Polygonal Chain Approximation with Applications

Ovidiu Daescu

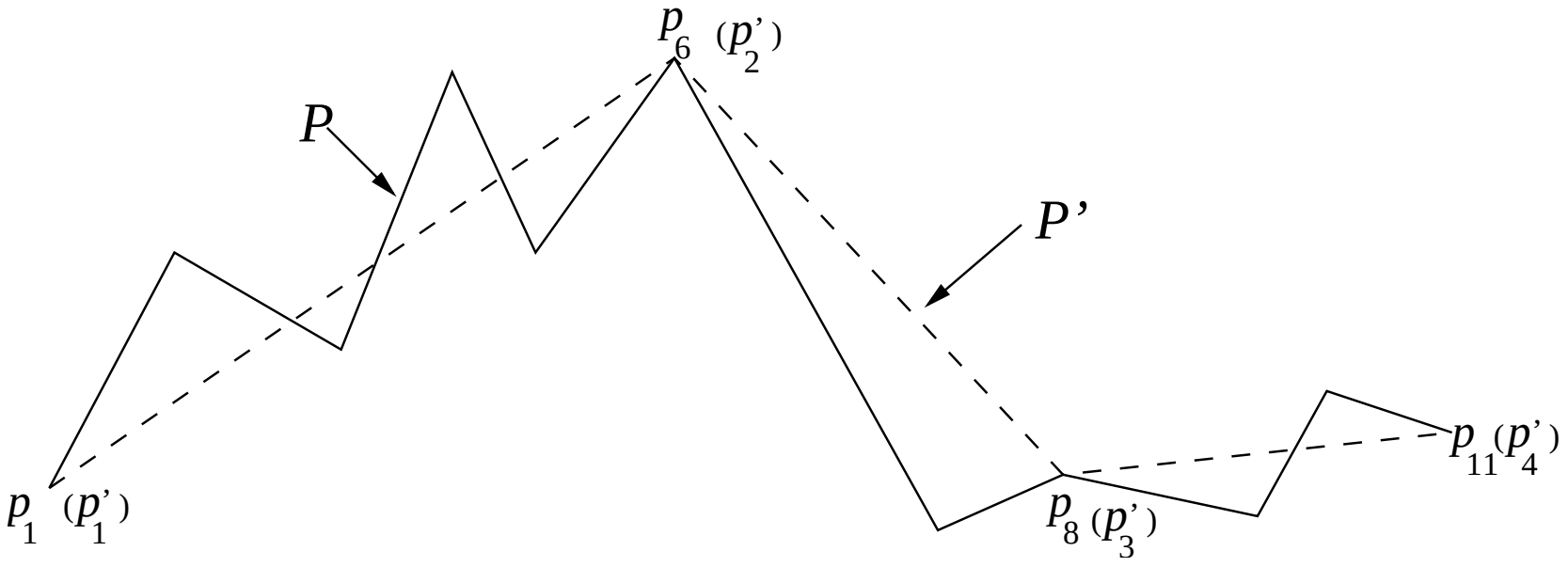
Department of Computer Science

University of Texas at Dallas

Min-# Polygonal Chain Approximation

Given a polygonal chain/path P of n vertices and a constant $\varepsilon > 0$, approximate P by another chain P' of $m < n$ vertices:

1. Vertices of P' are an **ordered** subsequence of vertices of P.
2. The **approximation error** of P' is no greater than the error ε.
3. The number of vertices of P', $m = |P'|$, is minimized.



Min-# Approximation: Other Versions

1. Vertices of P' are not required to be among the vertices of P. Still, line segment $\overline{p'_1 p'_2}$ approximates a subpath $\{p_1, p_2, \ldots, p_i\}$, and so on.

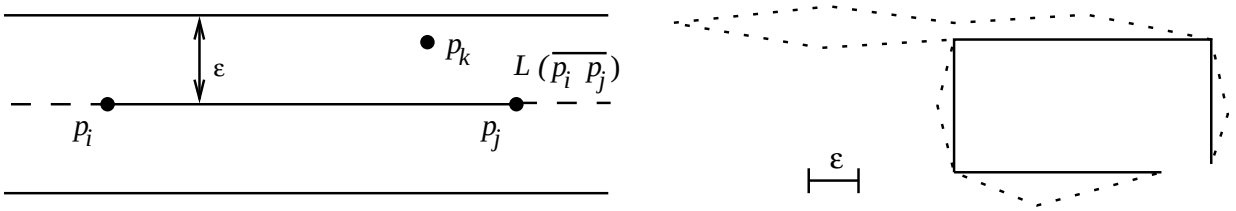
2. Fréchet Distance:
   - (continuous, 2-D) $O(n^2 \log^2 n)$ time (Guibas et al., '93)
   - (discrete, 3-D) $O(n \log n)$ time exploiting the *greedy* property (Bereg et al., '08)
   - Difficult for alignment in 3-D (appl.: protein backbones)
     - $O(n^{20} \log n)$ time continuous (Wenk, 2002)
     - $O(n^{14} \log n)$ time discrete (Jiang et al.)
   - Better use $O(n^{10}/\epsilon^6)$ ϵ-approximation (Kolodny and Linial, 2004)

Motivations / Applications

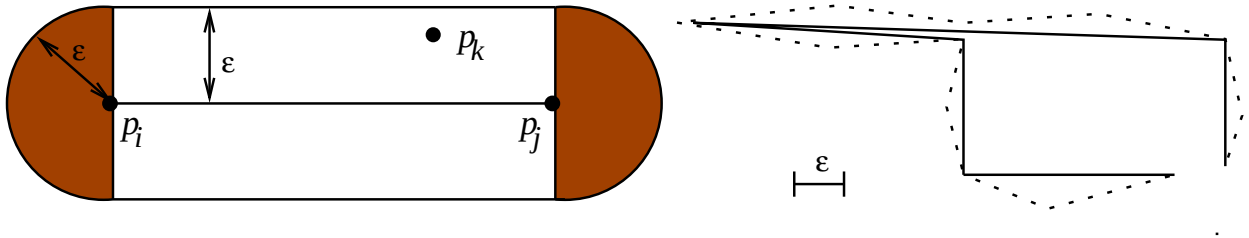
- Object simplification
  - Robotics
  - Image processing
  - Data compression
  - Computer graphics
  - Molecular modeling
- Technique introduced in this talk: measure extent of data.
  - Statistical analysis of data frames

Error Criteria (commonly used)

- **Infinite beam**



- **Tolerance zone**



- All results (2-D, 3-D) on general input data: at least $O(n^2)$ time

In This Presentation

- Consider min-# problem with L_2 distance metric and infinite beam criterion.
- Introduce a *dual space approach* and use it to obtain a subquadratic time exact algorithm in 2-D
  - infinite beam criterion
  - L_2 distance metric
  - when $d(p_i, p_j) \notin [\varepsilon, \varepsilon\sqrt{2}]$, for $1 \leq i < j \leq n$.
  - greedy breadth first traversal (BFT) algorithm.

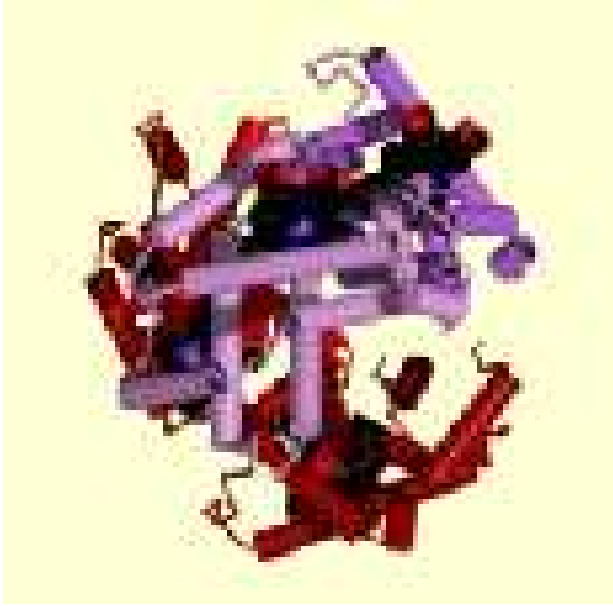
Applications

- An $O(n \log n)$ breadth first search procedure for some quadratic size graphs.
- An $O(n \log n)$ preprocessing, $O(\log n)$ query time solution for the following problem:

  (1) Given a set of n equal radius disks $D_1, D_2, \ldots, D_n$, construct a data structure that, for a query triplet (L, i, j), where L is a line and i and j are integers such that $1 \leq i < j \leq n$, answers whether L intersects all disks $D_i, D_{i+1}, \ldots, D_j$.
- An $O(n \log n)$ time, factor 2 approximation for the min-# problem with infinite beam criterion.
- All results can be extended to tolerance zone criterion.

Applications (cont.)

(1) above can be extended to 3-D and higher dimensions, with applications in representation and manipulation of molecular configurations (e.g., cover a molecular chain by cylinders) and statistical data analysis.



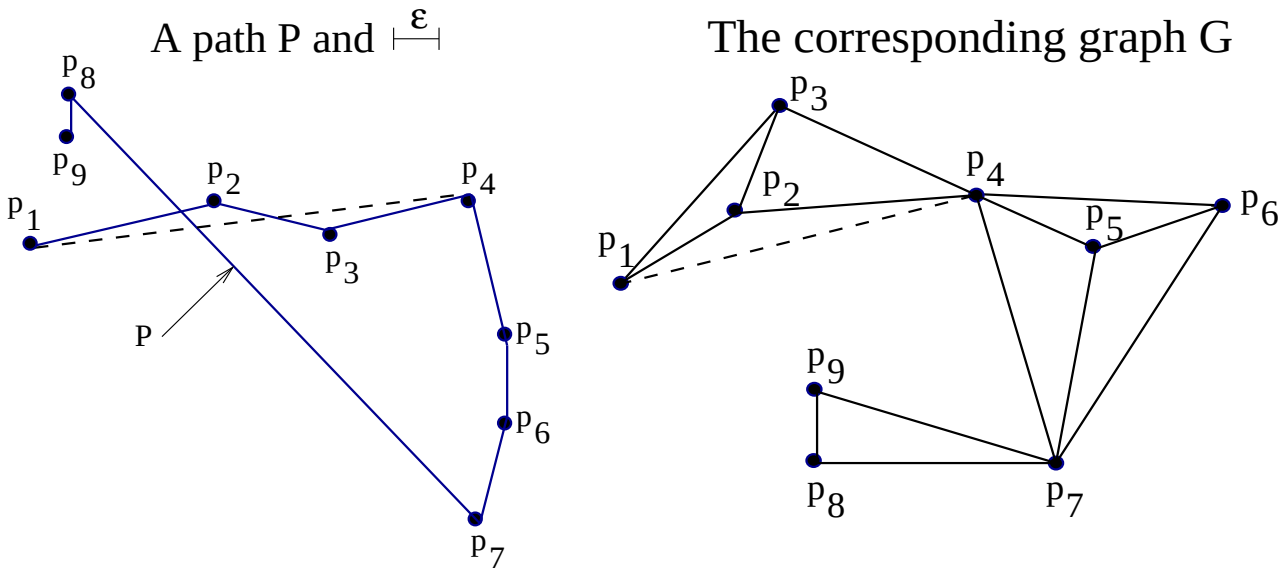
Applications (cont.)

Related Problems and Techniques:

- The # of connected components and complexity for the space of line transversals of simple convex objects.
- Farthest point from a query line.
- Farthest point q from query point p, constrained by a hyperplane (or, is there a q such that $d(p, q) > \varepsilon$?).
- Linearization and duality.
- Ray shooting in collections of hyperplanes.

Preliminaries

- General idea for Min-# problem (Imai & Iri)
  1. Build a graph G on the vertices of the path: each edge is a "valid" approximation.
  2. **Compute a shortest path** in G.



Preliminaries (Cont.)

- Observation: an optimal path can be computed while computing the ε-approximating segments.
- Previous solutions:
  1. Iterative, incremental approach: Chen & Daescu
     - Path length at a vertex of P is updated multiple times.
  2. Breadth first traversal (BFT) approach: Daescu
     - Standard queue operations.
     - Shortest p_1-to-p_i path length at a vertex p_i of P is computed when p_i is *enqueued* (not updated again by future computation).

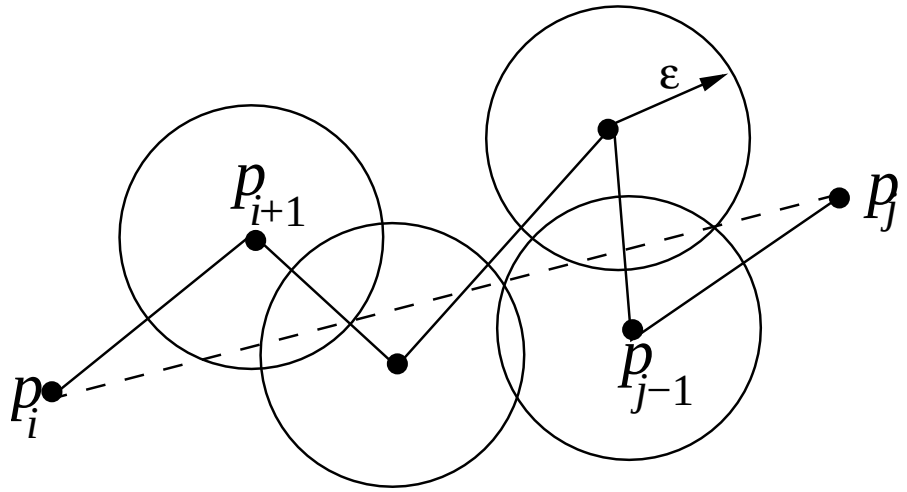
Preliminaries (Cont.)

Subquadratic time is possible if L_2 metric is replaced by L_1 metric (Agarwal and Varadarajan).

- Construct compact (bipartite clique cover) representation $\mathcal{G}$ of the ϵ-approximation graph $G(P)$ using divide-and-conquer:
  - Partition P into P_1, P_2, where $P_1 = (p_1, \ldots, p_{\lfloor n/2 \rfloor})$ and $P_2 = (p_{\lfloor n/2 \rfloor + 1}, \ldots, p_n)$.
  - Recursively compute the clique covers $\mathcal{G}_1$ and $\mathcal{G}_2$ of $G(P_1)$ and $G(P_2)$.
  - In the merge step, compute cover $\mathcal{G}_{12}$ of the edges E_{12}, where an edge $e \in E_{12}$ if it is an ϵ-approximating edge with one endpoint in P_1 and the other one in P_2.
  - $\mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_{12}$ is a clique cover of $G(P)$ of the size $O(n^{4/3+\delta})$.
- Using this clique cover representation, a shortest path from p_1 to p_n can be computed in $O(|\mathcal{G}| + |V|)$ time.

A Query Based Approach

- We combine the BFT algorithm with a query method to compute the shortest p_1-to-p_n path while constructing the edges of the graph (no need to store the graph).
- To decide if $\overline{p_i p_j}$ is a "valid" (ε-approximating) segment, we need to check if $\overline{p_i p_j}$ intersects all disks $D(p_{i+1}, \varepsilon), D(p_{i+2}, \varepsilon), \ldots, D(p_{j-1}, \varepsilon)$.



A Query Based Approach (Cont.)

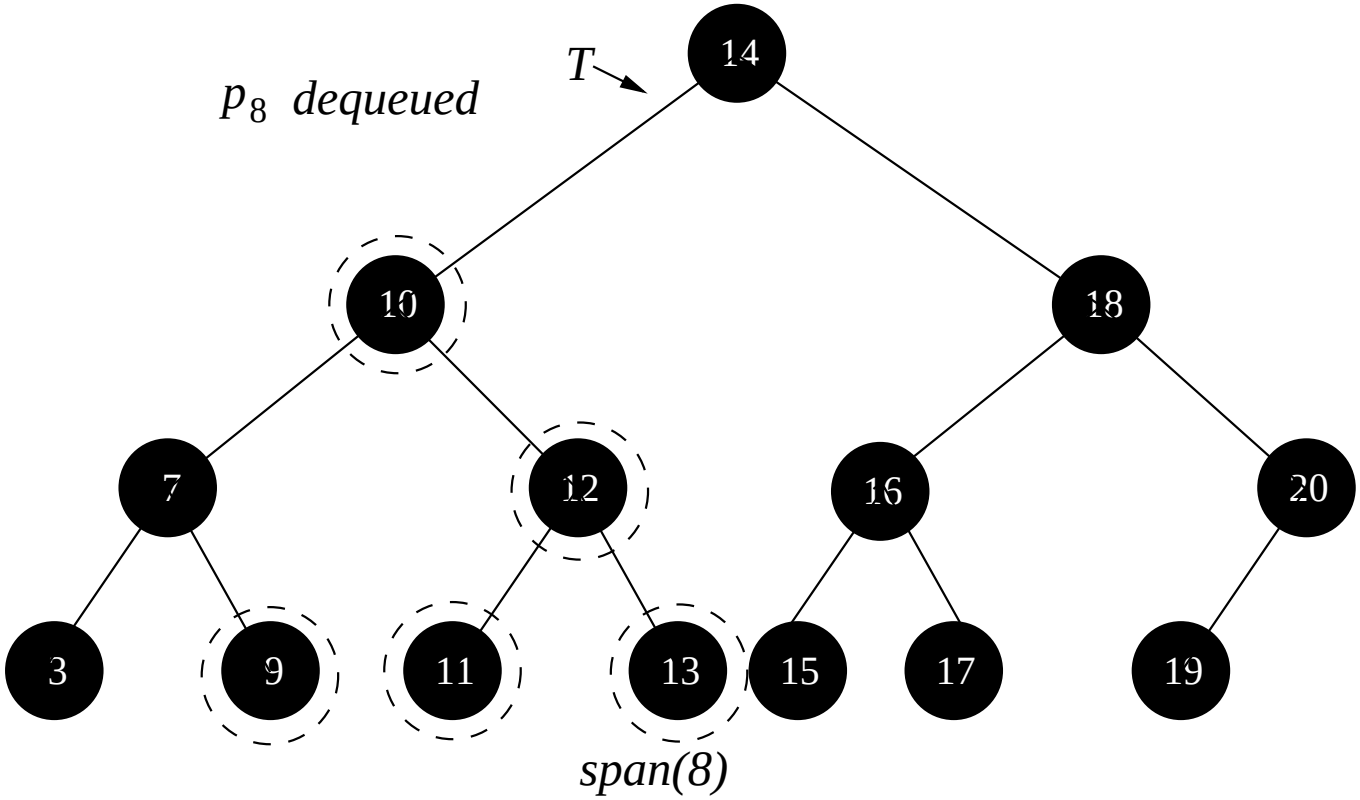
- Three categories of vertices of P:
  1. *processed*: visited by BFT and no longer in the queue.
  2. *active*: visited by BFT and in the queue.
  3. *inactive*: not yet reached by BFT.
- Two key operations:
  1. $Span(i)$:
     compute the largest index j such that there is a line stabler through p_i for the set of disks $D(p_{i+1}, \varepsilon), D(p_{i+2}, \varepsilon), \ldots, D(p_j, \varepsilon)$.
  2. $Query(i, j)$:
     answer if $\overline{p_i p_j}$ is an ε-approximating segment.

A Query Based Approach (Cont.)

- Greedy BFT approach
  - Vertices in **active** set are maintained in two priority queues having as keys the indices of vertices of P.
    - 1^{st} queue: vertices reached with $k-1$ links.
    - 2^{nd} queue: vertices reached with k links.
  - A *dequeue* operation on each of the queues returns the largest index in the queue.
- Observation: only pairs (i,j), with $p_i \in 1^{st}$ **active** set, $p_j \in$ **inactive** set and $i<j$ should be considered for $Query(i,j)$.

A Query Based Approach (Cont.)

- Assuming $Span(i)$ is known, the inactive vertices for p_i can be found in $O(\log I + k)$ time, where I is the set of currently inactive vertices and k is the number of inactive vertices in I with indices between i and $Span(i)$.



A Query Based Approach (Cont.)

Application:

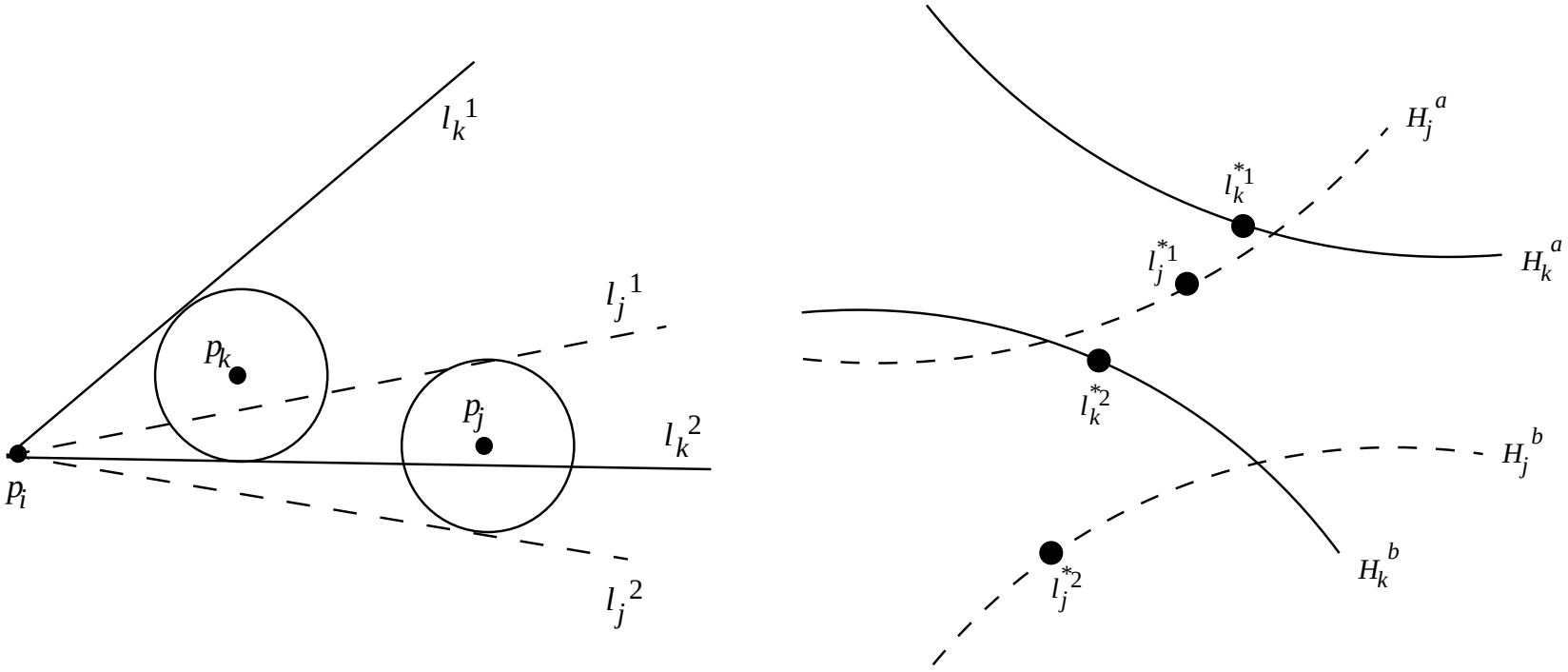
- Let G be an unweighted directed graph with n vertices, such that for each vertex $v_i \in G$, there are edges $(v_i, v_{i_1}), (v_i, v_{i_1+1}), (v_i, v_{i_1+2}), \ldots, (v_i, v_{i_2}) \in G$.
- G can be specified by its set of vertices and the index ranges (i_1, i_2) associated with each vertex $v_i \in G$.
- Computing a single source shortest path tree in G:
  - Standard BFT: $O(n^2)$ time.
  - BFT with tree T : $O(n \log n)$ time and $O(n)$ space.

A Query Based Approach (Cont.)

- Observation: the time complexity of the greedy BFT algorithm depends on:
  - The time complexities for performing $Span(.)$ and $Query(.,.)$ operations.
  - The number of **failed** $Query(.,.)$ operations.
- The time complexities for $Query(i,j)$ and $Span(i)$:
  - $Query(i,j)$: $O(\log n)$ time.
  - $Span(i)$: $O(\log n)$ time, if $d(p_i,p_j) \notin [\varepsilon, \varepsilon\sqrt{2}]$, for all $1 \leq i < j \leq n$.
- How???

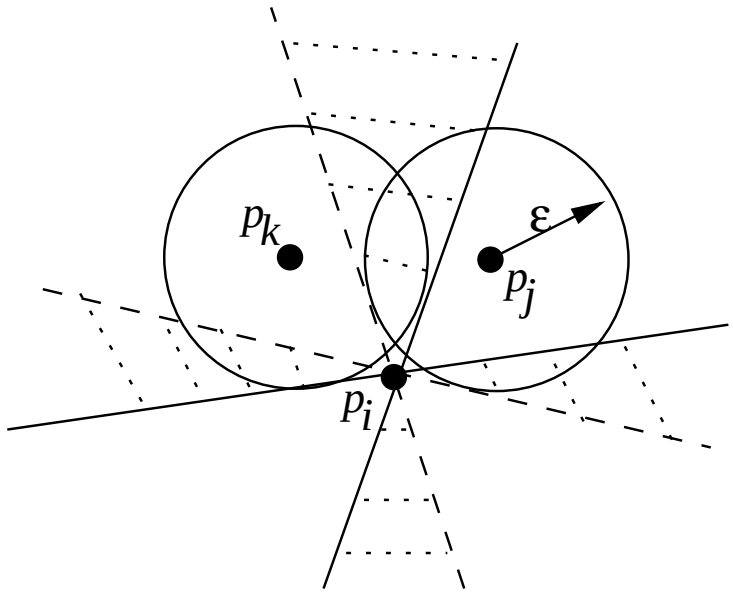
A Query Based Approach (Cont.)

- Using a standard point-line duality transform, the set of lines tangent to the disk $D(p_k, \varepsilon)$ is mapped to two hyperbolic branches H_k^a and H_k^b in the dual plane.
  - L_{ij}: the lower envelope of upper branches.
  - U_{ij}: the upper envelope of lower branches.



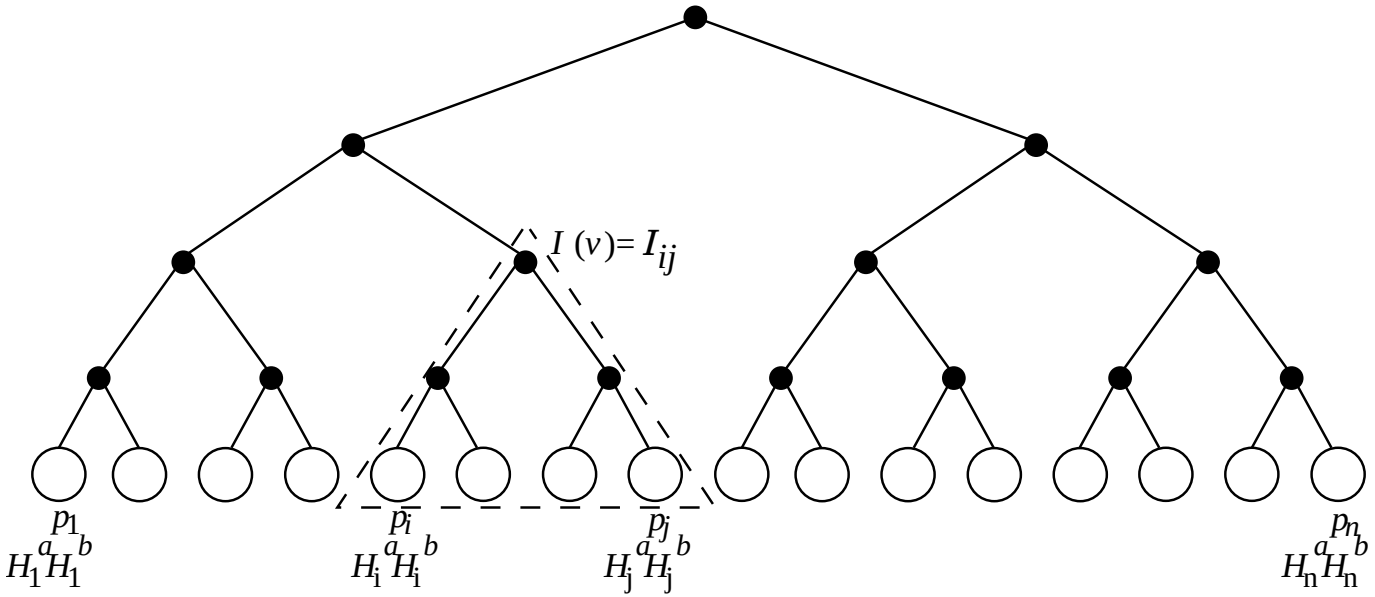
A Query Based Approach (Cont.)

- Observation:
  - Line $L(\overline{p_i p_j})$ is a common transversal of $\{D(p_{i+1}, \varepsilon), D(p_{i+2}, \varepsilon), \ldots, D(p_{j-1}, \varepsilon)\}$ *iff* its dual point lies in $\mathcal{I}_{ij}$ (sandwiched between $\mathcal{L}_{ij}$ and $\mathcal{U}_{ij}$).
  - Requirement, $d(p_i, p_j) \notin [\varepsilon, \varepsilon\sqrt{2}]$, assures that the line dual to p_i has at most one line segment in $\mathcal{I}_{ij}$.



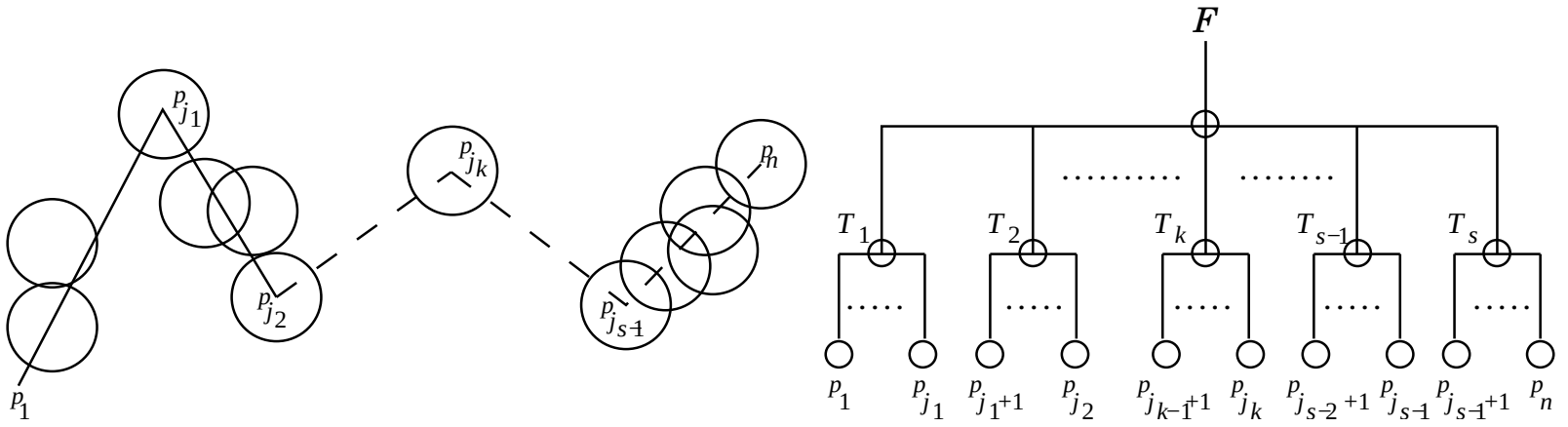
A Query Based Approach (Cont.)

- Build complete binary tree T:
  - Leaves: the vertices of P with hyperbolic branches.
  - Internal node: the region $I(v) = \mathcal{I}_{ij}$, where i and j are the smallest and largest indices of the leaf descendants of v.
  - $I(v)$ for all vertices of T can be computed by simple merge-like operations in $O(n \log n)$ time.



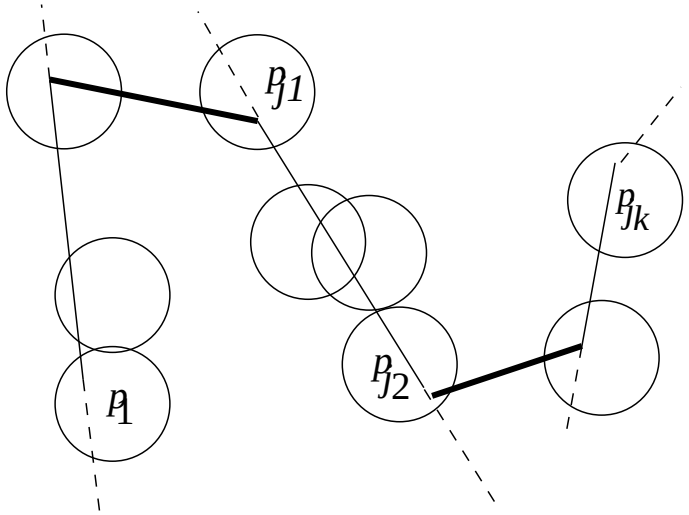
A Query Based Approach (Cont.)

- In general, obtain a family of trees $\mathcal{F} = \{T_1, T_2, \ldots, T_s\}$:
  - T_1's leaves correspond to the longest possible prefix $D_1 = \{D(p_1, \varepsilon), D(p_2, \varepsilon), \ldots, D(p_{j_1}, \varepsilon)\}$ that admits a line transversal.
  - T_2's leaves correspond to the longest possible prefix starting at p_{j_1+1}, and so on.
  - Total $O(n \log n)$ time.



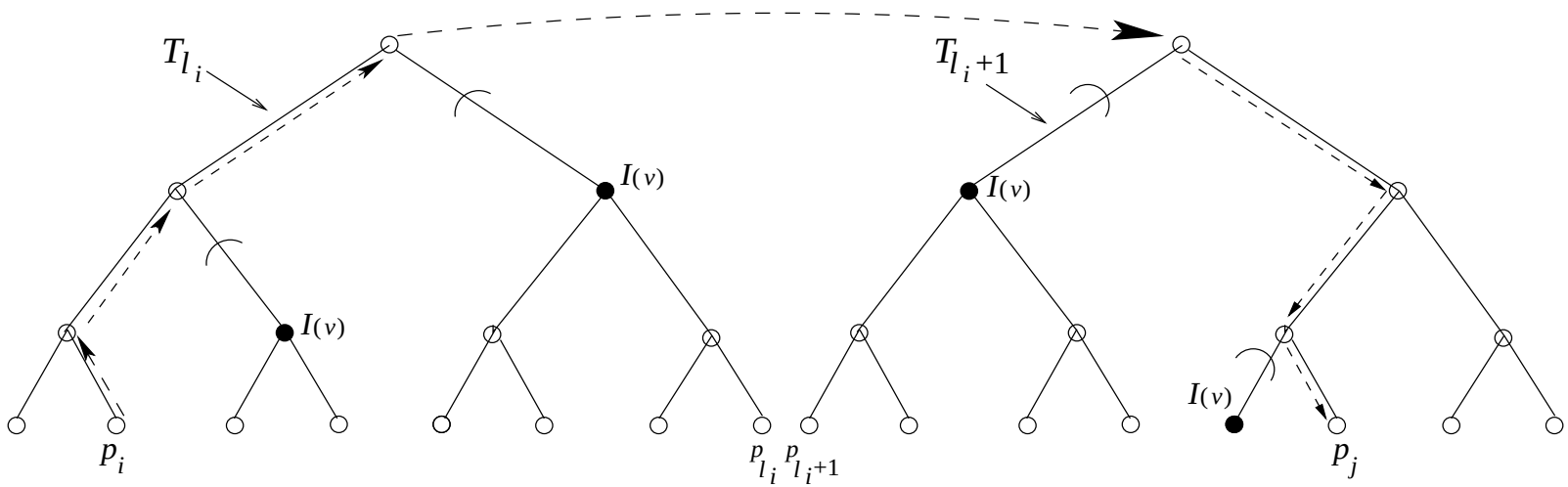
A Query Based Approach (Cont.)

- $\mathcal{F}$ can be used to compute an approximating path with vertices inside the error tolerance regions of vertices of P and of size at most twice the size of an optimal approximation.
  - An optimal approximating path must have at least $|\mathcal{F}|$ vertices.



A Query Based Approach (Cont.)

- Computing $Query(i, j)$:
  - Let p_{ij} be the dual of the line $L(\overline{p_i p_j})$.
  - Form a search path π_{ij} in $\mathcal{F}$, from the leaf l_i to the leaf l_j.
  - Answer $Query(i, j)$: determining if p_{ij} inside $I(v)$ for each v that is a right (left) fringe node of $\pi_{ij} \rightarrow O(\log n)$ time.
- Computing $Span(i)$:
  - In a similar way $\rightarrow O(\log n)$ time.



Computing *Query(i,j)*

A Query Based Approach (Cont.)

Result:

(1) Given a set of n equal radius disks $D_1, D_2, \ldots, D_n$, in $O(n \log n)$ time one can construct a data structure of size $O(n \log n)$ such that, for a query triplet (L, i, j), where L is a line and i and j are integers, $1 \leq i < j \leq n$, it can be decided in $O(\log n)$ time whether L intersects all disks $D_i, D_{i+1}, \ldots, D_j$.

Theorem:

Under the condition, $d(p_i, p_j) \notin [\varepsilon, \varepsilon\sqrt{2}]$, $1 \leq i < j \leq n$, the min-# problem with the infinite beam criterion and L_2 distance metric can be solved in $O(n^{4/3+\delta})$ time, where $\delta > 0$ is an arbitrarily small constant.

Proof: Combine dual space, query approach with Agarwal and Varadarajan's clique based method.

What about higher dimensions (3-D)

- The problem is much harder.
- Brute force is simple, takes $O(n^3)$.

Consider the subproblem: Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n indexed points in R^d, where $d \geq 3$ is a constant, for each line $L(p_i, p_j)$, $1 \leq i < j \leq n$, report the farthest point p_k such that $i < k < j$.

- $O(n^{3-3/(\lfloor \frac{f(d)}{2} \rfloor + 1)} \log^{O(1)} n)$ time and space possible using a query approach.

Related Problems

Preprocess a set P of n points in R^d, $d \geq 3$, so that for a query line L one can efficiently report the farthest point $q \in P$ from L (and thus the minimum enclosing cylinder with central axis L).

- $O(s \cdot \log^{O(1)} n)$ space and time preprocessing.
- $O(n \log n / s^{1/\lfloor \frac{f(d)}{2} \rfloor})$ query time.

Evaluate extent of a data by computing a minimum-width cylindrical shell with central axis L and containing the data set P, where a *cylindrical shell* is the region enclosed between two co-axial cylinders.

Related Problems (cont.)

Theorem: A set of n indexed points $P = \{p_1, p_2, \ldots, p_n\}$ in R^d, where $d \geq 3$ is a constant, can be preprocessed in time and space $O(n^{\lfloor \frac{f(d)}{2} \rfloor} \log^{O(1)} n)$ such that for a query line L one can report in $O(\log^2 n)$ time the farthest point p_k to L that is in a specified range $\{p_i, p_{i+1}, \ldots, p_j\}$, where $1 \leq i < j \leq n$ and $i < k < j$.

Theorem: A set of indexed points in R^d, for any constant $d \geq 3$, can be approximated by a polygonal chain P whose vertices are an ordered sequence of the points, under two commonly used optimization versions (min-# and min-ϵ), with two commonly used error measures (tolerance zone and infinite beam) in time and space $O(n^{3-3/(\lfloor \frac{f(d)}{2} \rfloor + 1)} \log^{O(1)} n)$.

Related Problems (cont.)

Theorem: An n-point set P in R^d, $d \geq 3$, can be preprocessed with $O^*(s)$ space and time such that for a query line L the minimum-width cylindrical shell of P, with central axis L, can be be found in $O(n \log n / s^{1/\lfloor \frac{f(d)}{2} \rfloor})$ time.

Theorem: Given a set of n indexed points $P = \{p_1, p_2, \ldots, p_n\}$ in R^d, where $d \geq 3$ is a constant, with $O^*(n^{3-3/(\lfloor \frac{f(d)}{2} \rfloor + 1)})$ time and space one can report for each line $L(p_i, p_j)$, $1 \leq i < j \leq n$, the minimum-width cylindrical shell of the points $\{p_i, p_{i+1}, \ldots, p_j\}$, with central axis $L(p_i, p_j)$.

Linearization and Farthest Point from Line

$\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ collection of n d-variate polynomials of constant degree. $\mathcal{F}$ admits a *linearization* of dimension k if, for some integer $b > 0$, there exists a $(d+b)$-variate polynomial

$$g(x,a) = \psi_0(a) + \psi_1(a)\varphi_1(x) + \psi_2(a)\varphi_2(x) + \ldots + \psi_k(a)\varphi_k(x) + \varphi_{k+1}(x)$$

for $x \in R^d$ and $a \in R^b$, such that $f_i(x) = g(x, a_i)$, for $i = 1, 2, \ldots, n$ and some $a_i \in R^b$.

- A line L in 3-D can be uniquely defined by 4-tuple (x_1, x_2, x_3, x_4).
- Euclidean distance from point $p = (a_1, a_2, a_3)$ to line L: $d(L, p)$.

Linearization (cont.)

$$d(L,p) = d(x,a) = d(x_1,x_2,x_3,x_4,a_1,a_2,a_3) =$$
$$\{[(x_4^2+1)x_1^2 + (x_3^2+1)x_2^2 - 2x_1x_2x_3x_4] + 2[x_2x_3x_4 - x_1(x_4^2+1)]a_1$$
$$+2[x_1x_3x_4 - x_2(x_3^2+1)]a_2 + 2[x_1x_3 + x_2x_4]a_3 - 2[x_3x_4]a_1a_2$$
$$-2[x_3]a_1a_3 - 2[x_4]a_2a_3 + [1](a_1^2+a_2^2) + [x_3^2](a_2^2a_3^2)$$
$$+[x_4^2](a_1^2+a_3^2)\}/(x_3^2+x_4^2+1).$$

For n-point set P and line $\mathcal{L} = (\xi_1,\xi_2,\xi_3,\xi_4)$, a point $p_i \in P$ is the farthest point from $\mathcal{L}$ if $d(\mathcal{L},p_i) = \max\{d(\mathcal{L},p_k) : k = 1,2,\ldots,n\}$, that is $d(L,p_i)$ is on the upper envelope of $\{d(L,p_1), d(L,p_2), \ldots, d(L,p_n)\}$ at $(\xi_1,\xi_2,\xi_3,\xi_4)$.

Linearization (cont.)

$\varphi_1(x) = x_2x_3x_4 - x_1(x_4^2+1)$, $\varphi_2(x) = x_1x_3x_4 - x_2(x_3^2+1)$,
$\varphi_3(x) = x_1x_3 + x_2x_4$, $\varphi_4(x) = x_3x_4$, $\varphi_5(x) = x_3$, $\varphi_6(x) = x_4$,
$\varphi_7(x) = x_3^2$, $\varphi_8(x) = x_4^2$,
$\varphi_9(x) = (x_4^2+1)x_1^2 + (x_3^2+1)x_2^2 - 2x_1x_2x_3x_4$,

$$\begin{array}{llllll} \psi_0(a) = a_1^2 + a_2^2, & \psi_1(a) = 2a_1, & \psi_2(a) = 2a_2, \\ \psi_3(a) = 2a_3, & \psi_4(a) = -2a_1a_2, & \psi_5(a) = -2a_1a_3, \\ \psi_6(a) = -2a_2a_3, & \psi_7(a) = a_2^2a_3^2, & \psi_8(a) = a_1^2 + a_3^2. \end{array}$$

Thus, we obtain a linearization of dimension 8:

$$g(x,a) = \psi_0(a) + \sum_{i=1}^{8} \varphi_i(x)\psi_i(a) + \varphi_9(x).$$

In R^d, linearization of dimension $f(d) = O(d^2)$.

Farthest Point Problem: Reduction

Reduce problem of finding farthest point $p \in P$ from line $\mathcal{L}$ defined by the 4-tuple $\xi = (\xi_1, \xi_2, \xi_3, \xi_4)$ to a ray shooting query in a set of n 9-dimensional hyperplanes $g(x, p_i)$, for $i = 1, 2, \ldots, n$:

- Shoot a vertical ray from the 9-dimensional point $(\varphi_1(\xi), \varphi_2(\xi), \ldots, \varphi_8(\xi), \infty)$ in the negative direction of the last coordinate. If $g(x, p_i)$ is the first hyperplane hit by this ray then p_i is the farthest point of P from $\mathcal{L}$.

Note: Only the points on the convex hull of P matter. The convex hull of n points in R^d can be computed in $O(n^{\lfloor \frac{d}{2} \rfloor})$ time.

Farthest Point Problem

Preprocess a set P of n points in R^d, $d \geq 3$, so that for a query line L one can efficiently report the farthest point $q \in P$ from L (and thus the minimum enclosing cylinder with central axis L).

Solution: Build a static data structure for ray-shooting queries:

- Can answer a query in $O(n \log n / s^{1/\lfloor \frac{f(d)}{2} \rfloor})$ time with $O^*(s)$ space and preprocessing time.
  - Can answer a query in $O(\log n)$ time using $O^*(n^{\lfloor \frac{f(d)}{2} \rfloor})$ space and preprocessing time.
  - When $P \in R^3$, this implies $O^*(n^4)$ preprocessing.

Example

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n indexed points in R^d, where $d \geq 3$ is a constant, for each line $L(p_i, p_j)$, $1 \leq i < j \leq n$, report the farthest point p_k such that $i < k < j$.

Proof: Use balanced binary tree T on the set of points P.

- At each internal node v, compute and store a ray shooting data structure for leaf descendants of v.
- At level k, 2^k ray shooting data structures:
  - Each constructed on $\frac{n}{2^k}$ points.
  - Each answers $O(\frac{n^2}{2^k})$ queries.
  - Query/preprocessing trade-off:
    $s = O((\frac{n^3}{2^{2k}})^{\lfloor \frac{f(d)}{2} \rfloor / (\lfloor \frac{f(d)}{2} \rfloor + 1)}) =$
    $O(n^{3\lfloor \frac{f(d)}{2} \rfloor / (\lfloor \frac{f(d)}{2} \rfloor + 1)} (2^k)^{-2\lfloor \frac{f(d)}{2} \rfloor / (\lfloor \frac{f(d)}{2} \rfloor + 1)})$.

- At level k:
  $O^*(2^k s) = O^*(n^{3\left\lfloor \frac{f(d)}{2} \right\rfloor / (\left\lfloor \frac{f(d)}{2} \right\rfloor + 1)} (2^{(1-\left\lfloor \frac{f(d)}{2} \right\rfloor)/(\left\lfloor \frac{f(d)}{2} \right\rfloor + 1)})^k)$.

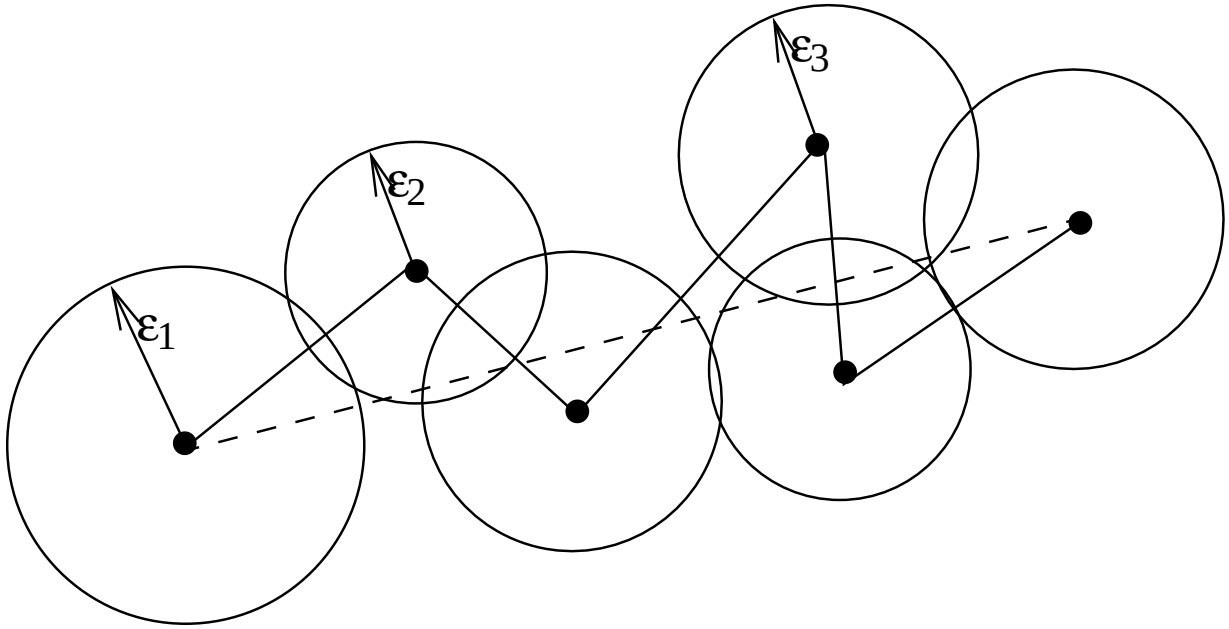
- Over T:
  $O^*(n^{3\left\lfloor \frac{f(d)}{2} \right\rfloor / (\left\lfloor \frac{f(d)}{2} \right\rfloor + 1)} \sum_{k=1}^{\log n} (2^{(1-\left\lfloor \frac{f(d)}{2} \right\rfloor)/(\left\lfloor \frac{f(d)}{2} \right\rfloor + 1)})^k) =$
  $O^*(n^{3-3/(\left\lfloor \frac{f(d)}{2} \right\rfloor + 1)})$.

- Most of the other results follow from this one.

- In 3-D, $f(d) = 9$ resulting in $O(n^{2.4})$ time.
  - Can have tens-of-thousands of data points / molecules.
  - If $n = 10^6$ then $n^3 = 10^{18}$ but $n^{2.4} < 10^{15}$ **(computable!)**.

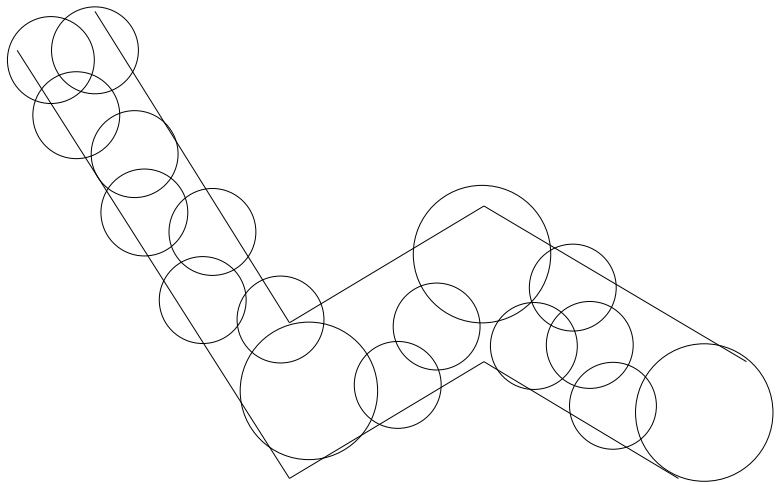
Chain of Molecules: 3-D

- Atom is a "sphere", center is a point.
- There are a few atom types.
  - Spheres with different radius (**van der Waals** radius).
  - Atoms in a chemical bond interpenetrate.
  - They cannot get too close ($[\varepsilon_i, \varepsilon_i\sqrt{2}]$ condition may hold).



Chain of Molecules

- Can extend the query approach to handle it.
- It may be possible to use an approximate chain or segment for allignment, structural matching, folding, etc.
- Can pack the molecular chain into cylinders.
  - Can optimize various functions.



Conclusion

Plenty of work ahead.