18.417 Introduction to Computational Molecular Biology	
Lecture 23: December 2, 1999	Scribe: Michael Baker
Lecturer: Jon Kleinberg	Editor: Ishan Anand

Distance Geometry: Reconstruction from Distance Data

Introduction

Biological and chemical sciences apply a number of techniques which, as a matter of data collection, report a set of distances between a set of markers or points in space. A field of study concerning the reconstructing of the positions of points for these markers is distance geometry.

Two classes of reconstruction problems are reviewed in this lecture.

- *Unlabelled distance data*: we are given a list of distances between points without indications of which distances correspond to which pair of markers.
- *Labelled distance data*: we are given distances as well as the endpoints corresponding to each distance.

In both problems, we must reconstruct the geometery of the original structure. In this lecture we will look at both kinds of problems and discuss some of the theory that arises in each case.

Unlabelled distance data

Example 1: Mapping with restriction enzymes.

When DNA is digested by a restriction enzyme, the sizes of the resulting segments can be obtained using gel electrophoresis. Thus, if there are n restriction sites on the DNA strand, we ideally get $\binom{n}{2}$ distance values between the sites, one for each pair of points. The problem is then to reconstruct the restriction sites from this unlabelled data. This is an example of a one-dimensional distance geometry problem.

For instance, say the resulting distances are $\{1, 3, 4, 6, 9, 10\}$. They yield the points in Figure 23.1 as a solution.

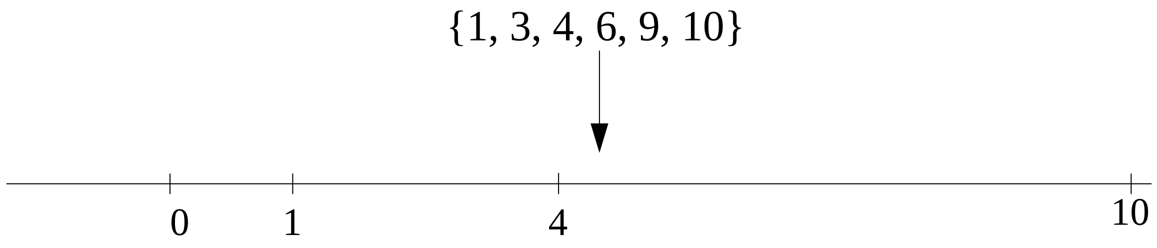

Figure 23.1: An unlabelled distance geometry problem and a possible solution.

Example 2: Phase recovery in X-ray crystallography.

By crystallizing a protein structures and applying x-ray diffraction methods the electron density function over the protein's structure can be determined. The intent here is to fully map out the density, ρ, where, $\rho : \mathbb{R}^3 \to \mathbb{R}$. Roughly, this technique yields the Fourier expansion of ρ by measuring the power of the diffracted x-rays at specific angles from the crystalline body.

Since each term of the Fourier expansion of this density function has both real and imaginary components, the diffraction angles do not allow for a complete interpretation of the coefficient. This complex coefficient is written $re^{2\pi i\phi}$. The value of r is computed from the angles of diffraction, the argument, ϕ, however is not recovered. This is known as the "phase problem of x-ray crystallography".

This problem can be transformed (Patterson, 1936) into an a set of unlabelled distances. Phase information can then be recovered by finding a solution to this three-dimensional problem instance. One noted complication applies to this particular form of the problem, however. Since the crystallization process produces a lattice of proteins, the angles produced by the x-ray diffractions repeat in three-space. By restricting our discussion to one-dimensional problems we can understand the basic issues with these problems.

Questions related to one-dimensional reconstruction problems

One question relevant to our discussion of the reconstruction problem is whether a given set of distances can be made to produce more than one solution. Indeed, by translating or reflecting the markers we can generate a solution with identical distance properties. These simple cases are illustrated in Figure 23.2.

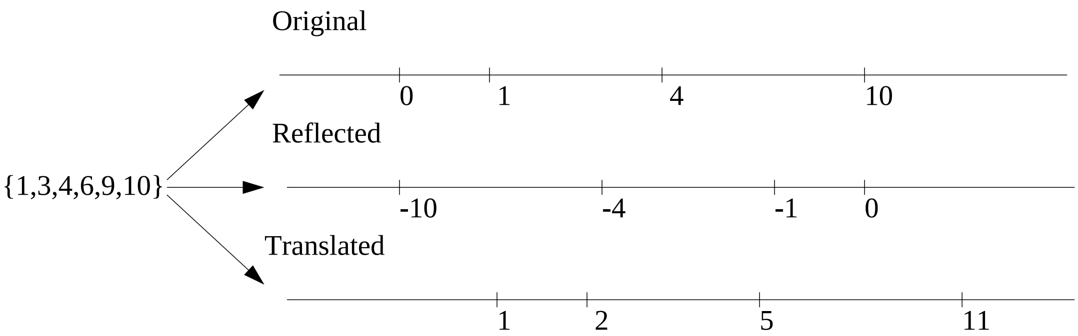

Figure 23.2: Three possible solutions indicate that arrangements are non-unique in general.

This is not, however, the only way to generate additional solutions given one set of distance data. In fact we construct two distinct sets of distance data which yield the same set of markers on the real-axis as a solution. One simple example is, $A = \{0, 1, 4, 10, 12, 17\}, B = \{0, 1, 8, 11, 13, 17\}$

To formalize this problem we define sets of points on the real-axis and related operations to manipulate these points:

Definition 1 *Given two sets A, B of points on the real line, we define the following sets:*

- $\triangle A$ *is the multiset of all distances between points in* A*.*
- $-A = \{-a_i | a_i \in A\}$*. This is the reflection of* A *around the origin.*
- $A + B = \{a_i + b_j | a_i \in A, b_j \in B\}$*.*
- $A - B = A + (-B)$*.*

Further, we must define what it means to have identical distance solutions given differing sets of distance data:

Definition 2 *Given two sets A, B of points on the real line, we say that A and B are homometric, if, $\triangle A = \triangle B$.*

We are now interested in showing that in general, we can construct homometric distance sets given any multiset of distances on the real axis. The following theorem indicates one possible way of accomplisihing this.

Theorem 1 *For any subsets A, B of the real line, $A+B$ and $A-B$ are homometric.*

Proof: Consider a distance d between points in $A+B$. Distance d is $(a+b)-(a'+b')$, for $a, a' \in A, b, b' \in B$. This can be rewritten, $(a-b')-(a'-b)$, which is a distance between points in $A-B$. Thus every element of $\triangle(A+B)$ is equal to some element of $\triangle(A-B)$ and vice versa, so $\triangle(A+B) = \triangle(A-B)$ as desired. ■

To sketch the intuitive reason for this setup, we can imagine which points will be in the set of $A+B$. Essentially, for every point in A, we are tacking on a copy of B. By applying this operation we are duplicating the distance data values without generating any distances that will not also be present in $A-B$. This approach will not produce all possible homometric pairs, however, since we have not attempted to introduce reflections of one data set around the other. Therefore, there exist sets of points on the real-axis whose multiset of distances are not constructed by the above method.

A stronger statement concerning constructing homometric sets is possible. A further abstraction of our point set will be necessary. Namely, we will write the the set of points as a generating function for a polynomial. In this way, we can use the polynomial notation to perform the relevant manipulations on the points of interest.

Definition 3 *For $A = \{a_1, a_2, a_3, \ldots, a_n\}$, where $a_i \in \mathbb{R}$, let $A(x)$ represent the polynomial $x^{a_1} + x^{a_2} + x^{a_3} + \ldots + x^{a_n}$.*

Operations on the points in A are now applied as follows:

- *Translation by a'* is accomplished by computing $(A+a')(x) = x^{a'}A(x)$.
- *Reflection of A to $-A$* is accomplished by computing $(-A)(x) = (A)(x^{-1})$.
- *To add points in set B to those in A* we compute $(A+B)(x) = (A)(x) \cdot (B)(x)$.

- *To compute the multiset* $\triangle A$ write $(\triangle A)(x) = (A)(x) \cdot (A)(x^{-1})$. Note that this generates all pairs, however, it also generates those with negative distances. For the purpose of the work here, this detail will not affect our results.

With these additional definitions, we can prove Theorem 1 more directly.

Proof: Theorem 1

First, expressing $(\triangle(A+B))(x)$:

$$\begin{aligned}(\triangle(A+B))(x) &= (A+B)(x) \cdot (A+B)(x^{-1}) \\ &= A(x) \cdot B(x) \cdot A(x^{-1}) \cdot B(x^{-1})\end{aligned}$$

We now wish to show that expressing $(\triangle(A+B))(x)$ yields the same expression:

$$\begin{aligned}(\triangle(A-B))(x) &= (A-B)(x) \cdot (A-B)(x^{-1}) \\ &= A(x) \cdot B(x^{-1}) \cdot A(x^{-1}) \cdot B(x)\end{aligned}$$

Both expressions are equal because these products commute. It follows that $\triangle(A+B)$ and $\triangle(A-B)$ are equal. ■

With these additional tools we are ready to express all homometric sets (Rosenblatt and Seymour, 1986).

Theorem 2 *(Rosenblatt and Seymour) The sets* $A, B \subset \mathbb{R}$ *are homometric iff* $\exists P$ *and* $Q \subset \mathbb{P}$*, and* u *and* $v \subset \mathbb{R}$ *such that,*

$$\begin{aligned}A(x) &= x^u \cdot P(x) \cdot Q(x) \\ B(x) &= x^v \cdot P(x) \cdot Q(x^{-1}) \ .\end{aligned}$$

In our previous approach to finding homometric sets, we did not consider all polynomials. Namely, we considered only polynomials with coefficients equal to one. Here, $P(x)$ and $Q(x)$ encode arbitrary polynomials whose coefficients can be greater than one. When $P(x)$ and $Q(x)$ encode points on the real axis, as per the previous definitions, $A(x)$ and $B(x)$ are translated and flipped from $Q(x)$ and $P(x)$. This gives us an indication of how these expressions encode all of the homometric representation of a set of points on the real axis.

The representation we are using to describe the distances between the points on the real axis lends itself to an algorithm for determining the points themselves. If we factor our representation of $A(x)$ into its monomials and write $A(x)$ as the product of these, we can determine the sign of each term. This allows us to determine the polynomial $\triangle A(x)$. Since polynomial factorization is a difficult problem this approach does not scale well when a large number of distances wish to be resolved to points.

Agorithms for one-dimensional unlabeled data

We want the candidate algorithm to accept a list of interpoint distances and return the points on the real axis which generate this list. As discussed above there are brute-force approaches to this which apply in general, however, these require a large number of operations since polynomial factorization is a difficult problem. Algorithms which perform better than polynomial factorization are in order. Although polynomial factorization has not been shown to be in NP, there is no presently known algorith which computes the factors in polynomial time.

It should be noted that just because an algorithm requires and exponential number of steps in the size of the input does not mean it is not of interest. If it can be made efficient for a small range of input sizes then it can still be of use.

A simple algorithm. If we were aware of the labelling of the input data-set, we could reconstruct the orignal points easily. We can think of this as trying to determine the labelling on an adjacency matrix, whose entries represent the unlabeled distance data-set. Labelling the indices of the matrix allows us to chose one point in the set of markers, fix its location, and proceed to its neighbors of known distance, placing them on the axis systematically. The adjacency matrix for our data-set, $\{0, 1, 3, 4, 6, 9, 10\}$ is,

$$\mathbf{X} = \begin{bmatrix} 0 & 1 & 4 & 10 \\ 1 & 0 & 3 & 9 \\ 4 & 3 & 0 & 6 \\ 10 & 9 & 6 & 0 \end{bmatrix}$$

As this algorithm requires us to match up $\binom{n}{2}$ labels with $\binom{n}{2}$ numbers in the data-set. For each arrangement of the labels we must reconstruct the adjacency matrix and make sure that all distances can be reconstructed accurately. A running time of roughly $\binom{n}{2}!$ operations are required assuming reconstruction of the adjacency matrix takes constant time.

If we try $n = 8$ points, then $\approx 3 \cdot 10^{29}$ operations are needed. This algorithm is not an acceptable performer for this problem. We can however, reduce its running time by making some inferences about our data.

A better algorithm. (Skiena, Smith and Lemke, 1990) Given our example so far, namely a data-set $\{0, 1, 3, 4, 6, 9, 10\}$, we know that the largest distance between any two points is 10. If we fix the origin at one of these points and place another point at 10 we can proceed to place the remaining points in the data set somewhere between the origin and 10. Once these are positioned we know the location relative to each other and we can proceed to the next largest distance in the set.

For each step in the algorithm we choose the larges distance value in the remaining data. We have eliminated 10 already, so we can now try 9. The point corresponding to this distance from either 0 or 10 is then determined evaluating whether it is a distance of 9 from position 10 or position 0. In other words, the point must either be at $10 - 9 = 1$ or at $0 + 9 = 9$. We know that it can not be outside these two points because then there would exist a larger distance in our data-set which we would have chosen first for placement.

When placing the first point, there is no advantage to placing it at 1 or 9 so we can choose arbitrarily. Subsequent placements, however, must ensure that they are not violating the remaining distance data. At each step we can choose to place the point on the left or right of the remaining real-axis interval. The resulting set of choices can be written in the form of a tree (Figure 23.3).

For each new point we wish to place, we have two choices. This will generate a tree with 2^n branches. As we proceed down a branch of the tree we check whether the resulting arrangement of points on the line is consistent with the remaining distance data. If we ever encounter an inconsistency we backtrack up the tree and reverse our most recent choice. At each branching of the tree we must evaluate the validity of our answer to decide whether or now we must backtrack on our most recent decision. This decision requires we examine the remaining points in our set – an operation that is (worst case) $O(n)$ binary searches in our remaining data. This gives $O(n \log n)$ total operations per branch. In a worst case, we must evaluate for each branch in the tree which gives $O(2^n n \log n)$.

This algorithm is still exponential, however, its performance is quite improved over

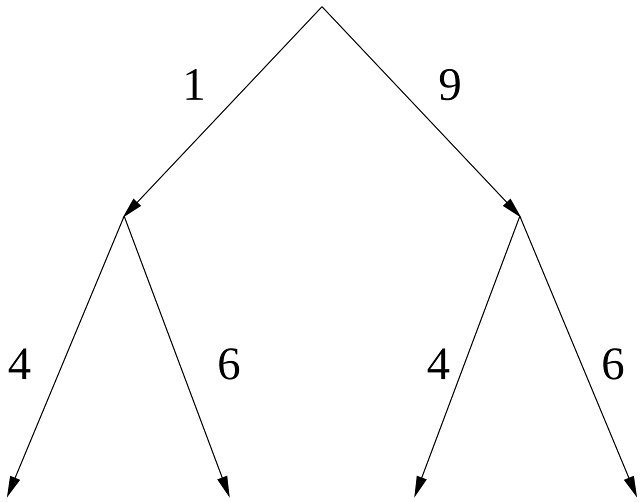

Figure 23.3: A backtracking tree with two decisions per branch.

the simple approach. For $n = 6$ the computation requires only 6000 operations. Although this algorithm's worst case performance is bounded by an exponential in n, its performance is usually quite good because the worst case is not likely to occur. Indeed, there is reason to believe that better bounds could be developed here, as this problem is suspected to be less difficult than NP-hard problems.

Labelled Distance Data

With labelled distance data situations we know the pair of points that correspond to the distances given. This happens, for example, when using nuclear magnetic resonance (NMR) imaging to determine the structure of protein, as we will describe below.

Nuclear Magnetic Resonance (NMR)

NMR is based on the principle that hydrogen nuclei have two spin states, up or down. There exists a specific energy separation between these two spin states so there's a particular frequency value that will cause a nuclei to flip its spin. We can then take a solution of the molecule we are interested in and scan it with increasing frequencies. When we get to the frequency that causes its spin to flip that frequency will be absorbed and we'll see a peak in the absorbtion spectrum. In essence, we get a characteristic spectrum for the molecule that lets us identify it later. For example, if we want to know if something has Carbon-13 we just check to see if it absorbs the frequencies at which Carbon-13 atoms flip their spin. However, this only tells us the content of a protein not the structure.

Structural information comes from *spin-spin coupling*: If two nuclei are very close in space then their spins interact and the frequency required for a spin flip is shifted. Hence the peaks in our spectrum become shifted slightly. The intensity of this effect depends on the distance between the nuclei so if you can determine all other the parameters involved you can infer the distance between nuclei based on how the spin frequencies have shifted. Specifically, between nuclei i and j spin-spin coupling varies as $\frac{c}{d_{ij}^6}$ where c is a constant and d_{ij}^6 is the distance between the nuclei.

There are two types of NMR experiments:

- **Correlated Spectroscopy (COSY):** In this experiment the spin-spin coupling is due to interactions between nuclei that are close covalently, i.e. via covalent bonds or covalently bonded atoms between them.
- **Nuclear Overhauser Enhancement Spectroscopy (NOSY):** In this experiment the spin-spin coupling is due to interactions between atoms close in space.

The difference between the two experiments is significant because if you have a protein that folds back on itself, as in Figure 23.4, nuclei may be close in space but not covalently so. Thus we use COSY to find the characteristic patterns of the protein to see where the peaks are coming from and so when we then do the NOSY experiment know what peaks are shifting, i.e. we get labelled distance data.

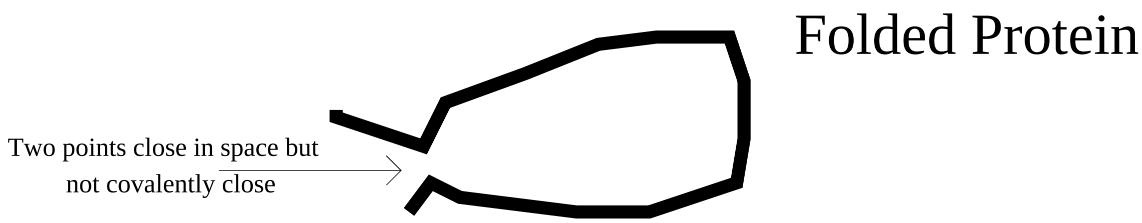

Figure 23.4: Covalent versus spatial proximity.

Unlike X-Ray crystallography where we have to crystalize the protein to examine it, in NMR we can study the protein in its native environment, i.e. in solution. However, NMR is more error prone and noisier. In paricular, errors are caused by the following factors:

- General measurement inaccuracy.
- The cofficient c in $\frac{c}{d_{ij}^6}$ is not known exactly. In fact, its not even constant over the entire molecule.
- The d_{ij} we are trying to find is not even constant. Because the protein is in solution it can and does move about, and these movements take place on the time scale of the experiment.
- The assignment of peaks is a difficult and human labor intensive process. Mistakes in peak assignment can lead to arbitrary errors, i.e. errors that are hard to precisely quantify.
- NMR has a limited horizon. It can't detect things beyond 5Å. As a comparison the protein being measured is roughly 30 Åin diameter and 1 atomic diameter is about 1.5Å. Thus we can only really detect things roughly 3 atoms away.
- NMR has a limited resolution. Within the 5Åhorizon of detection the resolution of the measurement is not very good. Because of the poor resolution, interactions are often classified as *strong* (1.5Åto 3Å), *medium* (3Åto 4Å), *weak* (4Åto 5Å), or *undetected* (5Åto ∞). Because the measurement scale is these four discrete classes instead of a continuum of real numbers one could argue that this problem is really a discrete classification problem.

Because of the level of error inherent in NMR data it is usually represented as bounds on the distances instead of actual distances themselves. In other words, we are given (l_{ij}, u_{ij}) where $\leq l_{ij} \leq d_{ij} u_{ij}$. Standard software packages refine the data by using the distance geometry techniques of Havel and Crippen, which has the following three phases:

1. **Bounds Smoothing:** Refine the upper and lower bounds using the triangle inequality.
2. **Generate:** Generate an initial very crude three-dimensional structure from the refined estimates.

3. **Optimize:** Globally optimize and refine the estimated structure to agree with what we know about the protein, e.g. it must be self avoiding in space, etc.

The last two phases are still solved by ad hoc methods. Bounds smoothing has a nice algorithmic component so we will describe that in further detail.

Bounds Smoothing

Recall that we are given (l_{ij}, u_{ij}) where $\leq l_{ij} \leq d_{ij} u_{ij}$ and the triangle inequality provides the additional constraint that $\leq d_{ij} d_{ik} + d_{kj}$ for all i,j, and k. We will derive two rules from these constraints, one that will raise lower bounds and one for lowering upper bounds and bounds smoothing will then consist of repeatedly applying these rules to tighten the bounds.

- **Upper Bounds Refinement Rule:** If for some i, j, k, we have that $u_{ij} > u_{ik} + u_{kj}$, then we set u_{ij} to $u_{ik} + u_{kj}$. For example, suppose in Figure 23.5 that $u_{ik} = 1$, $u_{kj} = 1$, and $u_{ij} = 3$ then we can change $u_{ij} = 2$ because the triangle inequality implies that $u_{ij} \leq u_{ik} + u_{kj} = 1 + 1 = 2$ (to visualize this, pretend the point k is on the line segment from i to j).
- **Lower Bounds Refinement Rule:** If for some i, j, k, l, we have that $l_{ij} < l_{kl} - u_{ik} - u_{jl}$, then we set l_{ij} to $l_{kl} - u_{ik} - u_{jl}$. For example, suppose in Figure 23.6 that $l_{kl} = 1000$, and $u_{ik} = u_{jl} = 100$ then we know that $l_{ij} \leq 1000 - 100 - 100 = 800$ (to visualize this, pretend points i and j are on the line segment between k and l).

The naieve method of applyng these rules, i.e. greedily apply a rule whenever you see one that is applicable, is very slow. There is however a polynomial time algorithm based on the shortest paths problem which will find the tightest bounds possible in $O(n^3)$ time. Furthermore, the rules are complete, i.e. once no more rules can be applied we know there exists a set of distances compatible with the bounds.

Because the protein is a three dimensional structure there are additional constraints the protein must satisfy. For example, the structure in Figure 23.7 could not exist in three dimensions because the point h would have to lie on all of the three distinct line segments between i, j and k. There is a tetrangle inequality and there are a set of rules you can derive from it for refining the bounds further. However, there's no known efficient way to run these rules to completion.

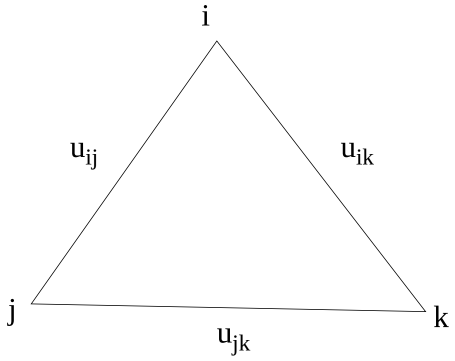

Figure 23.5: Refining upper bounds.

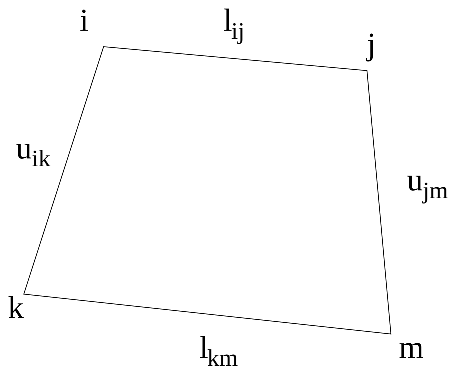

Figure 23.6: Refining lower bounds.

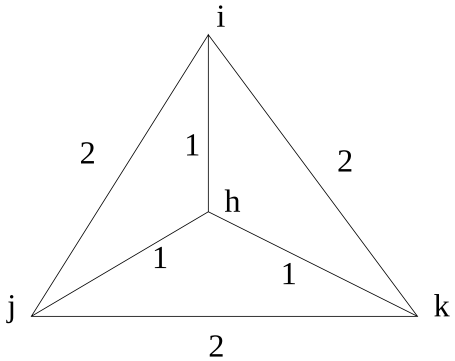

Figure 23.7: A configuration that can't exist in $\mathbb{R}^3$