Chan's randomized optimization technique

- T. M. Chan, "Geometric Applications of a Randomized Optimization Technique," Discrete and Computational Geometry, vol. 22, pp. 547–567, 1999. (We teach first three sections)
- For certains geometric problems, the technique can turn a deterministic algorithm for the decision version into a randomized algorithm for the optimization version.

Decision Problem:

- Given an instance I and a value k, answer if there exists a solution for I whose value is k, at most k, or at least k.
- E.G.: Given a set I of points in the plane and a value k, does there exist a spanning tree connecting all points in I whose length is at most k?

Optimization Problem:

- Given an instance I, answer a solution for I with the minimum or maximum value.
- E.G.: Given a set I of points in the plane, find a spanning tree connecting all points in I with the minimum length.

Importance of the Technique

- It is usually easier to develop an algorithm for the decision version of a problem than one for the optimization version.
- An algorithm for the decision version is probably a bit simpler, i.e., easier for implementation
- Expected behavior of an algorithm usually reflects its actual behavior, i.e., the worst case hardly occurs.

Finding the minimum of r numbers, i.e., $\min\{A[1], A[2], \ldots, A[r]\}$

Algorithm RAND-MIN

1. randomly pick a permutation $\langle i_1, \ldots, i_r \rangle of \langle 1, \ldots, r \rangle$
2. $t \leftarrow \infty$
3. for $k = 1, \ldots, r$ do
4. if $A[i_k] < t$ then (*decision*)
5. $t \leftarrow A[i_k]$ (*evaluation*)
6. return t

$O(Dr + E \log r)$ expected time

- Imagine $A[0], \ldots, A[r]$ have not yet been precomputed
- D: time to decide if $A[i] < t$
- E: time to evaluate $A[i]$
- The expected number of times that step 5 is execuated is $\ln r + 1$. (Exercise)
- $O(Dr + E \log r)$. If $E >> D$, it is better than $O(Er)$.

Consider an instance I with n elements for a minimization problem. Let $A[I]$ be the cost of the minimal solution for I. Assume we can randomly partitaion I into r subsets with almost equal size, $I_1, \ldots, I_r$ such that $A[I] = \min\{A[I_1], \ldots, A[l_r]\}$.

- if $A[l_i] < t$: a decision problem
- $t \leftarrow A[l_i]$: an optimization problem
- $O(D(n/r) * r + E(n/r) * \log r)$
  - $D(m)$: time to solve the decision problem for an m-size input
  - $E(m)$: time to solve the optimization problem for an m-size input

Denotation and Assumption

- Γ represent the *problem space*
- Given a problem $P \in \Gamma$, let $w(P) \in \mathbb{R}$ be its solution
- $|P|$ is the size of P (a positive integer)
- The solution of a problem of constant size can be computed in constant time.

Lemma *Chan's randomized technique*
Let $\alpha < 1$, $\epsilon > 0$, r be constants, and let $D(\cdot)$ be a function such that $D(n)/n^\epsilon$ is monotone increasing in n. Given any problem $P \in \Gamma$, suppose that within $D(|P|)$ time,

(i) we can decide whether $w(P) < t$ for any given $t \in \mathbb{R}$, and

(ii) we can construct r subproblems, $P_1, \ldots, P_r$, each of size at most $\lceil \alpha |P| \rceil$, so that

$$w(P) = \min\{w(P_1), \ldots, w(P_r)\}.$$

Then for any problem $P \in \Gamma$, we can compute the solution $w(P)$ in $O(D(|P|)$ expected time

Proof
General Idea

- Compute $w(P)$ by applying Algorithm Rand-Min to the **unknown** numbers $w(P_1), w(P_2), \ldots, w(P_r)$.
- Deciding $w(P_i) < t$ takes $D(|P_i|)$ time.
- Evaluating $w(P_i)$ is done recursively until $|P_i|$ drops below a certain constant.

Analysis

- let $T(P)$ be the random variable corresponding to the time needed to compute $w(P)$.
- Let $N(P_i)$ be 0-1 random variable, having value 1 if and only if $w(P_i)$ is evaluated

$$T(P) = (\sum_{i=1}^{r} N(P_i)T(P_i)) + O(rD(|P|)).$$

Note that the expected number of evaluations by Algoirthm RAND-MIN is $E[\sum_{i=1}^{r} N(P_i)] \leq \ln r + 1$

- Define $T(n) = \max_{|P| \leq n} E[T(P)]$.

Since $N(P_i)$ and $T(P_i)$ are independent, we have

$$E[T(P)] = \sum_{i=1}^{r} E[N(P_i)]E[T(P_i)] + O(rD(|P|))$$
$$\leq (\ln r + 1)T(\lceil \alpha |P| \rceil) + O(rD(|P|))$$

Which implies

$$T(n) = (\ln r + 1)T(\lceil \alpha n \rceil) + O(D(n)).$$

($O(rD(|P|)) = O(D(n))$ since r is a constant)

If we assume,

$$(\ln r + 1)\alpha^{\epsilon} < 1,$$

$T(n) \leq C \cdot D(n)$ for an appropriate constant C depending on α, r, and ϵ. (Exercise)

To enforce $(\ln r +)\alpha^{\epsilon} < 1$, we compress l levels of the recursion into one before appying Algorithm Rand-Min, where l is a sufficiently large constant. Then,

- r increases to r^l
- α decreases to α^l
- $\lim_{l \to \infty} (\ln r^l + 1)\alpha^{l\epsilon} = 0$

Note:
The above lemma still holds if (i) and (ii) require $D(|P|)$ expected time (rather than the worst-case).

Applications

Closest Pairs

- Let U be a collection of objects.
- Given a distance function $d : U \times U \rightarrow \mathbb{R}$,
  - *closest-pair* problem: to compute $w(P) = \min_{p,q \in P} d(p, q)$ for a given set $P \subset U$
  - *closest-pair decision* problem: to determine whether $w(P) < t$ for a given P and $t \in \mathbb{R}$.

Theorem.
If the closest-pair decision problem can be solve in $D(n)$ time, then the closest-pair problem can be solved in $O(D(n))$ expected time, assuming that $D(n)/n$ is monotone increasing.

- Arbitrarily partition P into three subsets P_1, P_2, P_3 of roughly equal size.

$$w(P) = \min\{w(P_1 \cup P_2), w(P_2 \cup P_3), w(P_1 \cup P_3)\}$$

- Applying the technique with $r = 3$ and $\alpha = \frac{2}{3}$.

Ray Shooting

- Let U be a collection of objects
- Let V be a collection of rays
- Let $\tau : U \times V \to \mathbb{R}$ be an ordering function, where $\tau(p_1, q) < \tau(p_2, q)$ means that ray q hit object p_1 before p_2.
- The *ray shooting* problem: to preprocess a given set $P \subset U$ of size n into a data structure that answers queries of the following type:
  - given $q \in V$, compute $w(P, q) = \min_{p \in P} \tau(p, q)$.
- The *ray shooting decision* problem: given any $q \in V$ and $t \in \mathbb{R}$, determine whether $w(P, q) < t$.

Theorem
If the ray-shooting decision problem can be solved with $P(n)$ preprocessing and $D(n)$ query time, then the ray-shooting problem can be solved with $O(P(n))$ preprocessing and $O(D(n))$ expected query time, assuming that $P(n)/n^{1+\epsilon}$ and $D(n)/n^{\epsilon}$ are monotone increasing forsome constant $\epsilon > 0$

proof

- Parition P into two subset P_1 and P_2 of roughly equal size, build the decision data structures for P_1 and P_2, and recursively preprocess P_1 and P_2.
- The new preprocessing time $P'(n)$ satisfies the recurrence

$$P'(n) = 2P'(n/2) + O(P(n)).$$

- If $P(n)/n^{1+\epsilon}$ is monotone increasing, $P'(n) = O(P(n))$
- To compute a given $q \in V$, we can divide the problem into two subproblems, each of size roughly $n/2$:

$$w(P, q) = \min\{w(P_1, q), w(P_2, q)\}$$

.

- Chan's technique implies the expected query time to be $O(D(n))$.