list vertices, list edges, vertex source)
   ::distance[],predecessor[]
   // This implementation takes in a graph, represented as
   // lists of vertices and edges, and fills two arrays
   // (distance and predecessor) with shortest-path
   // (less cost/distance/metric) information

   // Step 1: initialize graph
   for each vertex v in vertices:
       if v is source then distance[v] := 0
       else distance[v] := inf
       predecessor[v] := null

   // Step 2: relax edges repeatedly
   for i from 1 to size(vertices)-1:
       for each edge (u, v) in Graph with weight w in edges:
           if distance[u] + w < distance[v]:
               distance[v] := distance[u] + w
               predecessor[v] := u

   // Step 3: check for negative-weight cycles
   for each edge (u, v) in Graph with weight w in edges:
       if distance[u] + w < distance[v]:
           error "Graph contains a negative-weight cycle"
   return distance[ ], predecessor[ ]
```

If there is no negative cycle, the Bellman-Ford algorithm outputs set of feasible solutions $(x_1, x_2, ..., x_n)$ and if there exists a negative weight cycle the program recalculates the edge weights by changing the placement of probes and run the Bellman-Ford algorithm till it finds feasible solutions.

An interesting observation is that the problem of identifying probes in DNA mapping and the molecular conformation problem in distance geometry tends to be similar. Thus the probe location problem can be conveniently cast in the framework of distance geometry which is discussed in the following chapters.

Chapter 3

Distance geometry approach

In the distance geometry framework the coordinates of a set of points can be found if distances between all pairs of points are available. Interesting application in biology is that the experimental techniques were able to measure the distances between pair of atoms of a given molecule and the problem is to identify the three-dimensional conformation of the molecule (i.e. the position of all its atoms). This is called as the Molecular Conformation problem. The main interest is on proteins, because of the three-dimensional conformation which allows to get clues about the function they are able to perform. The particular problem which is mainly focused in this thesis is constructing a probe map(one-dimension) in DNA mapping which helps biologists to locate the specific sequence of nucleotides in DNA. The approximate distance between the probes are estimated from FISH(Fluorescence in-situ hybridization) experiments and are provided in terms of distance intervals $[l,u]$.

3.1 Background and review of the Distance geometry techniques

In this section we discuss some techniques [30] used for solving the distance geometry problem and the software packages developed on basis of these techniques.

Some of them are mentioned in the upcoming section. The first and second technique is concerned with solution to the problems with all exact distances. The third technique tells how to reduce a given problem into smaller subproblems. The least-square minimization algorithm is used to solve the distance geometry problem as a special type of optimization problem.

3.1.1 Cayley-Menger Determinant

Suppose the exact distances between all the points are known, the necessary and sufficient condition for that the distance matrix

$$D(p_O, ..., p_n) = \begin{pmatrix} 0 & d_{01} & ... & d_{0n} \\ d_{10} & 0 & ... & d_{1n} \\ ... & ... & ... & ... \\ d_{n0} & d_{n1} & ... & 0 \end{pmatrix}$$

n+1 points $p_0, p_1, ..., p_n$ is embeddable in euclidean space E^n is given by Cayley-Menger [24] that the CM determinant$(p_1, ...p_n) \geq 0$. The given distance matrix represented by a determinant [24] form:

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & d_{01}^2 & d_{02}^2 & d_{03}^2 \\ 1 & d_{10}^2 & 0 & d_{12}^2 & d_{13}^2 \\ 1 & d_{20}^2 & d_{21}^2 & 0 & d_{23}^2 \\ 1 & d_{30}^2 & d_{31}^2 & d_{32}^2 & 0 \end{vmatrix}$$

According to Gale and Householder, the rank of the Cayley-menger determinant tells exactly the dimension in which the given points could be embedded. For details please refer to [31].

Cayley-Menger matrix can also be used to find the missing distances in the distance matrix. For a query graph with n vertices, the pre-distance matrix $D =$

[D$_{ij}$] is a symmetric matrix such that $D_{ij} = d_{ij}^2$, where d_{ij} is the distance between the vertices (points) i and j of the query graph. The Cayley-Menger matrix, $C =$ *[C$_{ij}$]* is a symmetric $(n + 1) \times (n + 1)$ matrix such $C_{0i} = C_{i0} = 1$ for $0 < \ i \leq n$, *C[0, 0]* $= 0$ and $C_{ij} = D_{ij}$ for $1 < i, j \leq n$ [11].

The vertices of the query graph has a valid linear placement provided the rank of the matrix B is at most 3 (this is a special case of the result that there exists a d-dimensional embedding of the query graph if the rank of B is at most $d + 2$; our claim follows by setting $d = 1$) [31]. It's interesting to check this out for the query graph in Fig. 3.1 on 3 points. The Cayley-Menger matrix B for the above query graph is:

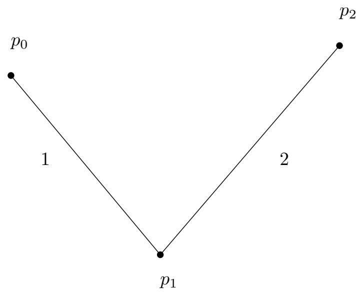


Figure 3.1: A query graph on 3 vertices

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & x^2 \\ 1 & 1 & 0 & 4 \\ 1 & x^2 & 4 & 0 \end{bmatrix}$$

where $x = d_{13}$, the unknown distance between the points p_1 and p_3. By the above result, the 4 × 4 minor, det(B) = 0. This leads to the equation

$$x^4 - 10x^2 + 9 = 0$$

which has two solutions $x = 3$ and $x = 1$, corresponding to the two possible placements of the points p_1, p_2 and p_3. Assuming p_2 is placed to the right of p_1, in one of these placements p_3 is to the right of both p_1 and p_2; or to the left of them both.

3.1.2 Decomposition of Distance matrix

Suppose if we know the exact distance between all the points, then it can be arranged into a matrix, $d = [d_{ij}]$, with d_{ij} corresponds to the distance between i and j. If we have set of points $x_0, x_1, x_2, ..., x_n$ our problem is to find coordinates of those points. The distance constraints can be written as:

$$|x_i - x_j| = d_{ij}, \quad i, j = 1, ...n$$

or equivalently,

$$|x_i|^2 = d_{i0}^2$$

,

$$|x_i - x_j|^2 = d_{ij}^2$$

, by expansion,

$$d_{i0}^2 - d_{ij}^2 + d_{j0}^2 = 2x_i^T x_j, \quad i, j = 1, 2, ..., n$$

Let

$$D_{ij} = (d_{i0}^2 - d_{ij}^2 + d_{j0}^2)/2,$$

we can then define a matrix

$$D = [D_{ij}]$$

. Let X be an $n \times 3$ and

$$X = [x_1^T; .....; x_n^T;]$$

we then have

$$D = XX^T$$

For a solution to exist the matrix D must be of rank 3. Therefore, we can make a singular value decomposition for D to obtain

$$D = U\sigma U^T$$

Where U is an $n \times 3$ orthogonal matrix and σ be the eigen value diagonal matrix with diagonal elements $\sigma_1, \sigma_2, \sigma_3$ being three non-zero singular values of D. A solution for

$$D = XX^T$$

can be obtained with

$$X = U\sigma^{(1/2)}$$

Here the singular value decomposition can be done in $O(n^3)$ time. Therefore the solution to the distance geometry problem can be obtained in polynomial time if all the exact distances are given. More details can be found in [7].

3.1.3 Graph Reduction

Considering the points as nodes and distances as edges, the distance geometry problem can be described in a distance graph and the solution would be to realize the graph in an Euclidean space. This problem is therefore called graph embedding. The edge weights in the graph are sparse so there won't be a unique embedding. In other words we can say that there are more than one ways to position the points so that the distance constraints can all be satisfied. To conclude there are infinitely many ways to embed the graph, so the graph is called flexible or rigid.

The rigidity of the graph is important for the study of the distance geometry problem. A necessary condition that a graph has a unique embedding is that it must be rigid. Another conditions for the graph to have a unique embedding is that it doesn't have partial reflections. For three-dimensional embedding a graph has to be four-connected. These conditions can be used to find graphs or subgraphs that have unique embeddings. The embedding problem for a given distance graph can solved by decomposing the graph into such sub-graphs. Once the solution for subgraphs are found, they are combined to form a solution for the whole graph. For mode details refer to [14].

3.1.3.1 ABBIE

Hendrickson developed the ABBIE software package [14] to obtain the three dimensional embedding of the molecular structure by giving pairwise distance measurements as inputs. The method in this software is based on graph reduction. The problem of determining a set of points in space is divided into smaller subset of points whose relative locations can be determined uniquely. The basic idea is to use divide-and-conquer rule and dividing the problem into smaller problems to ultimately find a unique solution. The solutions found for the subgraphs can then be combined into a solution for the whole graph. This is done by using the method of graph reduction discussed in section 2.9.2 in which the given distance graph is decomposed recursively into sub-graphs. These sub-graphs are then solved by minimizing a least-square error function. The sub-graphs consist of subset of points whose location can be determined. Once such a subset is positioned its points can be treated as a rigid body. There are several advantages of this algorithm. To begin with even if there is insufficient information, the method will identify and solve unique sub problems. Secondly the solution to the sub-graphs can be as important and of interest. Third this method determines if there is sufficient information for the problem. Finally erroneous data can be identified by the inability to solve a sub problem.

3.1.4 Least-Squares Formulation

The distance geometry problem can be formulated as a global least-squares problem. Considering the problem with exact distances, the problem can be defined with set of equality constraints,

$$|x_i - x_j| = d_{ij}, \ \ \textit{(i,j)} \in S$$

Where S may or may not be the whole set of distance pairs. In order to solve this class of problems, we measure the following relative errors between the calculated and given distance,

$$\frac{|x_i - x_j|^2 - d_{ij}^2}{d_{ij}^2}, \ \ \textit{(i,j)} \in S$$

and collected to obtain an error function,

$$f(x_1, ...x_n) = \sum_{i,j \in S} [\frac{|x_i - x_j|^2 - d_{ij}^2}{d_{ij}^2}]^2$$

Here we see that if the distance constraints are satisfied then the error function is equal to zero. Similarly, for problems with bounds on the distance we have,

$$l_{ij} \leq |x_i - x_j| \leq u_{ij}, \ \ \textit{(i,j)} \in S$$

Then an error function can be constructed as,

$$f(x_1, ...x_n) = \sum_{i,j \in S} min^2[\frac{|x_i - x_j|^2 - d_{ij}^2}{d_{ij}^2}, 0] + max^2[\frac{|x_i - x_j|^2 - d_{ij}^2}{d_{ij}^2}, 0]$$

It is not easy to verify that if all the inequality constraints are satisfied , the error function is equal to zero.

Given the above error function *f*, it is easy to see that a set of coordinates $x_1, x_2, ...x_n$ is a solution to the distance geometry problem if and only if it is the global minimizer of f with the global minimum equal to zero. Therefore, the distance geometry problem can be formulated as an optimization problem.

$$min_{x_1,..,x_n} f(x_1, ..., x_n)$$

More details could be found in [7].

3.1.4.1 DGSOL

DGSOL software package is developed by More and Wu [16][17] used global smoothing and continuation for solving molecular distance geometry problem. This particular method does not require all distance or bounds to be available. This method considers the least-squares formulation of the distance geomtry problem.

The least-squares problem may have many local minimizers. In order to locate the global minimizer, the global smoothing and continuation method first transforms the least-squares function into a set of gradually deformed but smoother or easier functions with fewer local minimizers. This method is applied to some small to medium-sized test problems with around 200 points or atoms. The result showed that the method was able to find the global minimizer of the least-squares function with a very high probability.

One of the advantages of this method is that it does not need all the distances or bounds. Since they use smaller number of terms the cost for solving the distance geometry problem is cheaper. The method is more practical in the sense since only sparse set of distance bounds are available. The bound smoothing technique may be helpful for providing some additional distance data, but they are not so reliable in general.

3.1.5 Alternating Projection Algorithm

Alternating Projection algorithm developed by Glunt et al [12] is used for solving the distance geometry problem with a given set of bounds on distances. The main idea used is to first get the set of distances from the distance bounds. Then this distance geometry problem is solved by minimizing an error function(optimization).The program is done until a solution is found, otherwise the violated constraints are used to adjust the distance and algorithm is repeated for new set of distances.

The bounds on all the distances should be available for the program to execute. In every iteration a least-squares problem is solved, which requires large amount of computation. For example, if a Newton's algorithm is used, the total cost can be as much as $O(n)^3$ and if n is large and the problem needs to be solved many times, it can be too expensive to use. Therefore spectral gradient algorithm which is much cheaper is used in the alternating projecting algorithm instead.

3.2 Crippen and Havel's algorithm

Crippen and Havel [13], pioneered the work in distance geometry for molecular conformation. Their algorithm is used for solving the molecular conformation problem arising in NMR spectroscopy and protein structure determination. There are three main stages in the algorithm. The first stage takes the input distance bounds and converts into distance limits(bound smoothing). In the second step, a random value is chosen between the limits and fix the distance for all pairs of probes, final stage would be to retrieve the coordinates from the distance constraints(least-squares optimization). The brief description of all the stages is described in the upcoming section.

3.2.1 Bound Smoothing

Due to imprecision in measurements, the distance between the probes are specified as pairs of upper and lower bounds. In order to identify the coordinates of the points, the distance bounds has to be tightened into limits and this step to convert the given bounds into limits is called bound smoothing. These limits that satisfies triangle inequality are called triangle inequality limits. A modified version of Floyd's algorithm presented by Dress and Havel [9] is used to convert the bounds into limits that satisfy the triangle inequality. If there is a triangle inequality violation $l_{ij} > u_{ij}$ found, then the program exits the current process and then repeatedly iterates over to find out the limits.

Some geometric rules are used in the bound smoothing. For given three points *i,j,* and *k*, let the lower and upper bounds be denoted as l_{ij}, u_{ij}, l_{jk} and u_{jk}. Then the lower and upper bounds for the distance between points i and k must agree with the following rules [9],

$$l_{ik} = max(l_{ik}, l_{ij} - u_{jk}, l_{jk} - u_{ij})$$

$$u_{ik} = min(u_{ik}, u_{ij} + u_{jk})$$

Other rules can also be derived similarly for the distance bounds for more than three points [9].

3.2.2 Metrization

The next step would be to convert those distance limits into distances, this process is known as Metrization. Here in this process we take one of the distances and sets it to some random number between its lower and upper limits. Later we set its lower and upper limits to this number and recompute the triangle inequality limits using these modified limits as the upper and lower bounds. Repeating this process for each distance will result in set of lower and upper triangle inequality limits that are equal to each other and lies within the original limits and these distances will be the desired matrix of distances that satisfies both the triangle inequality and original limits. For details please refer to [13],[9].

3.2.3 Embedding

Final step in the Crippen and Havel's algorithm is to find the coordinates from the distance matrix. This consists of the following steps(Havel et al.,[13]):

(i) The distance of each point from the center-of-mass is calculated, to avoid over emphasizing any set of points, according to

$$D_{i0}^2 = \frac{1}{N}\sum_{j=1}^{N} D_{ij}^2 + \frac{1}{N^2}\sum_{j=2}^{N}\sum_{k=1}^{j-1} D_{jk}^2$$

where D_{i0} is the distance of the point i from the origin and D_{ij} is the distance between points i and j.

(ii) The elements a_{ij} of the metric matrix A are computed from the distance of points from the origin,

$$a_{ij} = \frac{1}{2}(D_{i0}^2 + D_{j0}^2 - D_{ij}^2)$$

(iii) Let W be the diagonal matrix of weights $W = diag(w_1, ..., w_n)$, then in our case the weights are all assumed to be 1, then the calculation of the B matrix is

$$B = W\,A\,W$$

(iv) If B matrix is positive semi-definite then according to Gale and Householder equation [31], the final coordinate matrix X is obtained by diagonalizing the B matrix,

$$B = \sigma\, L^2\, \sigma'$$

and

$$L^2 = [\lambda_1^2, \lambda_2^2, ...\lambda_r^2, 0, ...0]$$

Finally,

$$X = \sigma \sqrt{L}$$

where L is the diagonal matrix of latent roots of the B matrix, and σ is the diagonalized eigen vectors of the corresponding latent roots.

Chapter 4

Distance geometry based probe location

Building on the material of the previous chapters, in this chapter we described the main contribution of this thesis. We propose a new algorithm DGPL for the probe location problem based on the distance geometry approach. The subsequent sections discuss in this order: how synthetic data is generated for the DGPL algorithm, followed by a formal description of the algorithm.

4.1 Preliminaries

Synthetic data is generated based on an adversarial model. An adversary is a system which knows the placement of points in the respective dimension. The idea underlying the notion of an adversary is to check the placement of points generated by the algorithm against this. Once the user inputs the number of points for which the coordinates have to be found, the adversary creates a distance matrix with a valid layout for those points. Then based on the number of unknown distances provided by the user, adversary chooses random set of point pairs for which the distances are not known. The adversary creates a lower and upper bound distance matrix which assigns $[-\infty, \infty]$ as the distances for points with unknown distances, i.e. in the upper bound distance matrix for the corresponding points with unknown distance it will be ∞, and -∞ in the lower bound distance matrix. For the pair of points with exact distances known, the values in the upper and lower bound distance matrix will be the same(exact distance). For instance, three points with one unknown distance, a sample input upper and lower bound distance matrix generated by the adversary are:

$$U(p_0, p_1, p_2) = \begin{pmatrix} 0 & 60 & \infty \\ 60 & 0 & 3 \\ \infty & 3 & 0 \end{pmatrix}$$

$$L(p_0, p_1, p_2) = \begin{pmatrix} 0 & 60 & -\infty \\ 60 & 0 & 3 \\ -\infty & 3 & 0 \end{pmatrix}$$

Thus the adversary creates these upper and lower bound distance matrices which will be the input for the DGPL.

4.2 Algorithm description

The DGPL algorithm works in three phases:

a) Phase 1: Preparation - Adversary first creates a valid layout based on the number of points provided by the user and sets up the distance intervals to $[-\infty, \infty]$ for unknown distances and finally creates a lower and upper bound distance matrix.

b) Phase 2: Processing - uses the upper and lower bound distance matrices from the phase 1 and convert those distances into the coordinates.

c) Phase 3: Embedding - produces a visualization of the points plotted in a graph and verify it against the initial layout generated by the algorithm.

A flowchart of DGPL algorithm is shown in the upcoming section.

4.2.1 Flowchart

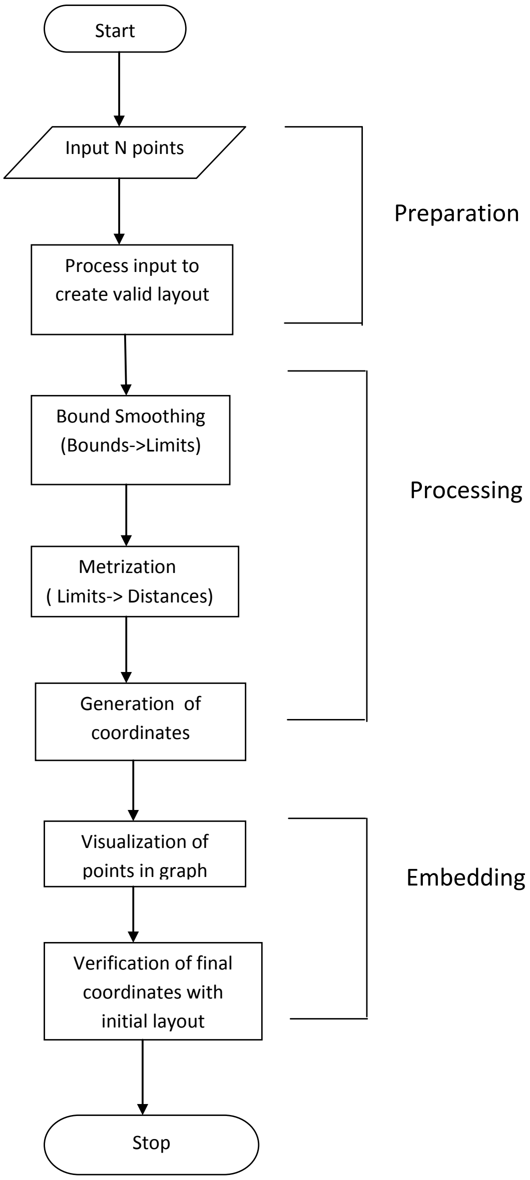

Start
Input N points
Process input to create valid layout
Preparation
Bound Smoothing (Bounds->Limits)
Metrization (Limits-> Distances)
Generation of coordinates
Processing
Visualization of points in graph
Verification of final coordinates with initial layout
Embedding
Stop

Figure 4.1: Flowchart depicting step by step process of DGPL

4.2.2 DGPL Algorithm

Input data: i. The total number of points used. ii. The number of unknown distances to embed the points.

Output: Coordinates of the given points.

Process:

//phase 1

Step 1: Create a random valid layout with the fixed number of points such as $\{p_0, p_1, ..., p_n\}$ where n is the number of points.

Step 2: Based on the number of unknown distances entered by the user, assign $[-\infty, \infty]$ as the corresponding values in upper and lower bounds distance matrix.

// phase 2

Step 3: A modified version of the Floyd's shortest path algorithm[13] is applied to convert the bounds into limits.

Step 4: Choose a random number between upper and lower limit and assign the same value as the upper and lower limit for each pair of unknown distance intervals and apply step 2 to get a fixed distance between all the points. Repeat the process till all the distances are fixed.

Step 5: Calculate matrix B =$[b_{ij}]$, where b_{ij} is given by [31]:

$$b_{ij} = (d_{io}^2 + d_{jo}^2 - d_{ij}^2)/2$$

where d_{ij} is the distance between points i and j, o is the starting point(origin) p_0. i, j lies between 1,.....,n-1 and n is the total number of points.

Step 6: Computer the eigenvalue decomposition of the B matrix and the product of the largest eigenvalue with its corresponding normalized eigen vectors will give the values of the coordinates for all the points in one-dimension.

//phase 3

Step 7: The coordinates are plotted in a graph and the final coordinates are verified with initial layout generated by the program.

4.2.3 Details

1. A sample input to the program as follows: number of points: 4, number of unknown distances: 1 and dimension to embed the points: 1.

2. A random valid layout is created in one-dimension, the points are:

$$p_0 = 0, p_1 = 59, p_2 = 48, p_3 = 74.$$

3. The input upper and lower bound distance matrices are:

$$LB = \begin{pmatrix} 0 & 59 & 48 & 74 \\ 59 & 0 & 11 & -\infty \\ 48 & 11 & 0 & 26 \\ 74 & -\infty & 26 & 0 \end{pmatrix}, UB = \begin{pmatrix} 0 & 59 & 48 & 74 \\ 59 & 0 & 11 & \infty \\ 48 & 11 & 0 & 26 \\ 74 & \infty & 26 & 0 \end{pmatrix}$$

where the distance d_{13} between the points p_1 and p_3 is contained in the interval $[-\infty, \infty]$.

4. Shortest path limits obtained through bound smoothing from the given upper and lower distance bounds are:

$$LL = \begin{pmatrix} 0 & 59 & 48 & 74 \\ 59 & 0 & 11 & 15 \\ 48 & 11 & 0 & 26 \\ 74 & 15 & 26 & 0 \end{pmatrix}, UL = \begin{pmatrix} 0 & 59 & 48 & 74 \\ 59 & 0 & 11 & 37 \\ 48 & 11 & 0 & 26 \\ 74 & 37 & 26 & 0 \end{pmatrix}$$

Here *UL,LL* are the lower and upper limits.

5. The distance limits are then metrized, as discussed in section 2.2, to fixed distances as:

$$D = \begin{pmatrix} 0 & 59 & 48 & 74 \\ 59 & 0 & 11 & 35 \\ 48 & 11 & 0 & 26 \\ 74 & 15 & 26 & 0 \end{pmatrix}.$$

6. The B matrix with the point p_0 as origin is found to be

$$B = \begin{pmatrix} 3481 & 2832 & 3866 \\ 2832 & 2304 & 3552 \\ 3866 & 3552 & 5476 \end{pmatrix}.$$

7. With the eigenvalue decomposition of B matrix, the product of the square root of largest eigenvalue with its corresponding normalized eigen vectors gives the coordinate matrix:

$$X = \begin{pmatrix} 58.677 \\ 48.828 \\ 73.671 \end{pmatrix}.$$

8. These coordinates are verified with the layout generated initially by the program at step 2. Finally a graph is drawn to plot the points finally obtained by the DGPL algorithm. A graph with the points embedded are shown here:

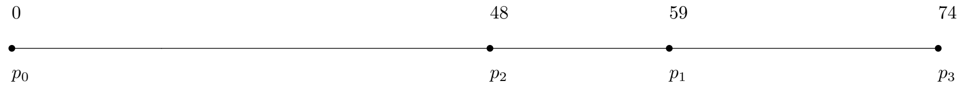


Figure 4.2: Final embedding of the four input points

4.2.4 Three dimensional embedding

The DGPL program could also be used to identify the three-dimensional conformation of a molecule. The structure of a protein could be determined experimentally through NMR spectroscopy or X-ray crystallography or theoretically through potential energy minimization or molecular dynamics simulation. More specifically the problem considered here is the determination of a structure of a protein given the distance between some pair of atoms in the protein and the unknown distances could be represented in terms of distance intervals. The known distances are obtained with our knowledge of certain bond lengths and bond angles or estimated from NMR experiments. This problem is generally called as a molecular conformation problem.

4.2.4.1 Protein Data Bank

The Protein data Bank (PDB) [2] was first conceived at Brookhaven National Laboratories in 1971. The archive initially contained only seven structures of macro-molecules. The advent of technologies such as nuclear magnetic resonance imaging and X-ray crystallography for structure determination in the early eighties quickly increased the number of available structures. A huge boost to the bank's accessibility and exponential growth was provided by a change in the attitude towards sharing the data and all above all the advent of the Internet. Proteins are gigantic sequential molecules of smaller recurring molecules. They are made up of amino acids which are linked by peptide bonds to form polymers in the polypeptide chains. A protein may consist of one or more polypeptide chains.

All known protein structures are stored in the repository in PDB format. The PDB format contains data for each atom in the structure, viz. its type and (x,y,z) coordinates, residue number and the type of the residue. Each atom takes up a single line in the PDB file. For instance, an entry in the pdb file for the globin FERRIC APLYSIA LIMACINA which has PDB code 2FAL is as follows:

ATOM 493 CA ARG A 66 56.089 1.103 41.810 .....(1)

Similarly if there are two other atoms:

ATOM 117 CA ASP A 14 14.969 37.123 6.770 .....(2)

ATOM 109 CA LEU A 13 15.162 35.549 X .....(3)

In short, a pdb file is a digitized version of the actual protein chemical. The above (1) indicates that there is a carbon atom with the value of x, y and z coordinates (56.089,1.103,41.810). Moreover, the 'CA' shows that is the central C_α atom of a residue, namely residue 66 of type 'ARG' from chain A. The value 493 is a unique atom identifier within the file. Similarly other atom(2) of residue 14 with it coordinates were shown and the atom(3) with the missing Z coordinates represented as X could also be identified through the DGPL algorithm, however the values of the coordinate matrix obtained from our algorithm is translated, further refinement has to be done to get accurate coordinates. The current steps followed to get approximate estimates of the missing coordinates in three-dimension is described in the upcoming section.

4.2.4.2 Generation of Coordinates in three-dimension

The steps applied to generate the coordinates in three-dimension from the DGPL is slightly different from the generation of the coordinates in one-dimension. The input to the DGPL would be the upper and lower distance intervals for each pair of atoms and the embedding would be three in this case. The steps described in the algorithm section 2.6 till step 5 are same. Finally the eigenvalue decomposition of the B matrix is done and the product of the three largest eigenvalues with their corresponding normalized eigenvectors will give the coordinates of all the points in three-dimension. A sample plot showing the three-dimensional embedding generated by DGPL is shown below:

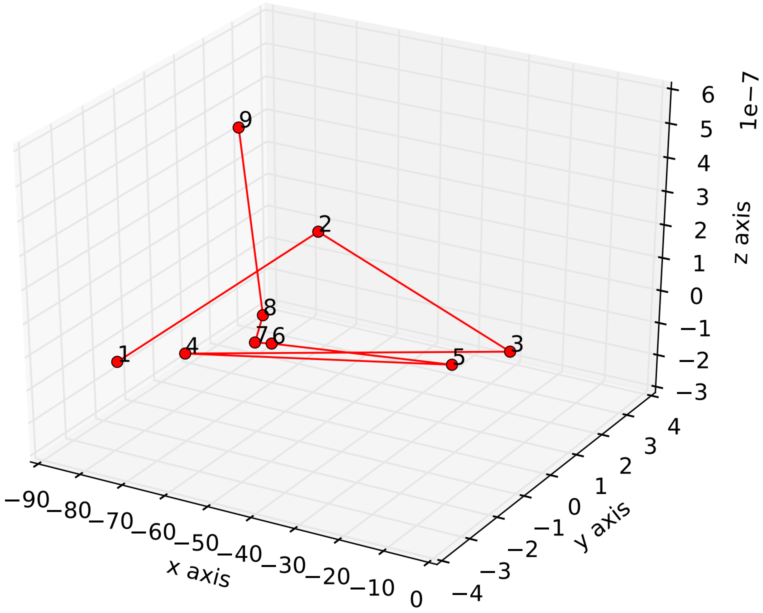


Figure 4.3: A sample three-dimensional embedding of 9 points generated by the DGPL program

4.3 Experminental results

We implemented both the DGPL and also the Mumey's approach discussed in the previous sections in Python 2.7 on a computer with the following configuration: Intel(R) Xeon(R) CPU, X7460@2.66GHz OS: Ubuntu 12.04.5, Architecture:i686. Some of the mathematical packages used were numpy.linAlg which helps in calculating eigen value decomposition and also for solving linear equations, matplotlib.pyplot is used to plot the final coordinates obtained from the DGPL and Mumey's program into a graph with respective dimensions. In this section we first present the computational results that were obtained using the algorithmic approach described in thesis. This is followed by a discussion of the results along with some conclusions that can be drawn from them. Finally, we look into some potential limitations of both the programs, and any possible future work that can emanate from it.

Consider a small example of four points p_0, p_1, p_2 and p_3 which has a valid layout in one-dimension generated by the program. Therefore the coordinates are $p_0 = 0$, $p_1 = 36$, $p_2 = 65$ and $p_3 = 85$.

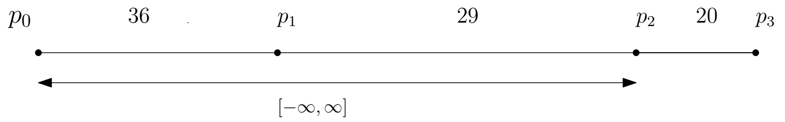


Figure 4.4: One-dimensional embedding of four points with one unknown distance

Our aim is to estimate the coordinates of all the points through Mumey's and DGPL program by randomly fixing distance intervals $[-\infty, \infty]$ for unknown distances. Here in this case the program sets one unknown distance d_{02} between p_0 and p_2 is set to in the range $[-\infty, \infty]$. Then as per Mumey's approach discussed at section 2.2, based on the orientation assigned between the points p_0 and p_2 it chooses upper bound or negative of the lower bound as the fixed distance and runs bellman-ford algorithm to detect the negative cycle and if not it finds feasible solutions.

DGPL program also runs with the same input for the given four points, as discussed in section 4.2 the program initially tightens the distance bounds for unknown distance pairs into distance limits, then it does metrization by randomly choosing values between the limits and it calculates the B matrix and does eigen value decomposition according to Gale and Householder equation [31] to find out the Coordinate matrix. The final layout obtained from both the algorithms were shown below:

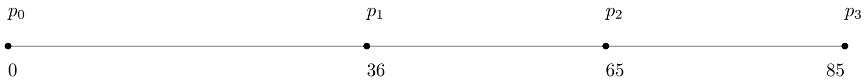
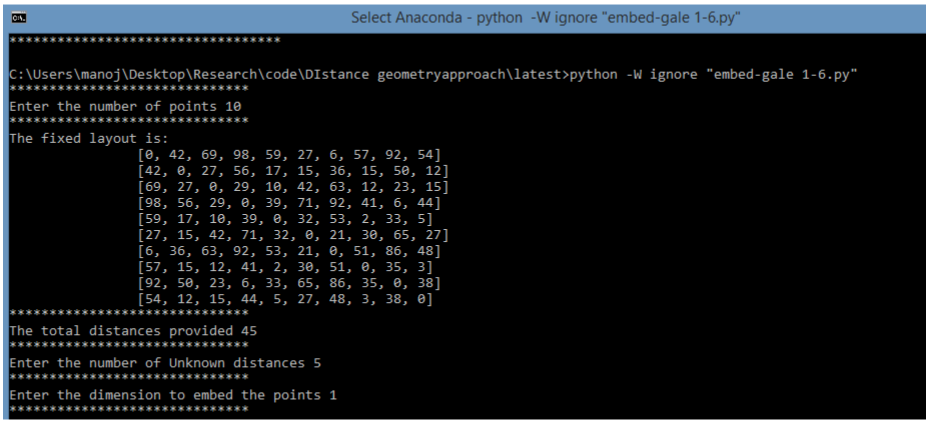


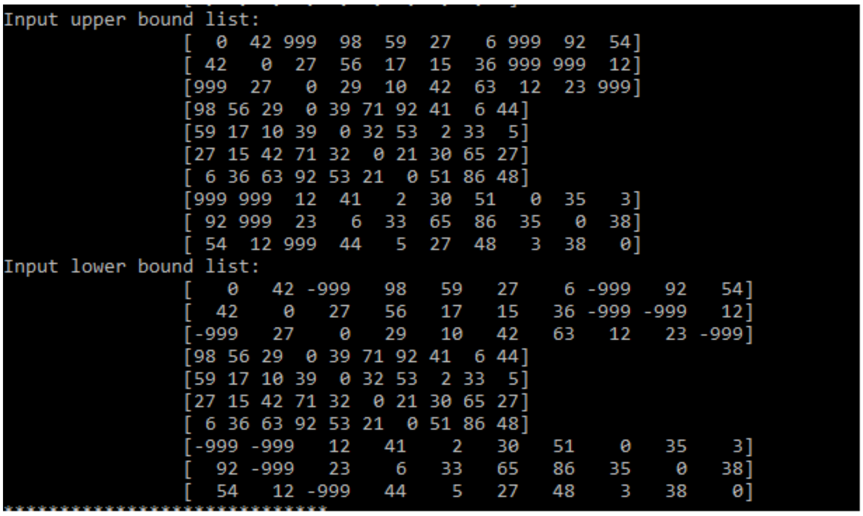
Figure 4.5: One-dimensional embedding of four points with coordinates

Some sample screenshots for large number of points by increasing number of unknown distances are shown in the upcoming sections.



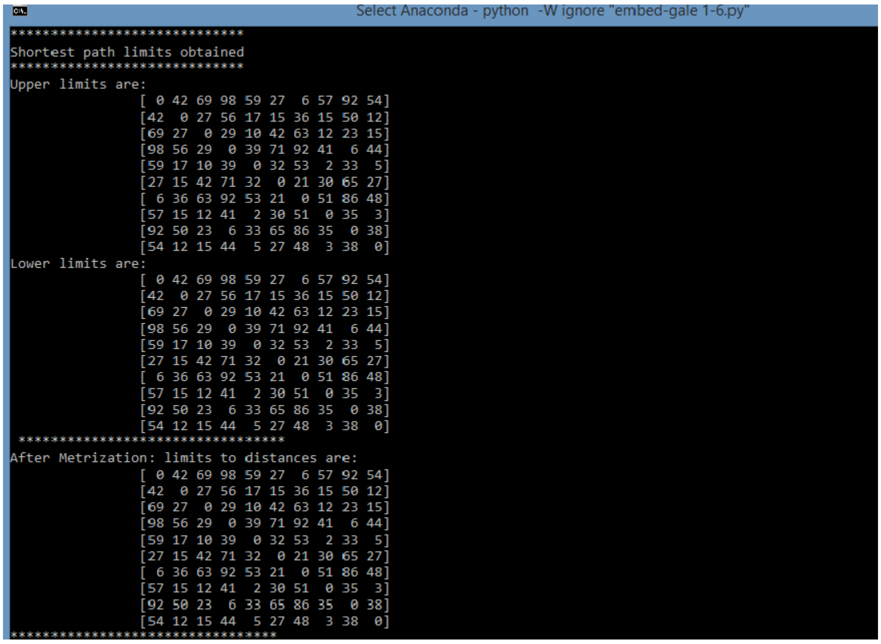
```
**********************************

C:\Users\manoj\Desktop\Research\code\DIstance geometryapproach\latest>python -W ignore "embed-gale 1-6.py"
******************************
Enter the number of points 10
******************************
The fixed layout is:
                [0, 42, 69, 98, 59, 27, 6, 57, 92, 54]
                [42, 0, 27, 56, 17, 15, 36, 15, 50, 12]
                [69, 27, 0, 29, 10, 42, 63, 12, 23, 15]
                [98, 56, 29, 0, 39, 71, 92, 41, 6, 44]
                [59, 17, 10, 39, 0, 32, 53, 2, 33, 5]
                [27, 15, 42, 71, 32, 0, 21, 30, 65, 27]
                [6, 36, 63, 92, 53, 21, 0, 51, 86, 48]
                [57, 15, 12, 41, 2, 30, 51, 0, 35, 3]
                [92, 50, 23, 6, 33, 65, 86, 35, 0, 38]
                [54, 12, 15, 44, 5, 27, 48, 3, 38, 0]
******************************
The total distances provided 45
******************************
Enter the number of Unknown distances 5
******************************
Enter the dimension to embed the points 1
******************************
```

Figure 4.6: Screenshot of the DGPL program input



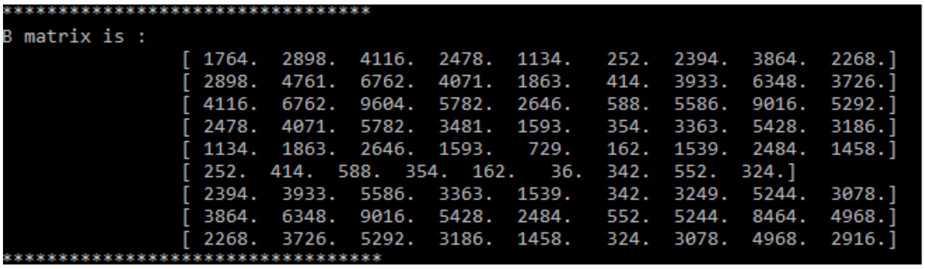
```
Input upper bound list:
                [  0  42 999  98  59  27   6 999  92  54]
                [ 42   0  27  56  17  15  36 999 999  12]
                [999  27   0  29  10  42  63  12  23 999]
                [98 56 29  0 39 71 92 41  6 44]
                [59 17 10 39  0 32 53  2 33  5]
                [27 15 42 71 32  0 21 30 65 27]
                [ 6 36 63 92 53 21  0 51 86 48]
                [999 999  12  41   2  30  51   0  35   3]
                [ 92 999  23   6  33  65  86  35   0  38]
                [ 54  12 999  44   5  27  48   3  38   0]
Input lower bound list:
                [   0   42 -999   98   59   27    6 -999   92   54]
                [  42    0   27   56   17   15   36 -999 -999   12]
                [-999   27    0   29   10   42   63   12   23 -999]
                [98 56 29  0 39 71 92 41  6 44]
                [59 17 10 39  0 32 53  2 33  5]
                [27 15 42 71 32  0 21 30 65 27]
                [ 6 36 63 92 53 21  0 51 86 48]
                [-999 -999   12   41    2   30   51    0   35    3]
                [  92 -999   23    6   33   65   86   35    0   38]
                [  54   12 -999   44    5   27   48    3   38    0]
******************************
```

Figure 4.7: Screenshot of the DGPL program upper and lower bound inputs for ten points with five unknown distances



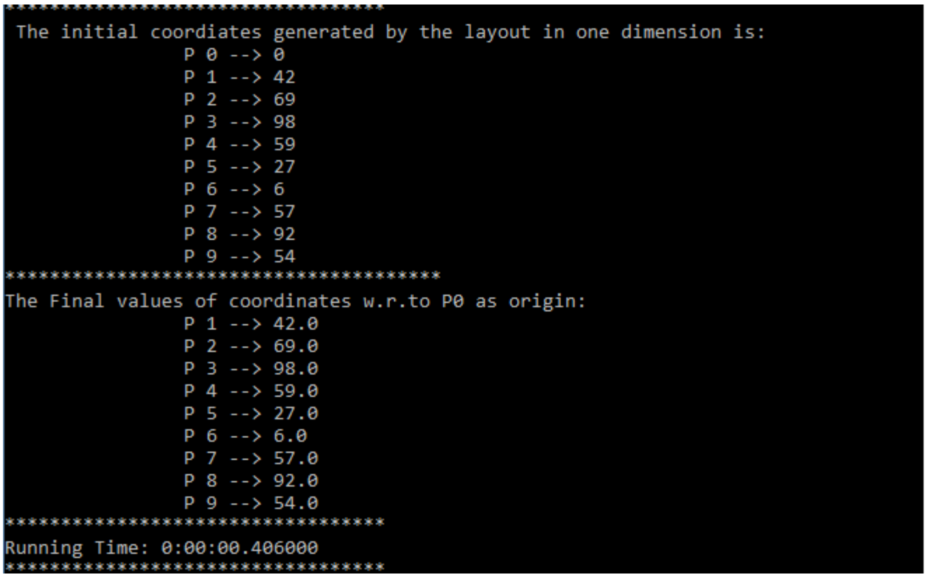
```
****************************
Shortest path limits obtained
****************************
Upper limits are:
                [ 0 42 69 98 59 27  6 57 92 54]
                [42  0 27 56 17 15 36 15 50 12]
                [69 27  0 29 10 42 63 12 23 15]
                [98 56 29  0 39 71 92 41  6 44]
                [59 17 10 39  0 32 53  2 33  5]
                [27 15 42 71 32  0 21 30 65 27]
                [ 6 36 63 92 53 21  0 51 86 48]
                [57 15 12 41  2 30 51  0 35  3]
                [92 50 23  6 33 65 86 35  0 38]
                [54 12 15 44  5 27 48  3 38  0]
Lower limits are:
                [ 0 42 69 98 59 27  6 57 92 54]
                [42  0 27 56 17 15 36 15 50 12]
                [69 27  0 29 10 42 63 12 23 15]
                [98 56 29  0 39 71 92 41  6 44]
                [59 17 10 39  0 32 53  2 33  5]
                [27 15 42 71 32  0 21 30 65 27]
                [ 6 36 63 92 53 21  0 51 86 48]
                [57 15 12 41  2 30 51  0 35  3]
                [92 50 23  6 33 65 86 35  0 38]
                [54 12 15 44  5 27 48  3 38  0]
 *********************************
After Metrization: limits to distances are:
                [ 0 42 69 98 59 27  6 57 92 54]
                [42  0 27 56 17 15 36 15 50 12]
                [69 27  0 29 10 42 63 12 23 15]
                [98 56 29  0 39 71 92 41  6 44]
                [59 17 10 39  0 32 53  2 33  5]
                [27 15 42 71 32  0 21 30 65 27]
                [ 6 36 63 92 53 21  0 51 86 48]
                [57 15 12 41  2 30 51  0 35  3]
                [92 50 23  6 33 65 86 35  0 38]
                [54 12 15 44  5 27 48  3 38  0]
*********************************
```

Figure 4.8: Calculation of triangle limits and setting distances based on these limits



```
**********************************
B matrix is :
                [ 1764.  2898.  4116.  2478.  1134.   252.  2394.  3864.  2268.]
                [ 2898.  4761.  6762.  4071.  1863.   414.  3933.  6348.  3726.]
                [ 4116.  6762.  9604.  5782.  2646.   588.  5586.  9016.  5292.]
                [ 2478.  4071.  5782.  3481.  1593.   354.  3363.  5428.  3186.]
                [ 1134.  1863.  2646.  1593.   729.   162.  1539.  2484.  1458.]
                [ 252.   414.   588.   354.   162.    36.   342.   552.   324.]
                [ 2394.  3933.  5586.  3363.  1539.   342.  3249.  5244.  3078.]
                [ 3864.  6348.  9016.  5428.  2484.   552.  5244.  8464.  4968.]
                [ 2268.  3726.  5292.  3186.  1458.   324.  3078.  4968.  2916.]
***********************************
```

Figure 4.9: Calculation of a B matrix



```
************************************
 The initial coordiates generated by the layout in one dimension is:
                P 0 --> 0
                P 1 --> 42
                P 2 --> 69
                P 3 --> 98
                P 4 --> 59
                P 5 --> 27
                P 6 --> 6
                P 7 --> 57
                P 8 --> 92
                P 9 --> 54
****************************************
The Final values of coordinates w.r.to P0 as origin:
                P 1 --> 42.0
                P 2 --> 69.0
                P 3 --> 98.0
                P 4 --> 59.0
                P 5 --> 27.0
                P 6 --> 6.0
                P 7 --> 57.0
                P 8 --> 92.0
                P 9 --> 54.0
************************************
Running Time: 0:00:00.406000
************************************
```

Figure 4.10: Final output generated by the program

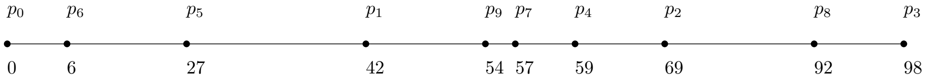


Figure 4.11: Final embedding of the given ten points in a line

However the results of the experiments by increasing the number of unknown distances for fixed points in both Mumey's and DGPL algorithm approach is shown in the graph below.

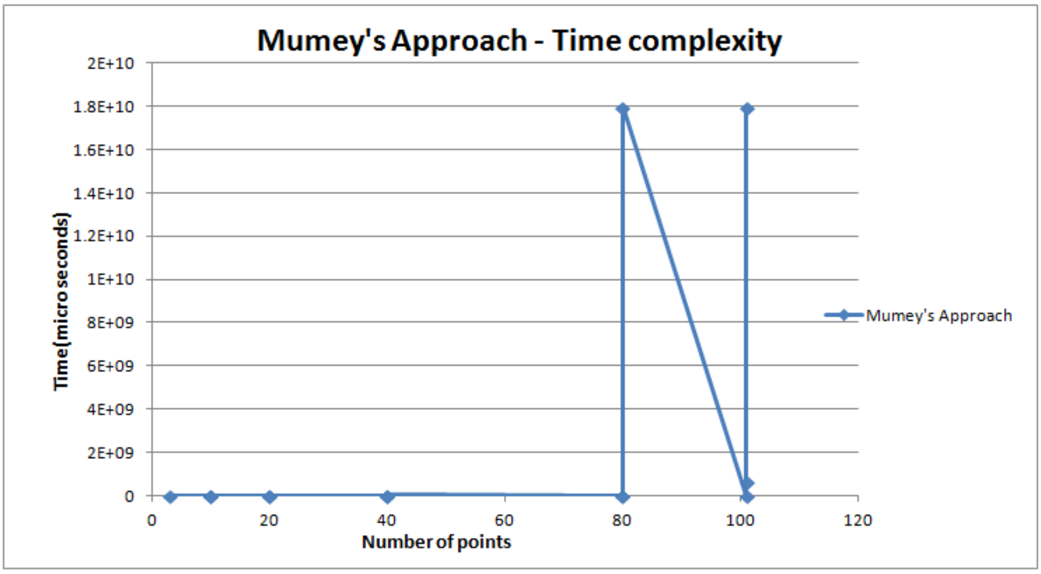


Figure 4.12: Graph depicting the run times of Mumey's approach

The above chart represents the run times of Mumey's approach, where the *x-axis* denotes the number of points and *y-axis* denotes the time in microseconds. The blue line in the graph shows the time-variation for fixed number of points and increasing number of unknown distances. For example in the graph, for 80 points with five unknown distances Mumey's approach takes approximately 10.60 seconds and as we increase the number of unknown distance say 15, time taken by Mumey's approach is more than five hours. For further details please refer to Fig. 4.12 and Table. 4.1.

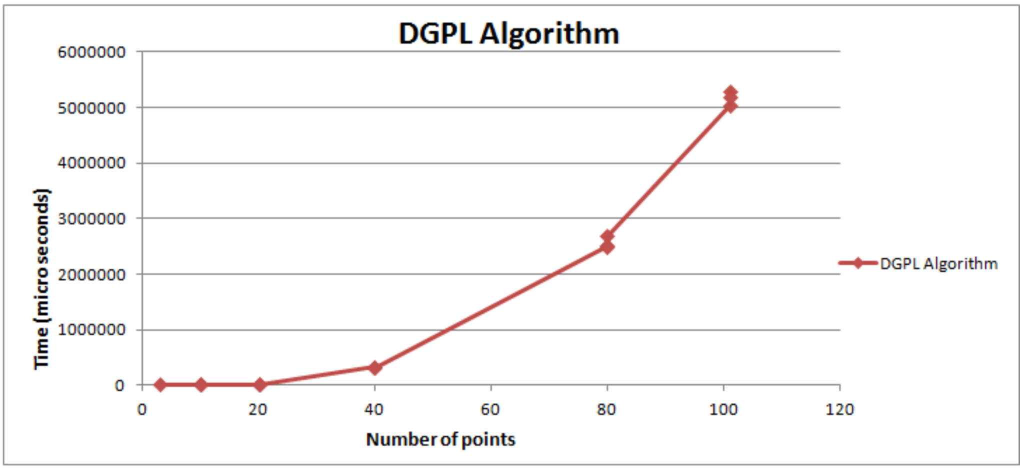


Figure 4.13: Graph depicting the run times of DGPL algorithm

The above chart represents the time complexity of DGPL algorithm, where the *x- axis* denotes the number of points and *y-axis* denotes the time-complexity. The red line in the graph shows the time-variation for fixed number of points and increasing number of unknown distances. For example in the graph, for same 80 points with five unknown distances DGPL algorithm takes approximately 2.5 seconds and as we increase the number of unknown distances say 15, time taken by Embed algorithm is almost the same. Here in this graph the increase in the run times is just due to the increase in number of points. For further details please refer to Fig. 4.13 and Table. 4.1.

No.of points	No.of unknown distances	Mumey's approach running time (hrs:mins:secs)	DGPL algorithm running time (hrs:mins:secs)
3	1	0:00:00.000184	0:00:00.001514
10	2	0:00:00.001339	0:00:00.006938
10	5	0:00:00.024560	0:00:00.006816
10	8	0:00:00.060520	0:00:00.017163
20	2	0:00:00.001369	0:00:00.007464
20	5	0:00:00.001336	0:00:00.007743
20	10	0:00:01.164363	0:00:00.007436
40	5	0:00:00.947250	0:00:00.328563
40	8	0:00:07.369925	0:00:00.315001
40	10	0:00:30.857658	0:00:00.312674
80	5	0:00:10.609233	0:00:02.503798
80	10	0:06:15.443501	0:00:02.496285
80	15	5:00:00.000000+	0:00:02.687672
101	5	0:00:14.256343	0:00:05.020695
101	10	0:10:32.299084	0:00:05.282747
101	15	5:00:00.000000+	0:00:05.192594

Table 4.1: Performance comparison of Mumey's and DGPL algorithm

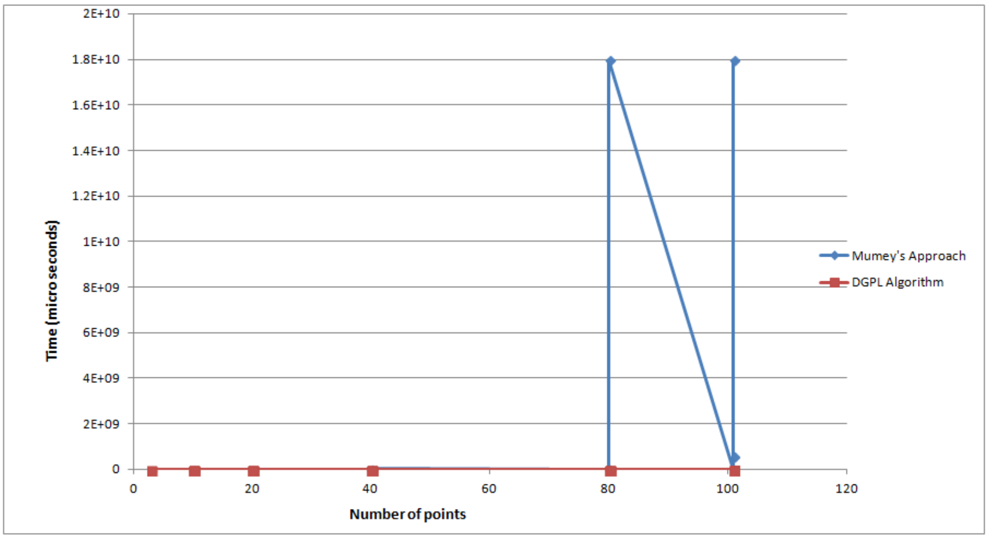


Figure 4.14: Time complexity graph Mumey's vs DGPL algorithm - Increasing number of unknown distances between fixed number of points

Combining the two charts presented in the previous section is the chart mentioned above which compares the time complexity of the Mumey's approach with the DGPL algorithm. Predictably enough, the above chart Fig. 4.14. shows that the Mumey's approach takes longer time when the number of unknown distances increases as shown in the graph and the table. Each of these algorithms were run on point sets of different sizes, up to 101 points.

Clearly, the DGPL algorithm is consistently the fastest; as we can see from the graph irrespective of the number of unknown distances in the fixed number of points the algorithm runs in linear time where as the Mumey's approach doesn't run consistently based on the fixed number of points due to the presence of negative cycle while running Bellmann-Ford algorithm with the distance matrix chosen and as the number of unknown distances increases, the running time also increases exponentially. After each detection of negative cycle, a new distance matrix will be plugged into Bellmann-Ford to find the feasible solutions and it's costlier to keep track of distances chosen between each unknown distance pair.

Chapter 5

Conclusions

This thesis contributes towards the goal of constructing probes in DNA mapping which is useful for genomic studies and also helps scientists to discover the heritable diseases and other traits that are common to human beings. To this end we focused on using a novel approach by reducing the time-complexity of embedding probes in one-dimension and adopting a distance geometry approach to arrive at better results for the probe location problem.

Chapter 2 discussed about the probe location problem in DNA mapping. Fundamental ideas and techniques for probe synthesis and mapping the probes in a chromosome were briefly described. The existing algorithm by Mumey [19] for finding the location of the probes in a chromosome is explained in detail and it's implemented to compare the results with our proposed algorithm discussed in the upcoming chapter. Apart from that, a literature review of other algorithms used before Mumey were also mentioned in this chapter with a detailed description.

The research work started in chapter 3 with a detailed description of the distance geometry techniques and softwares used to solve the Molecular conformation problem. EMBED algorithm which is the fundamental and chief distance geometry technique developed by Crippen and Havel [13] for embedding points in a space is explained in depth which formed the basis for our proposed algorithm. Main ideas and basic building blocks were defined in order to construct a solid understanding of the embedding, mathematical methods and other distance geometry approaches.

The main contribution of this thesis is mentioned in the chapter 4 and the proposed algorithm to solve the probe location problem is explained in detail. The core of this work took the form of experiments. A representative set of results from these experiments have been presented in the same chapter. Both Mumey's algorithm and our algorithm were tested with different set of points starting from 3 to 101 number of points with different set of unknown distances. DGPL algorithm can efficiently find the missing coordinates of the given probes than the Mumey's approach.

The interesting conclusions were drawn from experiments in the previous chapter. For example, with 15 unknown distances in 80 points DGPL algorithm shows superior performance than the Mumey's approach, since the Mumey's approach has to keep track of the distances chosen between each pair of unknown distance, it is costlier compared to the DGPL algorithm. The results have been shown in a graph with the coordinates being plotted in one-dimension. This final graph could be used to validate the correctness of the placement of points by verifying all coordinate values against the initial layout generated by the program. Some of the things to look forward in the future is described in the upcoming section.

5.1 Future work

Further work can be done on several fronts. So far from the experiments conducted and previous techniques used to solve the point placement problem there is no reference to the minimum number of distances to be known for fixed number of points to find the exact placement. So the fundamental question remains open about the minimum number of distances to be known to get the exact coordinates from the DGPL program. It would be interesting if there is any future work in response to this problem. Apart from that, the DGPL program could be applied in higher dimensions for identifying three-dimensional structure of a protein which helps in the identification of their biological function. Although the current implementation of the DGPL program works for finding coordinates in three-dimension, further upgrade of the DGPL program could be done to refine those coordinates to get exact values, which I am sure it will be in the near future.

Bibliography

[1] Roger Baker and Kenneth Kuttler. *Linear algebra with applications.* World Scientific, 2014.

[2] Frances C Bernstein, Thomas F Koetzle, Graheme JB Williams, Edgar F Meyer, Michael D Brice, John R Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. The protein data bank: a computer-based archival file for macromolecular structures. *Archives of biochemistry and biophysics*, 185(2):584–591, 1978.

[3] LM Blumenthal. Theory and applications of distance geometry. *Distance Geometry*, 1970.

[4] Kenneth H Buetow and Aravinda Chakravarti. Multipoint gene mapping using seriation. i. general methods. *American journal of human genetics*, 41(2):180, 1987.

[5] Francis YL Chin, Henry CM Leung, Wing-Kin Sung, and Siu-Ming Yiu. The point placement problem on a line–improved bounds for pairwise distance queries. In *Algorithms in Bioinformatics*, pages 372–382. Springer, 2007.

[6] Thomas H Cormen. *Introduction to algorithms.* MIT press, 2009.

[7] Gordon M Crippen, Timothy F Havel, et al. *Distance geometry and molecular conformation*, volume 74. Research Studies Press Somerset, England, 1988.

[8] Peter Damaschke. Point placement on the line by distance data. *Discrete Applied Mathematics*, 127(1):53–62, 2003.

[9] Andreas WM Dress and Timothy F Havel. Shortest-path problems and molecular conformation. *Discrete Applied Mathematics*, 19(1):129–144, 1988.

[10] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[11] Ioannis Z Emiris and Ioannis Psarros. Counting euclidean embeddings of rigid graphs. *arXiv preprint arXiv:1402.1484*, 2014.

[12] W Glunt, TL Hayden, and M Raydan. Molecular conformations from distance matrices. *Journal of Computational Chemistry*, 14(1):114–120, 1993.

[13] Timothy F Havel. Distance geometry: Theory, algorithms, and chemical applications. *Encyclopedia of Computational Chemistry*, 120, 1998.

[14] B.A. Hendrickson. *The Molecule Problem: Determining Conformation from Pairwise Distances*. PhD thesis, Cornell University, 1990.

[15] Thérèse E Malliavin, Antonio Mucherino, and Michael Nilges. Distance geometry in structural biology: new perspectives. In *Distance Geometry*, pages 329–350. Springer, 2013.

[16] Jorge J Moré and Zhijun Wu. Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7(3):814–836, 1997.

[17] Wu Zhijun More, Jorge. Distance geometry optimization for protein structures. *Journal of Global Optimization*, 15(3):219–234, 1999.

[18] Asish Mukhopadhyay, PijusKumar Sarker, and KishoreKumarVaradharajan Kannan. Randomized versus deterministic point placement algorithms: An experimental study. In Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Marina L. Gavrilova, Ana Maria Alves Coutinho Rocha, Carmelo Torre, David Taniar, and Bernady O. Apduhan, editors, *Computational Science and Its Applications – ICCSA 2015*, volume 9156 of *Lecture Notes in Computer Science*, pages 185–196. Springer International Publishing, 2015.

[19] Brendan Mumey. Probe location in the presence of errors: a problem from dna mapping. *Discrete Applied Mathematics*, 104(1):187–201, 2000.

[20] William R Newell, Richard Mott, Stephan Beck, and Hans Lehrach. Construction of genetic maps using distance geometry. *Genomics*, 30(1):59–70, 1995.

[21] B Pinkerton. Results of a simulated annealing algorithm for fish mapping. *Communicated by Dr. Larry Ruzzo, University of Washington*, 1993.

[22] Joshua Redstone and Walter L Ruzzo. Algorithms for a simple point placement problem. In *Algorithms and Complexity*, pages 32–43. Springer, 2000.

[23] Kaushik Roy, Satish Chandra Panigrahi, and Asish Mukhopadhyay. Multiple alignment of structures using center of proteins. In *Bioinformatics Research and Applications*, pages 284–296. Springer, 2015.

[24] Manfred J Sippl and Harold A Scheraga. Cayley-menger coordinates. *Proceedings of the National Academy of Sciences*, 83(8):2283–2287, 1986.

[25] Zachary Voller and Zhijun Wu. Distance geometry methods for protein structure determination. In *Distance Geometry*, pages 139–159. Springer, 2013.

[26] Wikipedia. Cytogenetics — wikipedia, the free encyclopedia, 2015. [Online; accessed 10-August-2015].

[27] Wikipedia. Distance geometry — wikipedia, the free encyclopedia, 2015. [Online; accessed 10-August-2015].

[28] Wikipedia. Fluorescence in situ hybridization — wikipedia, the free encyclopedia, 2015. [Online; accessed 9-August-2015].

[29] Wikipedia. Hybridization probe — wikipedia, the free encyclopedia, 2015. [Online; accessed 10-August-2015].

[30] Jeong-Mi Yoon, Yash Gad, and Zhijun Wu. Mathematical modeling of protein structure using distance geometry. *Department of Computational & Applied Mathematics, Rice University*, 2000.

[31] Gale Young and Alston S Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3(1):19–22, 1938.

APPENDIX

Eigenvalues and Eigenvectors: An eigenvector of a matrix is a vector that does not change its direction under the associated linear transformation. In other wordsif v is a vector that is not zero, then it is an eigenvector of a square matrix A if Av is a scalar multiple of v. This condition could be written as the equation:

$$AV = \lambda V$$

where λ is the eigenvalue associated with the eigenvector v.

Modified version of Floyds algorithm by Crippen and Havel:

This finds an all-pair shortest path in a directed graph. The algorithm is given below:

```
procedure Floyd( Natom,Lower,Upper )
       for k from 1 to Natom do
             for i from 1 to Natom - 1 do
                  for j from i + 1 to Natom do
      comment: Path lengths in left-hand network.
                     if Upper[i,j] > Upper[i,k] +  Upper[k,j] then
                             Upper[i,j] :=Upper[i,k] + Upper[k,j];
      comment: Path lengths from left to right-hand network.
                    if Lower[i,j] < Lower[i,k] - Upper[k,j] then
                            Lower[i,j] :=Lower[i,k] - Upper[k,j];
                   else
```

```
                    if Lower[i,j] < Lower[j,k] - Upper[k,i] then
                            Lower[i,j] :=Lower[j,k] -  Upper[k,i];
     comment: Check for triangle inequality violations.
                   if Lower[i,j] > Upper[i,j] then
                            exit( bad bounds );
     endfor endfor endfor
endproc
```

Positive semi-definite:

If all the eigenvalues of a matrix are positive then it is termed as positive semi-definite matrix.

Rank of a Matrix:

It is the dimension of the vector space generated by its columns or rows.

VITA AUCTORIS

Kishore Kumar Varadharajan Kannan was born in Salem, India in the year 1991. He passed Bachelor of Technology in Information technology from St.Joseph's college of engineering, India in the year 2012. He is currently a candidate for the Masters degree in Computer Science at the University of Windsor, Ontario and to graduate in Fall 2015.