Skyline Computation

Presentation for CSG399

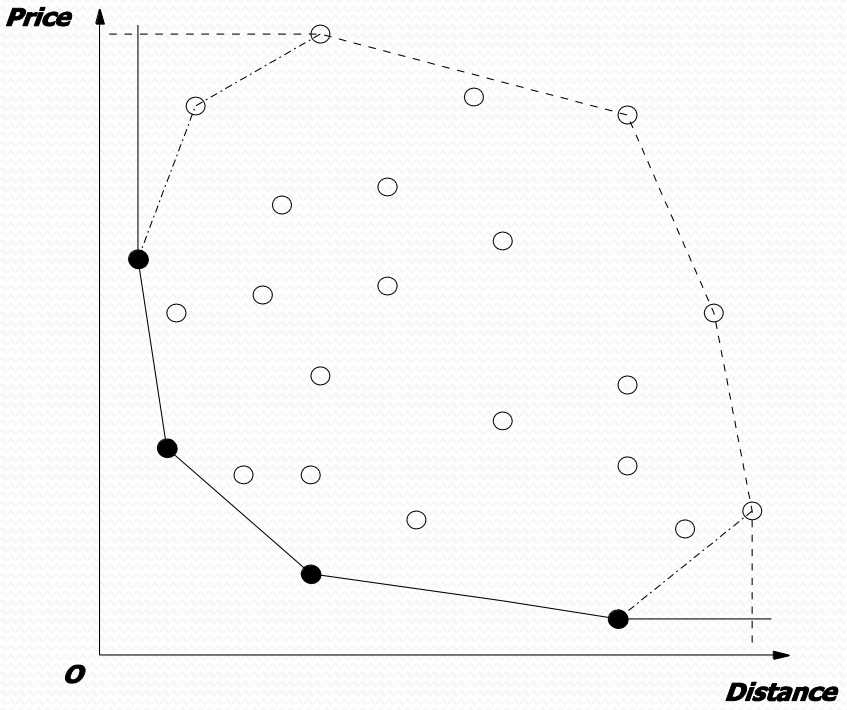
By Jian Wen

Outline

- Skyline
  - Definition and properties
  - Related problems
- Progressive Algorithms Analysis – SFS
  - Analysis
  - Algorithm
  - Cost Estimate
- Epsilon Skyline(overview and idea)

Skyline Example

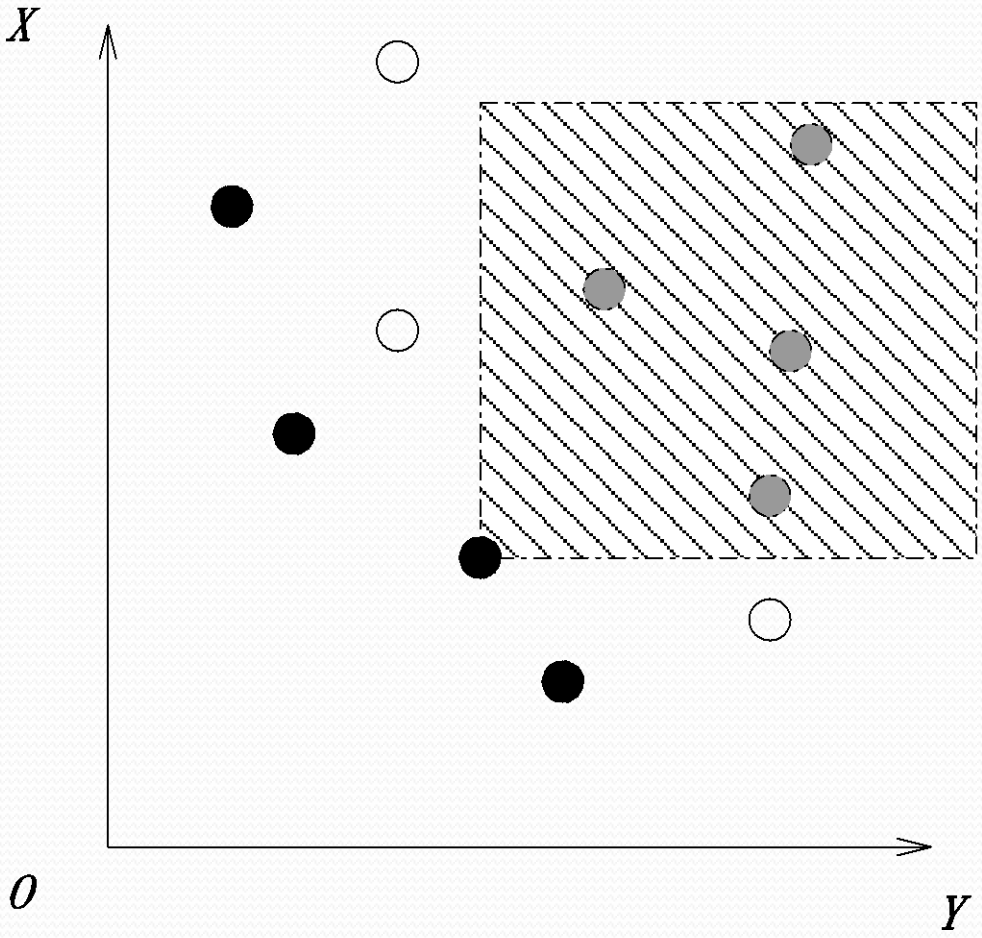
- Hotel Search:
  - Price (lower, better)
  - Distance(to downtown, etc.) (closer, better)
- Skyline Objects
  - Given a set of d dimensional points, a skyline point should have the property on each dimension no worse than any other points.



Dominance and Skyline

- Given a set of d dimensional points T:
  - We say that one point t_1 dominates another point t_2 if and only if:
    - t_1 is better than or equal to t_2 on all dimensions, and
    - t_1 is better than t_2 on at least one dimension.
  - Here "better" can be either "smaller better"(minimum skyline) or "larger better"(maximum skyline) or any other criteria measurable and comparable
- The skyline points will be the ones which will not be dominated by any other points in T

Dominance



- For the dominance relation, the following two properties hold:
  - Transitivity;
  - Asymmetry.

Formal Definition

- Given a set of d dimensional points T:
  - We say that one point t_1 dominates, denoted by $\prec$, another point t_2 if and only if:

$$\forall i = (1,2,\ldots, d), t_1[i] \leq t_2[i], \text{ AND}$$

$$\exists j = (1,2,\ldots, d), t_1[j] < t\ [j]$$

- Skyline points in T will be the points t satisfying:

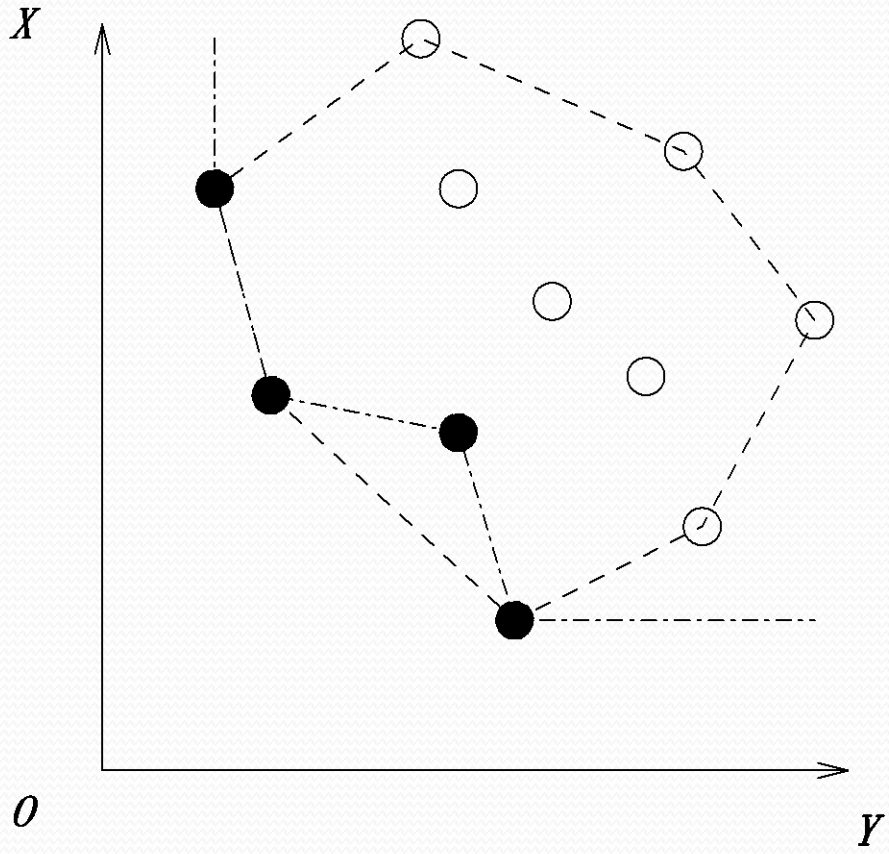
$$\neg \exists t' \in T, \text{so that } t' \prec t$$

Related Problems

- Convex Hull
  - Let S be the union of the both maximum Skyline and minimum Skyline, C be the convex hull.

$$C \subseteq S$$

  - In the image right, black points are minimum skyline points.



Related Problems

- Maximal Vectors
  - The concept of Skyline in fact comes from maximal vectors.
    - Find the vectors which are not dominated by any other vectors.
    - “Dominate” is just the same as the definition in the previous slide.
  - Skyline computation focuses on the tuples in databases, where there will be multiple attributes, instead of dimensions.

Related Problems

- Pareto Set
  - In microeconomics in general, and game theory in particular, the Pareto set, named after economist Vilfredo Pareto, is the set of all Pareto-efficient outcomes.
  - Given a set of alternative allocations of, say, goods or income for a set of individuals, a movement from one allocation to another that can make at least one individual better off without making any other individual worse off is called a **Pareto improvement**. An allocation is **Pareto efficient** or **Pareto optimal** when no further Pareto improvements can be made.

Related Problems

- Pareto Set is a restrict skyline
  - There will be a demand function where all of the points in the set will be the feasible solution to this function.
  - When Pareto Efficiency is reached, the marginal rate of each variable in the function will be equal.
    - That is, we can not make the solution totally better, or, we cannot improve any one's benefits without deducing any other one's benefits.
- In the general skyline computation, there is no such a function which can be used to measure the total benefits over all the variables.

Algorithms Analysis

- Assumption on the data set:
  - Independence
    - Values of each dimension will be independent to any other dimensions.
  - Distinct values
    - No duplicates on any dimensions
  - Uniformity
    - For each dimension, the values will be uniformly distributed.

Algorithms Analysis

- It can be seen clearly that Skyline computation can be done in polynomial time.
  - For a given d-dimensional set with n points
  - Straight-forward method:
    - For each point, compare it with all the other points to check whether it can be dominated by some points:
      - If so, remove it;
      - If not, mark this point as a skyline point.
    - Recursively run the step above until all points have been checked.

$$O(dn^2)$$

Algorithms Analysis

- Is it worse for a polynomial time algorithm?
  - Actual running experiment:
    - NBA player data, from 1946-2004, 16 comparable attributes, 16380 records.
    - Straight-forward way, all in main memory.
    - CPU 1.73G, main memory 1G
    - 2 mins.
- It is not very bad at the first view. But consider the situation:
  - Users want to get the results ASAP, and maybe at first just some options and later more choices(progressive).
  - Users may want to specify the number of Skyline records returned, which may be more or less then the actual number(Top-k Skyline).

Two attempts

- Prune the points dominated will deduce the number of comparison sharply.
  - SFS(Sort Filter Skyline):
    - Skyline With Presorting, Jan Chomicki, Parke Godfrey, Jarek Gryz, Dongming Liang, 2002
    - Progressive Skyline computation.
- Reduce the dimension
  - Epsilon Skyline: Top-k Skyline computation.

Sort Filter Skyline

- Idea: Presort the data
  - Make sure that no point can be dominated by the ones comes after it.
- Presort the data with the sum of all the attributes/dimensions of each record/point.

Sort Filter Skyline

- Analysis:
- After sorting on the data set, the one with the maximum sum will be certainly the skyline point.
  - Can be proved by the definition of dominance:
    - If this record is not a Skyline record, it must be dominated by some other point, so according to the definition of dominance, that point must have a greater sum than this point.
    - Contradiction! Proof done.

Sort Filter Skyline

- For all the other points:
  - Theorem: after sorting, for each record t in the sorted data set, t will not be dominated by any of the records after it in a descent order.
  - Proof can be made just like the previous one.
- So if we want to find all the Skyline records…
  - For each record, if it will not be dominated by any of the Skyline records with the sum greater than it, this record will not be dominated by any record in the data set, so then this record will be a Skyline record.

Sort Filter Skyline

- According to the analysis above, we have:
  - With the processing of the SFS algorithm, Skyline records can be returned in a progressive way.
    - When one record is added into S, it must be a Skyline record;
    - The first part of Skyline records will be returned fast:
      - First Skyline record is returned at once;
      - Other records with greater sum will be returned with less comparison.
  - Totally the algorithm is much more faster than the straight-forward way:
    - Comparison will be made just on the Skyline records;
    - All the records will only need to visit once.

Sort Filter Skyline

- Algorithm: data set *T*, *d* dimensions
- Sort all records by the sum of all attributes in descent way, get *T'*;
- Maintain a set *S* for skyline records. Move the first record into *S* from *T'*;
- While *T'* is not empty, for each record *t* in *T'*:
  - Compare *t* with all the records in *S*:
    - If *t* is dominated by some record in *S*, then remove t from *T'* and continue the while loop with the next record;
    - Else move t from *T'* to *S*, do the while loop with the next record.
- Return *S*

SFS – Time Cost Analysis

- To evaluate the SFS algorithm, we need to calculate all the comparisons, plus the cost on sort(O(nlgn)).
- The total comparison can be divided into two parts:
  - The comparison between Skyline records(m is the number of Skyline records):

$$(m-1)+(m-2)+\ldots+1=\frac{(m-1+1)*m}{2}=\frac{m^2}{2}$$

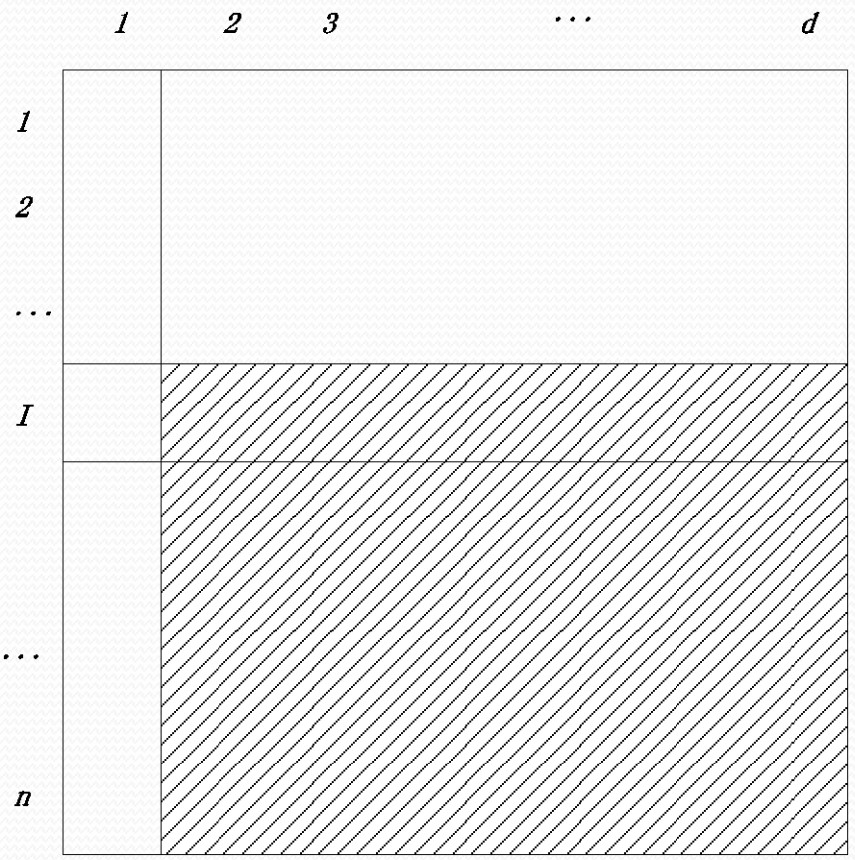
  - The comparison for eliminating the non-skyline records:
    - How to calculate?

SFS – Time Cost Analysis

- Consider the situations that:
  - Best case: all the non-skyline records will be eliminated at once when we make the comparison between it and a skyline record.
  - The total comparison will be n-m.
  - Worst case: to eliminate each non-skyline record, all of the skyline records should be used for comparison.
  - The total comparison will be at most m(n-m)

SFS – Time Cost Analysis

- Then, how to evaluate m(the number of Skyline records)?
  - Consider the situation that all the records will be sorted on the first column.
  - Then the probability of "i-th record is a skyline record" is equal to the probability of "i-th record is a skyline record in the space of (I->n, d-1)".



SFS – Time Cost Analysis

- Proof: The probability of "i-th record is a skyline record" can be divided into:
  - Probability of "i-th record is not dominated by the first i-1 records", which is definitely correct since in the first column, i-th record has greater value than the first i-1 records.
  - Probability of "i-th record is not dominated by the later n-i+1 records"
    - Since i-th record has the smallest value on the first column, this probability will be equal to the one on the (d-1)-dimensional space starting from i to n.

SFS – Time Cost Analysis

- Then let A(n, d) be the expectation of the number of Skyline records for n records on d-dimensional space

$$A(n,d) = \sum_{i=1}^{n} P\{t_i \text{ is a skyline record}\}$$

$$= \sum_{i=1}^{n} \frac{A(n-i+1, d-1)}{n-i+1}$$

- In the situation that data set is uniformly independent, we can assume that A(n, d) is monotone to n:

$$A(n,d) = \sum_{j=1}^{n} \frac{A(j,d-1)}{j} \le \sum_{j=1}^{n} \frac{A(n,d-1)}{j} \le A(n,d-1)H(n)$$

$$= H^{d-1}(n) \approx O(\ln^{d-1} n)$$

SFS – Time Cost Analysis

- Now from the estimation on the Skyline records, we can have the final cost analysis on SFS:
  - Best case:

$$n\lg n+\frac{[O(\ln^{d-1} n)]^2}{2}+n-O(\ln^{d-1} n)=O(n\lg n)$$

  - Worst case:

$$n\lg n+\frac{[O(\ln^{d-1} n)]^2}{2}+O(\ln^{d-1} n)[n-O(\ln^{d-1} n)]=O(n\lg n)$$

SFS – Time Cost Analysis

- Since we are using the assumption that the number of the Skyline records are expectation in the UI data set, the estimation we got above is just the average cost. In fact the worst may be much more worse, and the best case much more better:
  - Best: Only one skyline record, which will be returned at once, and then all the other records will be dominated, so the cost is O(n).
  - Worst: All the records are skyline records, so all the records will be checked and compared. The cost will be $O(n^2)$.

Epsilon Skyline

- Idea: reduce the dimension
- Consider the definition of dominance:
  - If we add or subtract a ε value on the left side when we make the comparison between two points, for the maximum skyline:
    - If ε -> +∞, any of the points will not be dominated by any other ones, since ε makes it larger enough. So the ε-skyline will be the whole data set.
    - If ε = 0, the ε -skyline will be just as the original skyline.
    - If ε -> - ∞, any of the points will be dominated by any other ones, since ε makes it smaller enough. So the ε-skyline will be an empty set.

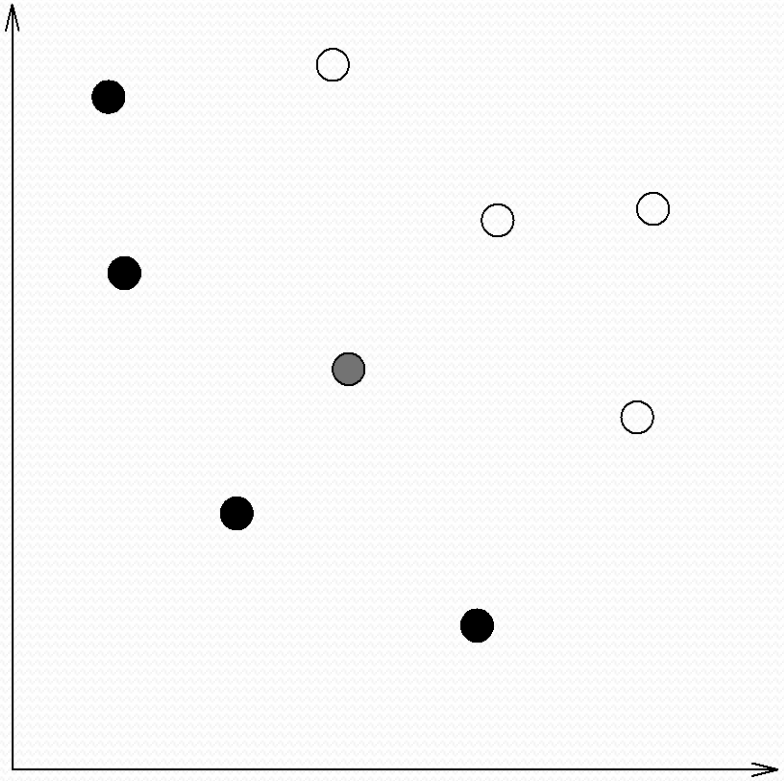
Epsilon Skyline

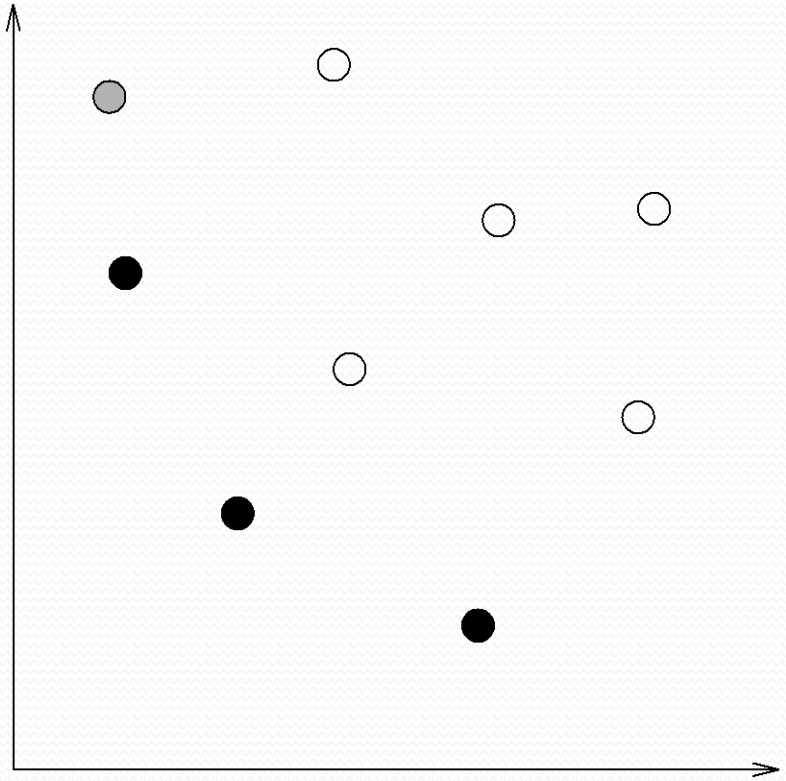
- Our goal:
  - Assign a distinct ε value to each points, so that when we need to get the skyline points, we only need to return all the points with ε <=0.
  - ε value can be seen as a rank on each points, so with the rank, we can return a given k points but not only the exact skyline points.
- Problem:
  - How to define ε?
  - How to calculate ε efficiently?

Epsilon Skyline

- (Intuitively)Definition on ε:
  - To increase the number of Skyline records: for all the non-skyline points, recursively find the one with the smallest ε so that after adding ε this point will not be dominated by any of the original skyline points.
  - To decrease the number of Skyline records: for all the skyline points, recursively find the one with the smallest ε so that after subtracting ε this point will be dominated by some point in the whole data set.

Epsilon Skyline





Epsilon Skyline

- This topic is still a on-going research:
  - Define the ε in a formal way so that we can use it in the analysis and the design of the algorithms;
  - More efficient algorithms: since now based on the intuitive idea, we can see that if we want to get the ε value for each points, we should do comparison with all the other points.

QUESTIONS?