Multi-Dimensional Divide and Conquer

Subhash Suri

September 28, 2015

1 A General Technique for Multi-Dimensional Data

- Article in Communications of the ACM, April 1980, by Jon Bentley.
- A *paradigm*, as opposed to a single-problem algorithm. A good illustration of the style of spatial thinking used in design of geometric algorithms.
- Problems dealing with N points in k-dim space. Models "N database records with k keys each," "Statistical data: N samples of k-variable data," etc.
- A few sample problems.
  - Ranking, Pareto, ECDF
  - Search structures for ECDF, Maxima
  - Closest pair, fixed-radius NN searching, etc.

Ranking Problem.

- Consider two k-dim points p and q. We will use p_i and q_i to denote their ith coordinates. (For ease of presentation, assume all coordinates are distinct; otherwise use an appropriate tie-breaking rule.)
- We say $p > q$, that is, p *dominates* q, if $p_i > q_i$, for every dim i, $1 \leq i \leq k$.
- If neither point dominates the other, we call them *incomparable.*
- Motivation: Imagine each coordinate being a desirable *attribute* (cost, distance, quality, runtime), and the points being *alternatives* (restaurants, cars, hotels, driving routes, software performance etc).
- If alternative p is better than alternative q in **all** attributes (dimensions), then there is never a good reason to choose q over p. In other words, p is dominant over q.

Pareto Optima.

- Given a set S of N points in k-dimensions, the subset of S that is **not dominated** by any other have a special status, and is known by various names in different scientific communities:

  1. Maxima (CG),
  2. Pareto Set (optimization, economics)
  3. Skyline (Databases)

  EXAMPLE 1.

Rank Ordering. More generally, we can explore the "extent" to which a point is dominant.

- Given a set S of N points, define the $rank(s)$ to be the number of points in S that are dominated by s.
- In 1D, the rank is determined by the sorted order: the smallest item has rank 0 the largest one has rank $N - 1$.
- But in 2 and higher dimension, there is no natural sorting order, and the best one can hope is the *partial order* given by domination.
- In statistics, the ranking describes what is called ECDF (Empirical Cumulative Distribution Function) and is a very common problem today in data analysis.

The Algorithmic Problem.

- A naive method computes each point's rank in $O(N)$ time, by brute force comparison with all other points. This takes $O(N^2)$ time to compute all the ranks, and to compute the pareto set.
- However, the amount of information computed per point is $O(1)$, so can one do this in better than N^2 time?
- In 1D, sorting algorithms accomplish this. What about k-dim?

2 Solving the Rank Problem in 2D

The Multi-dimensional D & C algorithm is best illustrated first in 2D. The algorithm has the following high level structure:

- Divides the input set S into two (roughly equal) halves
- Recursively compute the rank for each subproblem
- Conquer step works on the entire set and corrects the rank (to account for the half not considered)

Example Figure 2 shows ranks of all points.

- The Divide Step (shown in Figure 3a). Choose a vertical line L with $N/2$ points on each side. Call these subsets A and B.
- The Recursive Step (shown in Figure 3b). Compute the rank of each point in A among just the points of A. Similarly for B.
- The Conquer Step (shown in Figure 3c). Every point in A has smaller x-value than any point of B, which leads to two important observations.
  1. No point in A can dominate any point of B, and
  2. A point $b \in B$ dominates a point $a \in A$ if and only if b's y-value is larger a's y-value.
- By (1), the ranks of all the points in A (recursively computed) are already correct.
- The ranks of points in B need correction but by (2) this update problem is now a 1D problem:

  > *for each point $b \in B$, we just need to compute how many point in A have y-value smaller than b's.*

- To do this, we simply project all the points in A and B (together but keeping track of their *type*) onto the y-axis (same as projecting on L). Now simply scan all the points up the line L, and keep track of how many A points we have seen. Whenever we encounter a B point, we simple increment its rank by the running tally.

  See Fig. 3c.
- Correctness (by induction). Assuming the recursive halves A and B are correct, the conquer step correctly updates the ranks.

- The Runtime Analysis.

  The recurrence for the algorithm is:

$$T(N) \;=\; 2T(N/2) + O(N) + O(N \log N) \;=\; O(N \log^2 N)$$

- An improvement. An easy trick improves the running time. The costly step of the conquer was the sort. But since all dividing lines will be parallel to L, we can simply pre-sort S by y-values in the beginning, so that each subset A and B can maintain its y-sorting during the divide processing.

$$T(N) \;=\; 2T(N/2) + O(N) \;=\; O(N \log N)$$

3 3-Dimensional Divide-and-Conquer

- The high level structure is identical. We divide S into A and B using a vertical plane L. Solve the subproblems A and B recursively. The Conquer step then updates the ranks of B.

- The Conquer Step has the same special structure as before:

  1. the ranks of all the points in A are correct (each of them has a smaller x-value than B, and therefore cannot dominate a B point).
  2. In order for $b \in B$ to dominate any $a \in A$, it must be that b dominates a on both y and z-coordinates.

- But the latter is just a 2-dimensional ranking problem:

  > *for each point $b \in B$, we want to compute* **how many** *points of A does it dominate in both y and z?*

- Therefore, project all the point of S onto the plane L (the yz-plane), and run a slightly modified version of the 2D problem.

- The modification is simply that when computing the rank, we only count the A points being dominated (although A and B are together projected onto the plane). This requires just some minor book-keeping.

- The recurrence for the algorithm is (the conquer step solves the 2D problem):

$$T(N) \;=\; 2T(N/2) + O(N \log N) \;=\; O(N \log^2 N)$$

4 k-Dimensional Divide-and-Conquer

- The method generalizes in straightforward way to k-dimensions. The conquer step requires solving a $(k-1)$-dimensional subproblem, which by induction we know how to solve in $O(N \log^{k-2} N)$ time.
- The general recurrence can be written as:

$$T(N,k) \;=\; 2T(N/2,k) + T(N,k-1) + O(N)$$

- Stopping condition: $T(N,2) = O(N \log N)$.
- This solves to

$$T(N,k) \;=\; O(N \log^{k-1} N)$$

5 Pareto Optimal or Maxima Problem

Compute the subset of S that is not dominated by any other point in S.

Figure 7.

- A simple trick reduces this to the ranking problem.
- Transform the input set of points S into a *negated set* S', where each coordinate is multiplied by -1.
- This simply reflect all the points around the origin.
- A point s is a maxima in S **iff** the corresponding point s' has rank 0 in S'. (If it is dominated by no point in S, its negation dominates no one in S').