Randomized Techniques in Computational Geometry

I Fundamentals

Sandeep Sen
I.I.T. Delhi, India
Outline

- Kind of Problems
- Kind of Algorithms
- Random Sampling in Geometry
- Incremental Construction
Primary Source (for this talk)

- Clarkson and Shor, “Applications of random sampling in computational geometry”. DCG 89.
- Reif and Sen “Optimal parallel randomized algorithms for 3-D hulls and related problems. SIAMJC 92.
- Mulmuley and Sen, Dynamic point location in arrangements of hyperplanes. DCG 92.
The problems addressed

• Range searching
• Ray shooting/Point location
• Planar partitioning
• Convex hulls
• Linear programming
• Nearest neighbour
The problems addressed cont’d

- Triangulation
- Hidden surface
- Levels of arrangements
- Diameter/ Width
- Euclidean minimum spanning tree

Exact computations
“Real” RAM
Randomization in Computational Geometry

- Improved Complexities
  Range searching

- Simpler Algorithms
  Convex hulls, triangulation, LP

- Dynamic Algorithms
  with minimal modifications

- Parallel Algorithms
  Faster, more efficient
Randomized Algorithms

- RNG $O(\log n)$ bits in constant time
- Assumes **No** input distribution
- Halts with a correct output
- Running time is bounded by some probability distribution

Expectation $[T]$

Tail estimates: $\text{Prob. } [T_n > f(n)] \leq \epsilon$
Elementary Tools

Probabilistic Inequalities:

- Markov: only expectation
- Chernoff: moment generating function
- Chebychev: intermediate (bounding random bits)

**Linearity of Expectation**

\[ E[X + Y] = E[X] + E[Y] \]

for any \( X, Y \) not necessarily independent

\[ P(A \cup B) \leq P(A) + P(B) \]

**Notation:**

\( \widetilde{O} (\cdot) \overset{\text{def}}{=} O(\cdot) \) with prob. \( 1 - \frac{1}{n} \)
Quick Sort

Splitter

Sort recursively

Sort recursively

Ideal: \[ T(n) = 2T(n/2) + O(n) \]
\[ \leq O(n \log n) \]

randomized: \[ T(n) = T(n_1) + T(n - n_1 - 1) + O(n) \]
where \( n_1 \) is a random variable in \([1, n]\)

\[ E[T(n)] : O(n \log n) \]
Generalized: $r$ splitters

Ideal: \[ T(n) = \sum T(n/r) + O(n \log r) \leq O(n \log n) \]

\[ E[T(n)] : O(n \log n) \]
Kind of sampling

Choose a random subset $R \subset N$

- with replacement

- without replacement

- Bernoulli sample
  (expected sample size is $|R|$ by picking every element with prob. $(\frac{|R|}{|N|})$).

“parallel” sampling

Remark: Little difference in final results. We shall choose the one that simplifies proof.
Some Notations

$N$ : set of objects $|N| = n$

$\sigma$ : $(D(\sigma) \quad L(\sigma)) \quad (D, L)$

configuration define conflict

Assumptions (for technical reasons)

- $D$ is bounded by constant
- Valence (no. of $\sigma$ with same $D(\sigma)$ is bounded
Need for bounded degree
All lines tangential to a circle

Any subset of size $r$ induces a face that intersects $n - r$ lines.
Some Notations cont’d

- $\Pi(N)$: set of configurations (multiset) over $N$
- $\Pi^i(N)$: set of configurations with conflict size $= i$
- $\Pi^0(R)$: configurations active

For $R \subset N$, $\sigma$ is feasible (for $R$) if $D(\sigma) \subset R$

We shall often use $\Pi(N)$ to also denote $|\Pi(N)|$
A simple combinatorial bound

Claim \( \Pi^a(N) = O(2^{a+d} \cdot E[\Pi^0(R)]) \)
where \( R \) is a random sample of size \( n/2 \).

\( \Pi^0(R) = \sum_{\sigma \in \Pi(N)} I_{\sigma,R} \)
where \( I_{\sigma,R} \) is 1 if \( \sigma \) is feasible.

\( E[\Pi^0(R)] = E[\sum_{\sigma \in \Pi(N)} I_{\sigma,R}] = \sum_{\sigma \in \Pi(N)} E[I_{\sigma,R}] \)
\( = \sum_{\sigma \in \Pi(N)} \Pr\{\sigma \in \Pi^0(R)\} \)
\( \geq \sum_{\sigma \in \Pi^a(N)} \{\sigma \in \Pi^0(R)\} \)
\( = \Pi^a(N) \cdot \frac{1}{2^{a+d}} \)
Claim:

\[ \Pr\{ \max_{\sigma \in \Pi^0(R)} l(\sigma) \geq c \frac{n}{r} \log r \} \leq \frac{1}{2} \]

\[ |R| = r \text{ by Bernoulli sampling} \]

\[ p(\sigma, r) : \text{conditional probability that none of the } k \text{ conflicting } \]
\[ \text{element are selected given } \sigma \text{ is feasible} \]
\[ \leq (1 - r/n)^k \]
\[ \leq e^{-c \log r} \quad \text{for } k \geq cn/r \log r \]
\[ = 1/r^c \]

BAD \sigma

BAD \sigma
$q(\sigma, r) : \text{Prob. that } D(\sigma) \subset R$

**Prob.** that $\sigma \in \Pi^0(R) = p(\sigma, r) \times q(\sigma, r)$

**Prob.** that some $\sigma \in \Pi^0(R)$ is **BAD** ($l(\sigma) \geq c(n \ln r)/r$) :

\[
\leq \frac{1}{r^c} \sum_{\sigma \in \Pi(N)} q(\sigma, r)
\]

\[
= \frac{1}{r^c} \ E[\Pi(R)]
\]

(usually $\Pi(R) = r^{O(1)}$)

\[
\leq 1/2 \text{ for appropriate } c
\]
**Sum of subproblem sizes**

**Def:** $c$-order conflict $\left( \begin{array}{c} l(\sigma) \\ c \end{array} \right)$, for some $c \geq 0$

Let $T_c = \sum_{\sigma \in \Pi^0(R)} \left( \begin{array}{c} l(\sigma) \\ c \end{array} \right)$

**Remark** For technical reasons it is not $l(\sigma)^c$. $T_0 = |\Pi^0(R)|$. $T_1 =$ sum of subproblems.

**Claim** $E[T_c] = O\left( \left( \frac{n}{r} \right)^c E[\Pi^c(R)] \right)$

For constant $c$, $E[\Pi^c(R)] = O(E[\Pi^0(R)]$ implying that average conflict size is very close to $\frac{n}{r}$.
\[ T_c = \sum_{\sigma \in \Pi(N)(R)} \binom{l(\sigma)}{c} I_{\sigma,R} \text{ where } I_{\sigma,R} = 1 \text{ if } \sigma \in \Pi^0(R). \]

\[ E[T_c] = \sum_{\sigma \in \Pi(N)} \binom{l(\sigma)}{c} p^d(\sigma) \cdot (1 - p)^{l(\sigma)} \text{ for } l(\sigma) \geq c. \]

\[ = \sum_{\sigma \in \Pi(N)} \binom{l(\sigma)}{c} p^{d(\sigma)+c} \cdot (1 - p)^{l(\sigma)-c} \cdot \left(\frac{1-p}{p}\right)^c \]

\[ \leq \left(\frac{1-p}{p}\right)^c \cdot E[\Pi^c(R)] \]

since

\[ \Pr\{\sigma \in \Pi^c(R)\} = \Pr\{d(\sigma)\text{defining elements chosen and } c \text{ out of } l(\sigma) \text{ conflicting elements not chosen}\} \]

\[ \leq \left(\frac{1}{p}\right)^c \cdot E[\Pi^c(R)] \]
where $p = \frac{r}{n}$.
Improvements - Tail estimates ?
Improvements - Tail estimates?

\[ \text{Pr}\{\text{no segments in } S' \text{ is selected}\} \geq \left(\frac{1}{2}\right)^{\log \log n} \]

\[ = \Omega\left(\frac{1}{\log n}\right) \]

⇒ With Prob. \(\Omega\left(\frac{1}{\log n}\right)\), # of intersections is \(\Omega(n \log \log n)\)
Selecting a GOOD sample w.h.p.

Motivation: In divide-and-conquer algorithms, we are often interested in bounding the maximum size of subproblems for which we need tail estimates including the sum of subproblem sizes.

Since sample is GOOD in the expected sense, the probability that the sum of subproblems is $\leq 2 \times E[\text{sum of subproblems}]$ is $\geq 1/2$ from Markov’s inequality.

Implying

If we choose a set of $\log n$ independent samples - $R_1, R_2 \ldots R_{\log n}$, at least one is GOOD w.h.p.

How do we know which is good?
Polling: an efficient resampling technique

1. Choose $c \log n$ samples

2. Poll (sample) $S' = \frac{n}{\log^2 n}$ of the input

3. Estimate the goodness of the samples w.r.t. $S'$. Choose an $R_i$ that is good w.r.t. $S'$ (break ties arbitrarily).

**Polling Lemma** With high probability we obtain a good sample by the above procedure.
Consequences of Random Sampling

- Dynamization


- Improved bounds for important combinatorial measures like $k$-sets.
Randomized Incremental Construction (RIC)

Starting from an empty set

Repeat:

1. Insert the next object
2. Update the partial construction (data-structures)

Total Time = \( \sum_i \) Time to insert the \( i \)-th object.

\( T_s(N) = \) Total time to insert a sequence \( s \). (\( s \) is good if total time is less).

Expected total time = Expected time for a Random Insertion sequence (worst case for any input of size \( n \)).
Quicksort as R.I.C.
Gradual refinement of partition

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11} \]

\[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 \]

\((-\infty, +\infty)\)
Quicksort as R.I.C.

Conflict graph

\begin{itemize}
\item \(-\infty, 4\)
\item \(4, +\infty\)
\end{itemize}
Quicksort as R.I.C.

Conflict graph

(-\infty, +\infty)

1 2 3 4 5 6 7 8 9 10 11

(-\infty, 1)
(1, 4)
(4, 8)
(8, 10)
(10, +\infty)

(-\infty, +\infty)
A general bound for RIC

Total (amortised) cost = \( O(\text{edges created in conflict graph}) \)

Edges can be deleted at most once

**General Step:** \( R \leftarrow R \cup s \) (for a fixed \( R' = R \cup s \))

Expected work (\#edges created given \( R' \)) =

\[
\sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma) \cdot \Pr\{\sigma \in \Pi^0(R \cup s) - \Pi^0(R)\}
\]

From backward analysis this probability equals \( \frac{d(\sigma)}{r+1} \). Substituting

\[
\sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma) \cdot \frac{d(\sigma)}{r+1} = \frac{d(\sigma)}{r+1} \sum_{\sigma \in \Pi^0(R \cup s)} l(\sigma)
\]

This is expected cost conditioned on \( R' = R \cup s \).
Unconditioned Cost

\[ \mathbb{E}[ \text{# edges created}] = \mathbb{E}[\mathbb{E}[ \text{# edges created} \mid R']] \]
\[ = \frac{d(\sigma)}{r+1} \Pr(R' \text{ is chosen}) \cdot \sum_{R'} \sum_{\sigma \in \Pi^0(R')} l(\sigma) = O\left(\frac{d(\sigma)}{r} \cdot \frac{n}{r} \mathbb{E}[\Pi^0(R \cup s)]\right) \]

**A common scenario** \( \mathbb{E}[\Pi^0(R)] = O(r) \).

Total expected cost of RIC = \( \sum_{r=1}^{r=n} O\left(\frac{d}{r} \cdot n\right) \)
\[ = O(n \log n) \text{ (applicable to convex hulls)} \]

**OPEN PROBLEM**

Tail estimates
For a fixed set of $i + 1$ object, what is the probability that the $i + 1$-st insertion affects $\sigma$?

Because of random insertion sequence, any one of the fixed set of $i + 1$ objects is equally likely to be the last inserted object (by symmetry).

What is the probability that a random deletion from $i + 1$ objects defines $\sigma$? (pretending to run backwards).

$$\frac{d(\sigma)}{i + 1}$$

Since conditional expected cost depends only on $i$ (independent of the actual set of objects)

Conditional expected cost $= (\text{Unconditional})$ expected cost
Linear programming (fixed Dim.)

Max. $X_d$

Non degenerate: Exactly $d$ constraints define the optimum.
Analysis

\( T_d(n) = \) Expected running time in \( d \) dimension for \( n \) constraints.

From backward analysis, probability that \( i \)-th insertion changes optimum is \( \frac{d}{i} \) (\( d \) constraints define optimum).

\[
T_d(i) = T_d(i - 1) + T_{d-1}(i - 1) \cdot \frac{d}{i} + O(d)
\]

By induction [Seidel]

\[
T_d(n) = O(d!n)
\]
A generic search problem

Given a set $S \subset U$, build a data structure $D$, so that we can answer a query quickly.

**Issues**

- *query time*
- *space* for $D$ (space)
- *Preprocessing time* to construct $D$

**Dynamic version**

- **Insert** update $D$ for $S \cup x$, $x \in U - S$
- **Delete** update $D$ for $S - x$

Ideal goal is to match the static performance and minimize update times.
Arrangement Searching

**Problem:** Given $N$ lines (planes, hyperplanes), build data structure to do point location (report the face it lies in)
Arrangement Searching cont’d

Dynamic version:
Allow insertion/deletion of lines
Binary search

\[ T(n) \leq T\left(\frac{n}{2}\right) + O(1) \]

Approximate split:

\[ T(n) \leq T(\alpha n) + O(1) \]

where \( \alpha \) is a constant < 1 (independent of \( n \))
Binary search cont’d

Random split :

\[ T(n) \leq T(x) + O(1) \]

where \( x \) is random variable uniformly distributed in \([1..n]\)

\[ \Pr[T(n) > c \log n] < \frac{1}{n} \]

Examples : • Randomized search trees  
• Quick sort
Review of randomized search

Simple binary search

\[ T(n) = T\left(\frac{n}{2}\right) + O(1) \]

Approximate Split

\[ T(n) \leq T(\alpha \cdot n) + O(1), \alpha < 1 \]

Random Split

\[ T(n) \leq T(X) + O(1) \]

\( X \) is a r.v. \( \in [1 \ldots n] \)

\[ \Pr[T(n) > c \log n] < \frac{1}{n} \]

Randomized Search Trees/Quicksort
Generalizing the randomized search tree

- Choose a *good* sample $R$ of size $C$.

- Split the input using $R$ and build the data structure recursively for each subset.
  
  $R$ is good if each subproblem size is less than $n/2$

- $C$ is large enough such that $R$ is good with probability $\geq 1/2$, i.e. expected number of repetition $\leq 2$.

- Height of data structure is $O(\log n)$, so search time is $O(\log n)$

- **Space** Fragmentation makes it super-linear space. In this case $O \left( C^2 \frac{n}{C} \log C \right)$ or

  $$O(nC \log C)$$
From triangles to triangles
Reviewing skip list
Reviewing skip list cont’d
Reviewing skip list cont’d
A slightly different version

Choose an element to be in the sample with probability 0.5

\[ \text{Expectation}[L_i] = 2 \]

**PUGH** \[ \text{Exp.}[\text{Total}] = O(\log n) \]

**Improvement with careful analysis** [Sen 91]:

Total < \( c \log n \) with probability \( 1 - \frac{1}{n^c} \)
Overall structure

S

S₁

Interaction
Random
Sampling

Descendence
Oracle
Overall structure

S

S

S

S

Search

Data Structure

Interaction
Random Sampling

Descendence
Oracle
Descendence oracle for skip-lists

For each level $L_i$, maintain a linked-list of elements of $L_{i-1}$ that an interval $[a, b], a, b \in L_i$ intersects.
Descendence oracle for arrangement searching

How many triangles of level \( i \) intersect a triangle of level \( i - 1 \)?

\[
O \left( \frac{n}{r} \right) \quad \text{whp from Random sampling lemma}
\]

Descendence oracle can be a simple data structure storing the intersections between triangles of successive levels and takes time \( O(\log n) \) w.h.p.

implying \( O(\log^2 n) \) time w.h.p for overall query.
Updates
Update
Size of zone is $O(n)$, i.e. only $O(n)$ triangles must be updated at each level.
Bounds

- **Searching** \( O(\log n) \) w.h.p.
- **Update** \( O(n \log n) \) w.h.p.
- **Space** \( O(n^2) \)

Dimension \( d \) (fixed)

- **Searching** \( O(\log^{d-1} n) \) w.h.p.
- **Update** \( O(n^{d-1} \log n) \) w.h.p.
- **Space** \( O(n^d) \)

[Mu-Sen]