Adaptive Analysis of Algorithms

Vlad Estivill-Castro

March 29, 2010
Introduction
  Motivation
  Adaptive Algorithm

Instance Easiness
  measures of disorder
  ranking measures of disorder

Adaptivity in general
  Models
  Links to parameterized complexity

Adaptivity in Clustering

Coda
Sorting is a core problem

Central to the debate about models of computation

- comparison-based vs sorting integers
- worst-case vs expected case (maybe best case)
- lower bounds and optimality ($O()$, $\Omega()$, $\Theta()$).
- problems vs algorithm
- internal memory vs external memory
- parallel vs sequential
An ideal case to initiate students on the analysis and design of algorithms

- (and data structures).
- theoretical and experimental algorithmics
- algorithmic engineering (Quicksort / Insertion Sort)
A focus on the instances

A-Sort [3] seems to be the origin of the notion of ‘adaptive’ [2].

- Verifying an input sequence is sorted is $\Theta(n)$ time.
- Sorting (comparison-based) is $\Theta(n \log n)$.
- Both statements can be seen as remarks about the expected case (just the distribution of instances is extreme).

Should not need to do as much work if there is only a bit of disorder to remove.
A bi-dimensional (multi-dimensional) view on algorithm complexity

Adaptive algorithm
- (originally not a view on problem complexity)
- the complexity of the algorithm is a smoothly growing function
  - of a measure of instance-hardness (disorder)
  - the size of the input
Inversions

- number of inversions.
- Let $\text{Inv}(\pi) = \text{Inv}(\pi, \text{Id})$ (or Kendall-Tau) [distance, measure of disorder, measure of pre-sortedness, right-invariant metric $\text{Inv}(\pi, \sigma) = \text{Inv}(\pi \circ \tau, \sigma \circ \tau)$]

\[
\text{Inv}(X = \langle x_1, x_2, \ldots x_n \rangle) = \| (i, j) | i < j \text{ and } x_i > x_j \|
\]

Minimum number of adjacent swaps to bring the sequence into sorted order.
Illustration
Insertion Sort

**Straight Insertion Sort** (the insertion data structure is an array)

- \( \text{Inv}(x) + n - 1 \) comparisons
- \( \text{Inv}(x) + 2n - 1 \) data moves

Improve the data structure (just place a finger and count only comparisons)

\[
n \log(1 + \text{Inv}(X)/n).
\]
Lower bounds

1. $\text{below}(z, n, M) = \{ X \in S_n \mid |X| = n \text{ and } M(X) \leq z \}$
2. In the comparison-based model of computation the comparison tree has height at least $\Omega(\log \| \text{below}(z, n, M) \|)$.

Optimal adaptivity in the worst-case

$$T_s(X) \in O(\max\{|X|, \log \| \text{below}(z, n, M) \| \}).$$
Instance easiness can be measured in many ways

Operational

- **Exchanges** (swaps) – minimum number of exchanges to bring the sequence into sorted order.
- **Rem** – minimum number of removals to eave something sorted
- **Runs** (step downs) — passes for external sort
$M_1$ is algorithmically finer than $M_2$ if and only if whenever $A$ is optimal adaptive with respect to $M_1$, then it is also optimally adaptive with respect to $M_2$. 
Illustration

- Longest inversion = max displacement
- # runs
- # swaps
- # inversions
- # ascending subseq.
- "Melsort"
- # monotone subseq.
- Moffat & Petersson
- "oscillation"
- # removals

measures of disorder
ranking measures of disorder
Where things were left

- Optimal algorithm (comparisons) for finest measure of disorder [Moffat & Petersson]
- Does there exist a minimal element for the hierarchy?
- Does there exist an optimal algorithm for the optimum?
Adaptivity — Expected case

- Makes perfect sense for randomized algorithms
- Expectation [ required resources ] (time/space) is a smoothly growing function of the instance easiness.
### Adaptive Analysis

| $|X|$ vs $M(X)$ | 0 | 1 | 2 | ... | $k$ |
|--------------|---|---|---|-----|-----|
| 1            |   |   |   |     |     |
| 2            |   |   |   |     |     |
| 3            |   |   |   |     |     |
| 4            |   |   |   |     |     |
| ...          |   |   |   |     |     |
| $n$          |   |   |   |     | $f(n, k)$ |

Objectives

- $f(n, k)$ monotonically increasing for each fixed $n$
- proportional to
  
  $$below(z, n, M) = \{X \in P \mid |X| = n \text{ and } M(X) \leq k\}$$
Parameterized Complexity

| $|X|$ vs $k$ | 0 | 1 | 2 | ... | $k$ |
|---|---|---|---|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| ... | | | | | |
| $n$ | | | | | $pol(n)f(k)$ |

Objectives

- Understand the frontier of hardness
- avenue to break intractability
Adaptivity in NP Problems

- $\text{below}(z, n, M) = \{ X \in P \mid |X| = n \text{ and } M(X) \leq z \}$
- very close notion to parameterized complexity
- $z$ is the parameter, $M$ is the function of instance easiness (does this lead to the next chapter in parameterized complexity?)
- recall the argument about hierarchies of measures
Contrast between adaptive algorithms and parameterized algorithms

**Vertex Cover**

Let \( G = (V,E) \), and we measure instance easiness as

\[
\sum_{\text{Connected Component } C} \sum_{i=1}^{n-1} i \cdot \# \text{ vertices of degree } i
\]
Adaptivity vs parameterization

- Notion of measure of instance easiness (could be the parameter)
- The maximum value of the measure is $k << n$.
- Adaptivity seems to make more sense in the class $P$.
- Adaptive makes sense for any resource (time, number of processors, space, number of messages) proportional to instance easiness.
Adaptivity vs parameterization

Illustration

- Measures of structural simplicity
- Tree-Width (how tree-like), Path-width (how Path-like), genus (how planar-like).
- “Decision” version vs “Optimization” version
- Tricks also used in the adaptive case (because computing the measure may be as hard as solving the problem).

1. For a scheme $k = 0, \ldots, \max\{M(X)\}$, apply algorithm for $M(X) = k$.
2. If $A_1$ and $A_2$ are two algorithms, respectively optimal with respect to measures of easiness $M_1$ and $M_2$, then an algorithm that runs them alternatively is optimum with respect to both measures.
Distance-based and Representative-based Clustering

- Given $X$ set of $n$ points (vectors $\vec{x}_i \in \mathbb{R}^d$) find a partition $C_1, C_2, \ldots C_k$ ($\bigcup C_i = X$) that minimizes the loss (error).

- Total square error: Find representatives $\vec{c}_1, \ldots \vec{c}_k$ such that

$$\sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \text{dist}(\vec{x}, \text{rep}[C_j])^2$$

- Total error: Find representatives $\vec{c}_1, \ldots \vec{c}_k$ such that

$$\sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \text{dist}(\vec{x}, \text{rep}[C_j])$$

- Medoids (discrete case): $\vec{x}_i \in X$. 
Geometric difference of criteria
A consensus problem

The case \( k = 1 \) and \( \text{dist}=\text{Euclid} \).

- Total square error:
  \[
  \sum_{\vec{x} \in C_j} \text{dist}(\vec{x}, \text{rep}[C_j])^2
  \]

  - Solution is center of mass (Minimizes distortion).

- Total error:
  \[
  \sum_{\vec{x} \in C_j} \text{dist}(\vec{x}, \text{rep}[C_j])
  \]

  - Fermat-Webber Problem (Geometric Median).
  - No solution by digital computers.

- Discrete case: \( C \subset X \)
  - Problem is in \( XP \) (Test all subsets of size \( k \)).
Status of clustering from adaptive analysis

Clusterability

- Notions of how easy (instance easiness) is to cluster a particular instance $X$ in $k$ clusters [1]
  1. Center-perturbation clusterability.
  2. Worst-pair-ratio clusterability.
  5. Clusterability to a target cluster.

- Type A Results: Clusterability for one notion may not mean clusterability for the other.

- Type B Results: If an instance has high clusterability for one measure, then it is ”polynomial” to find a ”good” clustering.

- Type C Results: Computing ”clusterability” is NP-Hard
Center-perturbation clusterability

Center-based clustering

- Centers (and thus clusterings) \( \{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_k\} \) are \( \epsilon \)-close to \( \{\vec{c}'_1, \vec{c}'_2, \ldots, \vec{c}'_k\} \) if \( \forall j \|\vec{c}_j - \vec{c}'_j\| \leq \epsilon \).

- \( X \) is \( (\epsilon, \delta, k) \)-clusterable (for \( k \geq 1 \) and \( \epsilon, \delta \geq 0 \)) if \( \forall C \) a center-based clustering of \( X \) that is \( \epsilon \)-close to some optimal clustering
  \[ \mathcal{L}(C) \leq (1 + \delta) \text{OPT}_{\mathcal{L},k} (X). \]

Illustration:

1. \( \mathcal{L}(C) = \sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \text{Euclid}(\vec{x}, \text{rep}[C_j])^2. \)
2. \( \text{OPT}_{\mathcal{L},k} (X) = \min \{ \mathcal{L}(C) \mid C \text{ is clustering of } X \} \).
Type B Result:

If $X$ is $(\text{rad}(X)/\sqrt{l}, \delta)$-center perturbation clusterable, then there is an algorithm that runs in polynomial time in $n$ and outputs a cluster $C$ so that

$$\mathcal{L}(C) \leq (1 + \delta)OPT_{\mathcal{L},k}(X).$$

- Complexity is actually $O(n^{lk})$, i.e. polynomial only for fixed $k$ (and fixed $l$).
- $\text{rad}(X)$ is the radius of the minimum sphere that contains $X$.
Algorithm:

1. $C_A \leftarrow$; $L \leftarrow$ all $k$ tuples with entries an $l$-sequence of elements of $X$. /* A sample with replacement of $kl$ elements from $X */$

2. for each element of $L$:
   2.1 find the center of mass $c_j$ of each $l$-sequence
   2.2 find the clustering $\hat{C}$ induced by the $c_j$’s (Voronoi partition)
   2.3 if $C_A = \text{or} \ L(\hat{C}) < L(C_A)$, then $C_A \leftarrow \hat{C}$.

3. return $C_A$
Worst-pair-ratio clusterability

- For clustering $C$ of $X$,

$$\text{split}(C) = \min\{\text{dist}(\vec{x}, \vec{y}) \mid \vec{x} \in C_i, \vec{y} \in C_j, i \neq j\}$$

$$\text{width}(C) = \max\{\text{dist}(\vec{x}, \vec{y}) \mid \vec{x} \in C_i, \vec{y} \in C_i\}$$

- “Cluster-quality” of a clustering $C$ with respect to $X$

$$WPR(C, X) = \frac{\text{split}(C)}{\text{width}(C)}.$$  

- $WPR_k$ clusterability

$$WPR_k(X) = \max\{WPR(C, X) \mid C \text{ is } k \text{ clustering of } X\}.$$
Type B Result:

If $WPR_k(X) \geq 1$ for some $k > 2$, we can find a $k$-clustering $C$ with maximum split over width ration in $O(n^2 \log n)$ time where $n = |X|$.

1. Algorithms is single-linkage clustering until $k$ components.

2. Correctness: If there is a clustering $C$ (with $k$ non-trivial clusters!) such that $\text{width}(C) < \text{split}(C)$, then there is only one such clustering.
Separability clusterability

Drop in loss relative to number $k$ of clusters.

- The $k$-means loss

$$\mathcal{L}_k(C, X) = \sum_{j=1}^{k} \sum_{\bar{x} \in C_j} \text{Euclid}(\bar{x}, \text{rep}[C_j])^2.$$

- The set $X$ is $(k, \epsilon)$-separable if

$$\text{OPT}_C \text{ is } k \text{ clustering}[\mathcal{L}_k(C, X)] \leq \epsilon \text{ OPT}_{C'} \text{ is } k - 1 \text{ clustering}[\mathcal{L}_{k-1}(C', X)]$$

- The separability $S_k(X) \in [0, 1)$ is

$$\inf\{\epsilon > 0 \mid X \text{ is } \epsilon - \text{separable}\} \text{ (smaller value, easier to cluster).}$$
Type B Result:

If $X$ is $(2, \epsilon^2)$-separable, then a 2-clustering with $k$-means loss

$$\mathcal{L}_k(C, X) \leq \frac{\text{OPT}_C \text{ is 2 clustering} \left[ \mathcal{L}_2(C, X) \right]}{1 - \rho}$$

can be found with probability $1 - O(\rho)$ in time $O(dn)$ where $\rho = \Theta(\epsilon^2)$.

1. Approximation algorithm.
2. Probabilistic algorithm.
3. Theoretical algorithm.
Variance-ratio Clusterability

- Variance of $X = \sigma^2(X) = \frac{1}{\|X\|} \sum_{\tilde{x} \in X} \|\tilde{x} - \text{mean}(X)\|^2$.
- $k$-clustering $C = \{X_1, X_2, \ldots, X_k\}$, proportion $p_i = \|X_i\|/\|X\|$.
- Between cluster variance
  
  $$B_C(X) = \sum_{j=1}^{k} p_i \|\text{mean}(X_i) - \text{mean}(X)\|^2.$$  

- Within cluster variance $W_C(X) = \sum_{j=1}^{k} p_i \sigma^2(X_i)$.
- Variance-Ratio Clusterability
  
  $$VR_k(X) = \max_{C \text{ is a } k \text{ clustering}} \frac{B_C(X)}{W_C(X)}.$$
Type B Result:

Observations:

- \( \sigma^2(X) = W_C(X) + B_C(X) \).
- \( nW_C(X) = k\text{-means loss} = \mathcal{L}_k(C, X) \).

Therefore, \( VR_2(X) = \frac{1}{s_2(X)} - 1 \) for all \( X \).

- Equivalence of measures of clusterability for \( k = 2 \).
- Algorithms for separability also work for variance-ratio.
More common in the context of a graph $G(V, E)$.

- Sometimes weights for edges $w(e)$ with $e \in E$.
- $r$-Dominating Set
  - Is there a set $C \subset V$ of size $k$ ($\|C\| = k$) so that $\forall v \in V$ there is $c \in C$ so that $\text{dist}(v, c) < r$.
  - Vanilla Dominating Set ($w(e) = 1$, $\forall e$ and $r = 1$) is unlikely to be FPT;
  - but FPT for special cases (planar).
- However, few implementations.
Connexion between adaptivity and parameterized complexity

Determine the complexity:

- **Instance**: A set $X$ of $n$ vectors with “measure” of clusterability $k$.
- **Parameter**: $k$.
- **Question**: Does $X$ have a clustering of “quality” $k$.

Investigate combinations of “measures” and “quality” (or is the problem trivial).
Produce adaptive algorithms (optimization version).
Specific Open Problem

In the context of a graph $G(V,E)$.

- Sometimes weights for edges $w(e)$ with $e \in E$.
- $r$-DOMINATING SET
- Is there a set $C \subset V$ of size $k$ ($\|C\| = k$) so that $\forall v \in V$ there is $c \in C$ so that $\text{dist}(v, c) < r$.
- In the special case the the clusterability is high (for example, the worst-pair-ratio larger than 1 implies polynomial time).
- FPT? where the parameter $k$ is (inversely) related to the clusterability (Conjecture: FPT for instances with clusterability larger than $1/k$).
- Deliver good implementations.
Adaptivity, does it matter?

- For sorting, the additional machinery usually causes too much overhead.
- Closest to best engineered approach (carefully engineer Quicksort until instances are small enough, then apply a pass of Insertion sort).
- Eradicate Bubble Sort
Hierarchically finest measure

- For measures of disorder
- For competitive algorithms (Lopez-Ortiz discussion on LRU)
  - need suitable model of optimality
- For other environments of adaptive algorithms
  - Shortest Path (Dijkstra)
Other problems

Erik Demaine and several others

- Searching
- Sets
- Curves
- Integrals
Finding $M(X)$

- Are there problems where finding (approximating) $M(X)$ can be done much faster than actually solving $P$.
- A slight variation like the local-search parameterized complexity that should have sense for NP-Complete problems.
Thanks
Question?
