Polling: A Randomized Sampling Technique for Computational Geometry

presented by
Silvia Götz

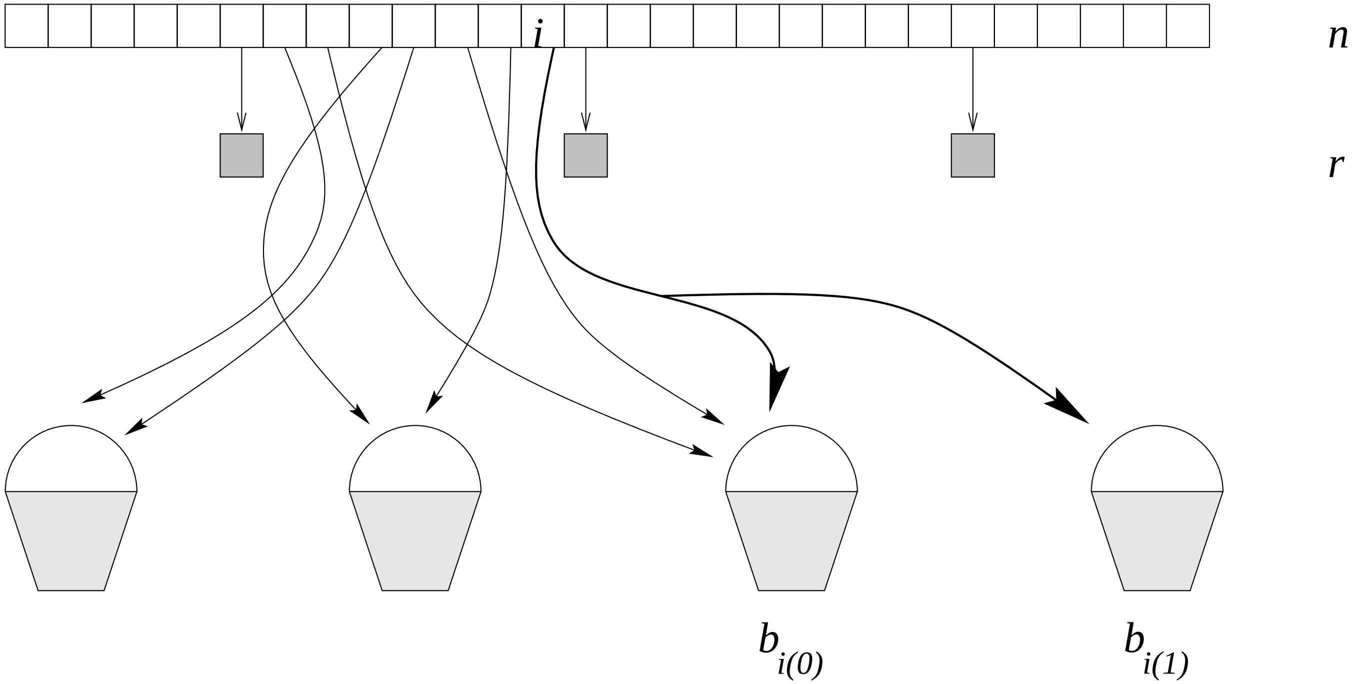
March 26, 1997

J. H. Reif, S. Sen, "Polling: A New Randomized Sampling Technique For Computational Geometry", SIAM Journal on Computing, June 1992

S. Rajasekaran, S. Sen, "Random Sampling Techniques and Parallel Algorithm Design", Synthesis of Parallel Algorithms, ed. J.H. Reif, 1993

Table of Contents

Randomized Recursive Parallel Algorithms



Running Time of a Recursive Parallel Algorithm

In general: Running time of parallel algorithm =
time between first processor starts and last processor ends
Time for one level: maximum time needed by a processor on this level

Problem: Bound the *maximal* (expected) time of the subproblems

Let ϵ_0 be such that $O(n^{1-\epsilon} \log n) = O(n^{\epsilon_0})$.
Then: max. problem size at level i is bounded by $O(n^{{\epsilon_0}^i})$

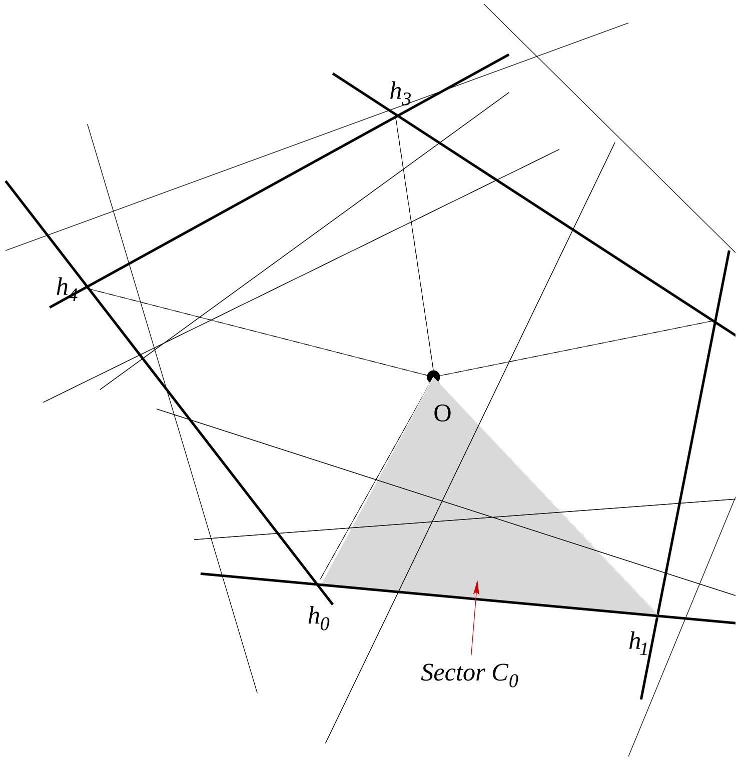
Theorem:
If a procedure at depth i in the recursion takes time T_i such that

$$Prob[T_i \geq kc\alpha(\epsilon_0)^i \log n] \leq n^{-c\alpha(1-\epsilon)^i}$$

$0 < \epsilon < 1$, then all the leaf-level procedures are finished in $\tilde{O}(\log n)$ time.

2d Intersection of n Half-Planes

dual: 2d convex hull of n points



2d Intersection of n Half-Planes (cont'd)

wlog: Assume that the origin O lies in interior of intersection
S: set of n half-planes

Algorithm IntersectHalfplane(S)

1. Choose random subset $Sample$ of $r = \lfloor n^\epsilon \rfloor$ half-planes, $0 < \epsilon < 1$
2. Costruct intersection of half-planes of $Sample$
3. Number vertices of the intersection in a cyclic order: $h_0, h_1, \ldots$
4. Consider triangles O, h_i, h_{i+1} (sectors): determine half-planes Int_i intersecting sector C_i
5. IntersectHalfplane(Int_i)
6. Output: Union of intersections over all sectors

Bounding the Problem Sizes

1. bound maximum number of half-planes intersecting any sector AND
2. bound sum of the subproblem sizes

LEMMA 1: The probability that the maximum number of half-planes intersecting any sector exceeds $O(c \cdot \frac{n}{r} \log r)$ is less than n^{-c}.

LEMMA 2: The **expected** value of the sum taken over all the sectors of all the half-planes intersecting a sector is $O(n)$.

LEMMA 3: For some constants k_{total}, k_{max}, and large n, the following conditions hold with **probability at least** $\frac{1}{2}$:

1. The maximum number of half-planes intersecting any sector is less than $k_{max} \cdot \frac{n}{r} \log r \in O(n^{1-\epsilon} \log n)$
2. The sum of half-planes taken over all the sectors of the number of half-planes intersecting a sector is less than $k_{total} n \in O(n)$.

$\longrightarrow$ called **"good example"**

Polling: The Motivation

goal: find good example w.h.p., i.e., bound sum of subproblems to $O(n)$ w.h.p.

Repeat sampling independently $m \log n$ times
$\overset{\text{Lemma 3}}{\Longrightarrow}$ all samples are bad with probability less than $(\frac{1}{2})^{m \log n} = \frac{1}{n^m} = n^{-m}$

Therefore: choose $O(\log n)$ independent sample sets, then one is "good" with high probability

Problem: determine if a sample is good requires $O(\log n)$ time
$\Rightarrow$ total $O(\log^2 n)$ time for testing all samples

Solution: determine goodness of sample by an estimation of intersections for each sector, estimation is done by taking only a **fraction of the input** half-planes

Polling: What is it ?

Idea: Determine goodness of sample by taking only a **fraction of the input** half-planes *more precisely:* chose randomly $c_0 \cdot \frac{n}{\log^d n}$, $d > 2$, c_0 a constant, of the input half-planes for **each** sample.

$Sample_j$: jth sample
Int_i^j: number of half-planes of input intersecting sector C_i of sample $Sample_j$
$IntPoll_i^j$: number of half-planes of polled input intersecting sector C_i of $Sample_j$

$\Rightarrow$ $IntPoll_i^j$ is a binomial random variable with parameters

- $\frac{Int_i^j}{n}$ (success prob.)
- $c_0 \cdot \frac{n}{\log^d n}$ (number of trials)

Polling: How to choose a good sample ?

Case 1: $Int_i^j \leq c \cdot \log^{d+1} n$
$\Longrightarrow n^{\epsilon} \cdot c \log^{d+1} n = o(n)$
$\Longrightarrow$ sum of the subproblems $\leq$ size of parent problem

Case 2: $Int_i^j > c \cdot \log^{d+1} n$
Chernoff bounds $\Longrightarrow$ Prob$(IntPoll_i^j \leq \alpha c_1 \frac{Int_i^j}{\log^d n}) \leq n^{-\alpha}$ and
Prob$(IntPoll_i^j \geq c_2 \alpha c_0 \cdot \frac{Int_i^j}{\log^d n}) < n^{-c_0 \alpha} < n^{-\alpha}$ (for $c_0 > 1$)

$\Longrightarrow$ With high probability for all i Int_i^j **are bounded**
from below by $LowInt^j := IntPoll_i^j \cdot \frac{log^d n}{c_0 c_2 \alpha} \in \Omega(\log^d n \cdot IntPoll_i^j)$ and
from above by $UpInt^j := IntPoll_i^j \cdot \frac{log^d n}{c_1 \alpha} \in O(\log^d n \cdot IntPoll_i^j)$

$$\boxed{\Longrightarrow \Omega(\log^d n \cdot IntPoll_i^j) \leq Int_i^j \leq O(\log^d n \cdot IntPoll_i^j)}$$

Choosing a Good Sample: The Algorithm

Input: Samples $Sample_1, \ldots, Sample_m$, $m = O(\log n)$
Output: A "good" sample $Sample_{j_0}$

/* case: clearly good */
IF $k_{total} \cdot n > UpInt^j n^\epsilon$ THEN
    accept sample $Sample^j$;

/* case: clearly bad */
IF $k_{total} \cdot n < LowInt^j n^\epsilon$ THEN
    sample is bad

/* case: choose the best */
IF $LowInt^j n \leq k_{total} \cdot n^\epsilon \leq UpInt^j n^\epsilon$ THEN
    accept sample $Sample_{j_0}$ for which $\sum_i IntPoll_i^j$ is minimum

Recall: at least one of the samples satisfies case 1 or 3 w.h.p.

Polling Lemma:

expected "good" samples $\overset{\text{polling alg.}}{\Longrightarrow}$ w.h.p. "good" sample

Bounding the Number of Processors

Good example:

sum of sizes of subproblems $= c\cdot$ size of parent problem

But: with $O(\log\log n)$ recursive levels

$\Rightarrow nc^{\log\log n} = n \cdot \log^{\log c} n \in O(n \cdot \log^b n)$, b constant

Opportunities:

- more processors
- longer running time
- bound sum of subproblems more tightly, so that it equals $c \cdot n$, c is the **same** for each level

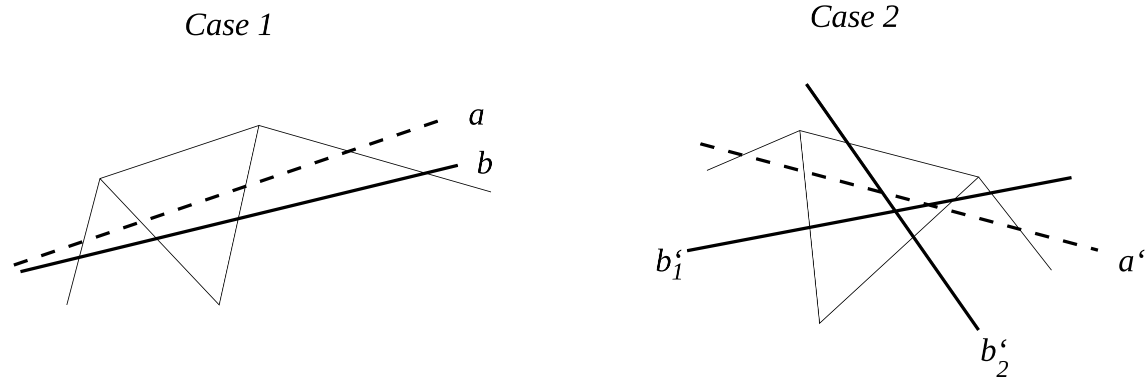
Bound the sum of the subproblems more tightly: *Filtering*

Filtering Procedure:

- after finding a good sample
- *Input:* subproblems with
  sum of the sizes $= c \cdot$ size of parent problem
- *Output:* subproblems, where at **each** level
  sum of size of subproblems = a fixed size (e.g. $O(input + output)$) $c \cdot n$
- use geometric properties of the specific problem

Intersection of half-planes: output size $\leq$ input size
$\Rightarrow$ eliminate half-planes from a sector that are not part of the output

Filtering for Intersection of Half-Planes in 2d



Case 3: Half-plane is visible in at most 1 sector
⇒ eliminated from all other sectors by case 1

Case 4: Half-plane is visible in more than 1 sector
⇒ in all but two of these sectors it eliminates all other half-planes
⇒ for intermediate sectors no recursive call is needed
⇒ in only two sectors it can contribute to the vertices of the intersection

Global view: at any stage at most 2 copies of a half-plane

Processor Allocation

Case 1 and 2: no processors

Case 3 and unsolved cases (fall into one sector): allocate 2 processors

Case 4: allocate 1 processor at each bounding sector

Global view: at any stage at most 2 copies of a half-plane

General Strategy

1. Select $O(\log n)$ samples, each of size $\lfloor n^{\epsilon} \rfloor$ for some $0 < \epsilon < 1$.
2. Select a good sample using *Polling*.
3. Divide the problem into subproblems using the good sample.
4. Use a *Filtering* scheme to bound the sum of the subproblems more tightly (e.g. output size+input size). Purpose: bound the number of processors
5. IF subproblem is small enough (usually $O(\log^k n)$) THEN
   use a direct method to solve subproblem
   ELSE call algorithm recursively

Time Complexity: Intersection of Half-Planes in 2d with Polling and Filtering

With high probability:

1. $O(\log n)$ coin flips on each processor: $O(\log n)$
2. intersection of each sample (brute-force): $O(\log n)$
   intersection of "polled" input ($\frac{1}{\log^d n}$th) with sectors built by each sample: $O(\log n)$
   (Fibonacci search on bimodal sequence)
3. intersection of (whole) input with sectors built by "good" sample: $O(\log n)$
   (Fibonacci search on bimodal sequence)
4. and 5. (base case): $\tilde{O}(\log n)$

an iteration with problem size n: $\tilde{O}(\log n)$
$\overset{\text{Theorem}}{\Longrightarrow}$ $\tilde{O}(\log n)$ with $2n$ processors on a CREW PRAM

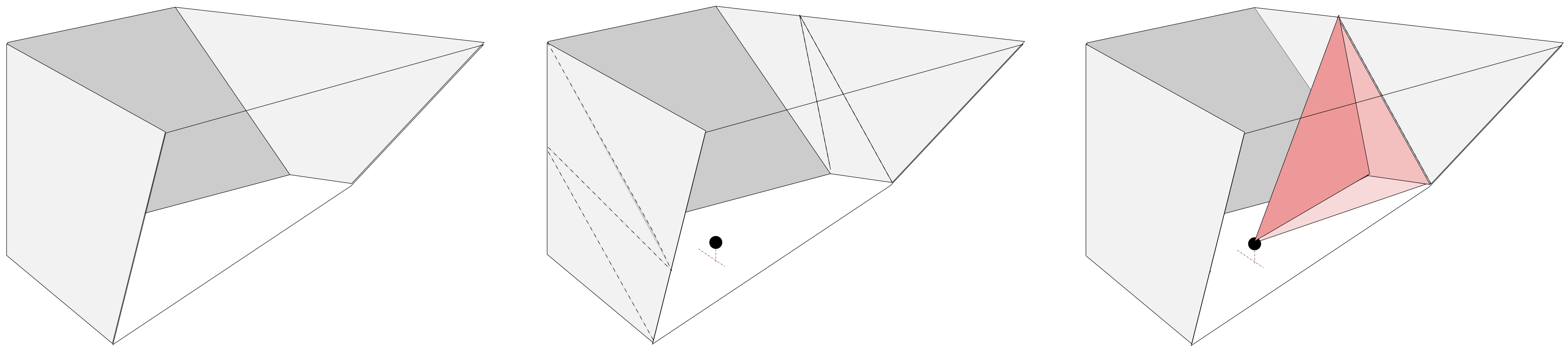
Another Application for Polling: Intersections of Half-Spaces in 3d

dual: 3d convex hull
also solvable:

- *Voronoi diagram*
- *all-points nearest neigbor*
- *Euclidean minimal spanning tree*

Results: *The above problems can be solved in* $\tilde{O}(\log n)$ *time on a* $O(n)$ *processor CREW PRAM.*

Partitioning by the Sample: the Cones



Conclusions and Open Problems

New Random Sampling Technique: Polling

$\Rightarrow$ *determine a "good" sample out of $O(\log n)$ samples by testing only a fraction of the input*

$\Rightarrow$ *bound sum of subproblems more tightly by* Filtering *(problem dependent)*

2d and 3d convex hull, intersection of half-spaces, 2d Veronoi diagram could be solved in $\tilde{O}(\log n)$ time on a CREW PRAM with $O(n)$ processors.

Open Problems:

- *applicabality of method to other problems*
- *algorithms for EREW PRAM*