Simple Object Access Protocol Services

▶ Web Services

Simulated Annealing

▶ Routing Vehicles, Algorithms

Simultaneous Autoregression

▶ Spatial and Geographically Weighted Regression

Simultaneous Autoregressive Model (SAR)

▶ Spatial Regression Models

Simultaneous Spatial Operations

▶ Concurrency Control for Spatial Access
▶ Concurrency Control for Spatial Access Method

Sincerity

▶ Participatory Planning and GIS

Single Image

▶ Photogrammetric Methods

Skyline Queries

NILU THAKUR
Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA

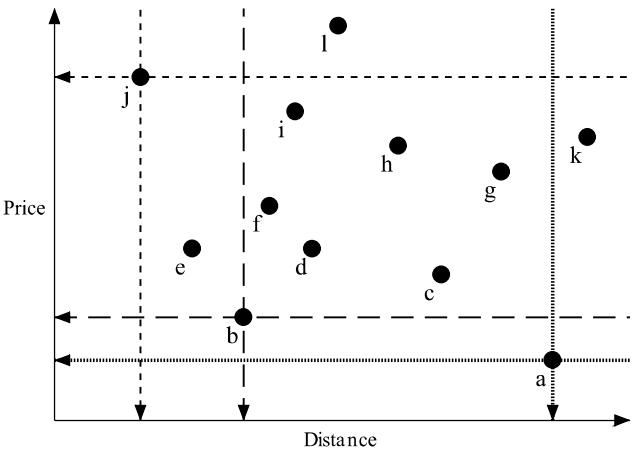
Synonyms

Top-N queries; Search, multi-criteria; Search, multi-dimensional; Ranking Results; Ranking, multi-dimensional; Information Retrieval; Nested-loop, blocked; Divide and conquer; Nearest Neighbor algorithm; Branch and Bound

Definition

Given a set of points $p_1, p_2, \ldots, p_n$, the skyline query returns a set of points P, called skyline points, such that any point $p_i \in P$ is *not dominated* by any other point in the dataset. A point p_1 dominates another point p_2 if p_1 is not worse than p_2 in all dimensions, and p_1 is better than p_2 in at least one dimension. In other words, a point p_1 dominates another point p_2 if and only if the coordinate of p_1 on any axis is smaller than the corresponding coordinate of p_2.

For example, Fig. 1 shows a dataset containing information about hotels. The distance of the hotel from the beach and the price for each hotel are assigned to the X, Y axis of the plot respectively. There are 12 hotels in total represented as data-points in the figure. The interesting data-points in the dataset are {a, b, j}, because 'a' has the lowest price and 'j' has the least distance from the beach. Although 'b' has neither the shortest distance nor the minimum price, it has a lesser distance value than 'a' and a lower price value than 'j'; hence, 'b' is not dominated by either 'a' or 'j'. All the other points in the dataset are dominated by the set of points {a, b, j}, i. e., for all other points, both the distance and price value are greater than one or more set of points {a, b, j}, also called skyline points.



Skyline Queries, Figure 1 Datasets of hotels with their price and least distances from any beach

Historical Background

Skyline and operator queries were first introduced by Bäorzsaonyi et al. [1]. Skyline, in the literal sense, defines an outline, perhaps of buildings or a mountain range against the background of the sky. As the skyline picture of a place shows only the taller or the prominent buildings and structures, similarly, the skyline query on a set of data points returns only the most interesting data points. Skyline queries and skyline operators were introduced to the

database context by applying the problem of finding the maxima of a set of points.

In database management systems, rank-aware queries, like top-N or skyline, are often used to mine for the most interesting data items among a large amount of data. Recently, due to the application of skyline queries in multi-criteria decision-making and mobile service applications, skyline queries have gained popularity and are now widely studied in database literature.

Scientific Fundamentals

Algorithms

There are many algorithms for evaluating skyline queries and many more new algorithms have been recently added. A few of the important ones are explained here.

Block Nested Loop (BNL) Algorithm: BNL, proposed in [1], is based on a very straightforward approach of computing the skyline in which each point p is compared with every other point in the dataset; if p is not dominated, then it becomes a part of the skyline or otherwise rejected. BNL builds on this concept by scanning the data file and keeping a list of candidate skyline points in the main memory. The following list summarizes the steps.

- The first data point is inserted into the list of skyline points.
- All the data points in the data file are scanned and tested for dominance criteria. For each subsequent point p, there can be three cases:
  a) If p is dominated by any point in the list, it is discarded as it is not part of the skyline.
  b) If p dominates any point in the list, it is inserted into the list and all points in the list dominated by p are dropped.
  c) If p is neither dominated nor dominates any point in the list, it is inserted into the list as it may be part of the Skyline.
- The active list of potential skyline points seen thus far is maintained and each visited point is compared with all elements in the list.
- The list self-organizes because any point found to be dominating other points is moved to the top of the list. This reduces the number of possible comparisons since points that dominate multiple other points are likely to be checked first in order to determine whether a new point is part of the skyline or not.
- This is continued until all the points in the data set get scanned.
- BNL does not require a precomputed index, and the execution remains independent of the dimensionality of the space.

A problem of BNL is that the list may become larger than the main memory. In such cases, all points falling in case 'c' are added to a temporary file. This fact requires multiple passes of BNL. In particular, after the algorithm finishes scanning the data file, only points that were inserted in the list before the creation of the temporary file are guaranteed to be in the skyline and are thus output. The remaining points must be compared against the ones in the temporary file. Thus, BNL has to be executed again, this time using the temporary (instead of the data) file as input. The advantage of BNL is its wide applicability since it can be used for any dimensionality without indexing or sorting the data file.

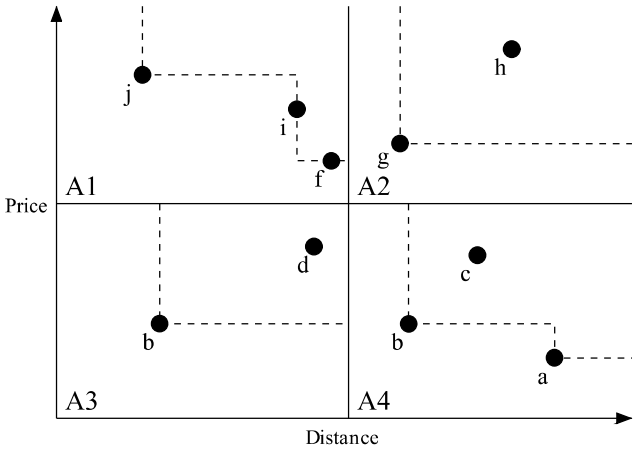
Shortcomings of the BNL algorithm:

- The reliance on main memory, since small memory may lead to numerous iterations
- Its inadequacy for on-line processing, since it has to read the entire data file before it returns the first skyline point
- It does perform redundant work and there is no provision for early termination.

Divide-and-conquer algorithm: This algorithm is based on a very simple approach of incremental evaluation of partial skyline points. This approach divides the large data set in to several smaller partitions which can fit in main memory and then computes a partial skyline of points in every partition using a main memory algorithm [2,3]. The steps are summarized here:

- Recursively large datasets are broken into smaller partitions. This process continues till each smaller partition of the dataset fits in the main memory
- The partial skyline is computed for each partition using any in-memory approach and later these partial skyline points are combined to form the final skyline query.

Figure 2 illustrates how the divide and conquer algorithm works. The data space has a dataset with 10 data points. The data space is further divided into 4 partitions, A_1, A_2, A_3 and A_4, and a partial skyline is computed for these partitions. The data space has its partial skyline as {j, i, f}, {g}, {e} and {b, a}, respectively. To obtain the final skyline, all partial skylines from different data spaces need to be compared with other partial skylines for the dominance criteria. As can be seen in the Fig. 2, the partial skyline of data space A_3 must appear in the final skyline, while those in data space A_2 should be discarded immediately as they are dominated by point in data space A_3. Then, each skyline point in A_1 must be compared only with skyline points in A_3, because no point in A_2 and A_4 can dominate points in A_1. In this figure, point 'i' and 'f' from data space A_1 will be removed because they are dominated by data point 'e'. Also, the skyline of data space A_4 should be compared with points in A_3, resulting in the removal of data point 'b'.



Skyline Queries, Figure 2 Divide and conquer approach

Finally, the skyline set will contain the data points {j, e, a} as the final skyline points.

Shortcomings of the Divide and conquer approach are:

- Efficient only for small data sets, because for large data sets partitioning requires reading and writing the entire data set once, and incurs significant IO cost
- Not suitable for online processing, as the output cannot be sent until the partitioning phase is completed.

Nearest Neighbor (NN) Algorithm: The NN algorithm is based on the results of a nearest neighbor search to partition the data universe recursively. It assumes that a spatial index structure on the data points is available for use. In general, NN identifies skyline points by repeated application of a nearest neighbor search technique on the data points using a suitably defined L1 distance norm. The various steps can be summarized as:

- First a nearest neighbor query is performed using an existing algorithm on the R-tree such as [4,5] and the point with minimum distance from the beginning of the index is found. For example, in Fig. 3a, the nearest neighbor in the data-point is 'a', as it is closest to the origin when an L1 distance measure is assumed
- **'a'** divides the entire space into 2d non-disjoint region, which now must be recursively searched for more skyline points
- Regions 2 and 4 need not be searched, because all the points in region 2 are dominated by 'a'. The rest of the regions need to be searched
- Next, the search is recursively done on region 1. In that region, the nearest neighbor would be 'f', then the region is exploded to form additional region. Now with f as the nearest neighbor, region 4 need not be searched and region 4 is added to the pruned region
- In this way, the set of partitions resulting after finding a skyline point are inserted in a to-do list until the to-do list is no longer empty, NN removes one of the partitions from the list and recursively repeats the process
- In the case of high-dimensional data sets, the number of unexplored region grows rapidly and therefore, the non disjoint condition can be relaxed
- The restriction that regions are non-overlapping is relaxed. An assumption can be made that a point query splits each dimension into two regions; instead of exploding a region to 2 to the power of d, it can be reduced to $2d$
- This leads to a fewer number of regions to search at the expense of performing the task of duplicate removal
- Laissez-Faire and propagate are the two duplicate removal techniques that can be used.

Duplicate Removal in the NN algorithm:

Kossmann et al. [6] proposed the following elimination methods for duplicate removal:

Laissez-Faire: This method maintains an in memory hash table which keys in each point and flags it as a duplicate if it is already present in the hash table.

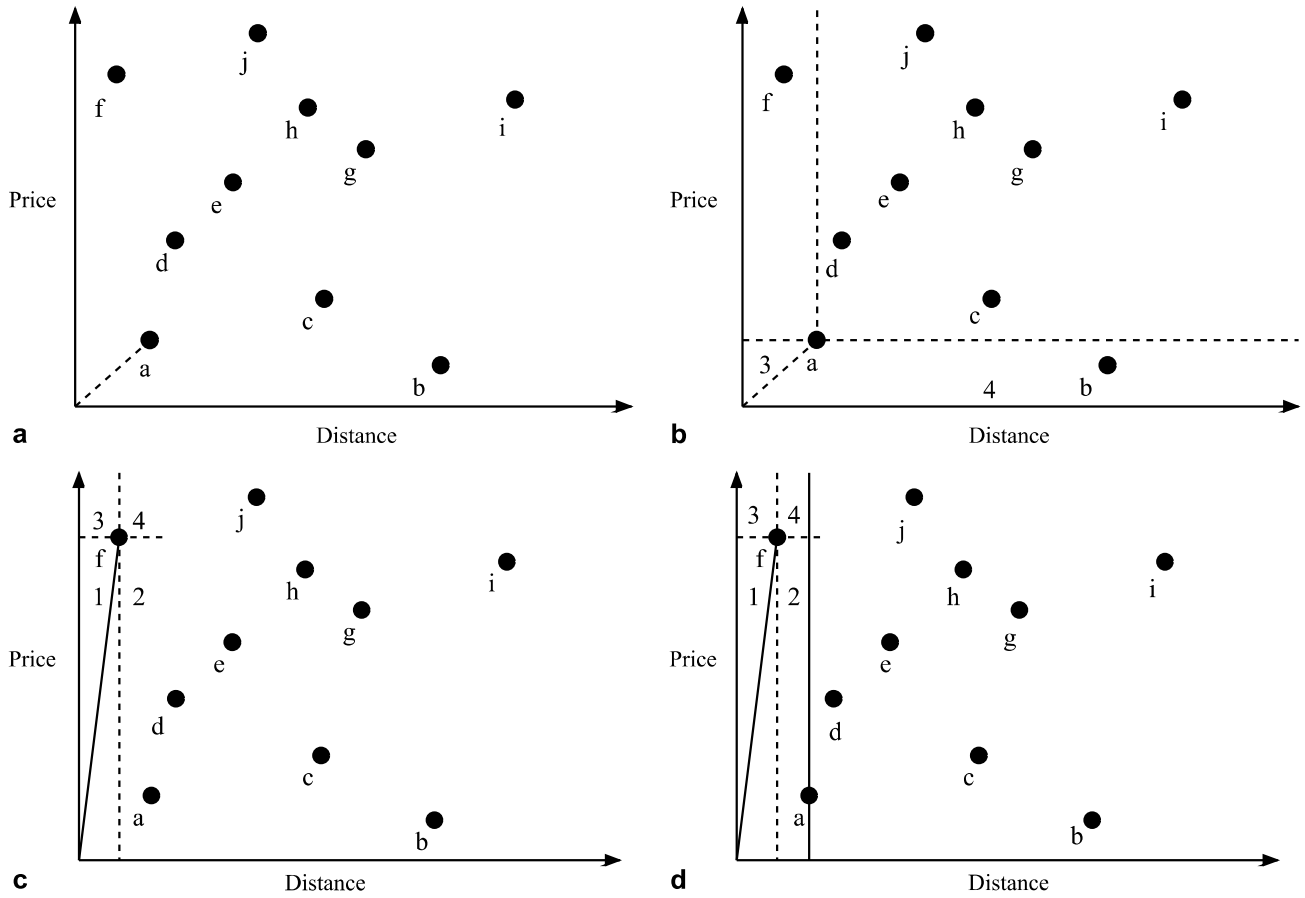
Propagate: When a point p is found as the skyline point, then all the instances of p are removed from all unvisited nodes.

Shortcoming of the Nearest Neighbor algorithm:

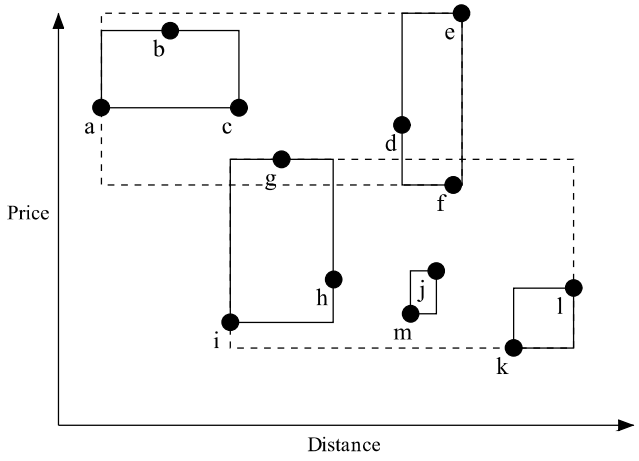
- In the case of high-dimensional data sets, the unexplored region or to-do list grows rapidly and in such cases, the non-disjoint condition needs to be relaxed for high-dimensional datasets.

Branch and Bound Skyline (BBS) Algorithm: The BBS algorithm proposed in [7], like the NN algorithm, is based on nearest neighbor search. Specifically, BBS uses a branch and bound paradigm. For the data set shown in Fig. 4, the steps of the BBS algorithm can be summarized as follows:

- Assuming an R-tree on the data set, the algorithm starts from the root node of the R tree and inserts all its entries in a heap sorted according to their mindist ordering relative to the origin using L1 distance norm
- Then, the entry with the minimum mindist is expanded, which is e7 in this case
- The expansion process removes the entry e7 from the heap and inserts its children e3, e4 and e5
- The next entry to be expanded is e3 with minimum mindist and the data point 'i' is found as the first nearest neighbor. This 'i' becomes part of the skyline
- Then, e6 is expanded and its children are checked for dominance criteria. Only the ones that are not dominated by some points in the skyline are inserted into the heap. For example, in the figure, 'e2' and 'h' are pruned because they are dominated by point 'i'
- The algorithm proceeds in this manner until the heap becomes empty.



Skyline Queries, Figure 3 **a** The nearest neighbor is 'a'. **b** 'a' divides the space in to 2d non-disjoint regions. **c** A search is done on region 1 and point 'f' found. **d** Region 4 is added to the pruned region and need not be searched



Skyline Queries, Figure 4 Dataset with an R-tree to illustrate the BBS algorithm

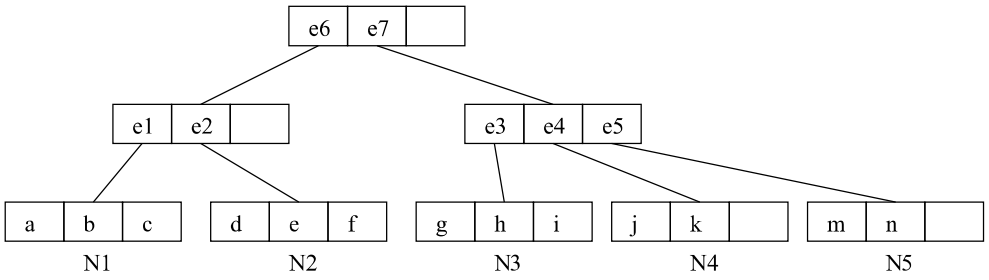
Variations of Skyline Queries

Ranked Skyline Queries: Given a set of points in d-dimensional space, a ranked or top-k skyline query takes a parameter 'k' and a preference function 'f', which is monotone on each attribute, and returns the skyline points that have a minimum score according to the input function. In the case of ranked skyline queries, in addition to the distinct score functions used for skyline computation a "total" score function is also provided. Then, all skyline points are returned in order of the "total" score function. An alternate preference function is used instead of the minimum criterion. The priority queue uses the alternate preference-function to compute the Minimum Distance to the elements in the queue.

For example, in Fig. 4, if x and y coordinate of 'i' and 'k' are (3,2) and (9,1), respectively, and if $K = 2$ and the preference function is $f(x, y) = x + 3y^2$, then the output skyline point would be $\{k,12\}$ and $\{i,15\}$. The Branch and bound algorithm can easily handle ranked skyline queries by modifying the mindist definition to reflect the preference function. By contrast, other algorithms like Block nested loop and divide and conquer first try to retrieve the entire skyline, and then sort the skyline according to their score, then outputing the best k points.

Constrained Skyline Queries: A constrained skyline query returns the most interesting points in the data space defined by the constraints. Generally, each constraint is expressed as a range along a dimension and the con-

S



Skyline Queries, Figure 5 R-tree representation of the Fig. 4 data set

junction of all constraints forms a hyper-rectangle in the d-dimensional attribute space. An example of a constraint skyline query would be a case where a user is interested only in hotels whose nightly rate is in the range of \$40–\$70.

The Branch and Bound skyline algorithm (BBS) can easily process constraint skyline queries by pruning the entries not intersecting the constraint region. In addition to the usual skyline input parameters, the data space is constrained and the skyline queries return skyline points only from the constrained data space. This is typically done by specifying a hyper-rectangle in which all data items have to be located. When inserting objects into the priority queue, objects that lie completely outside the constraint region are pruned.

Dynamic Skyline Queries: A dynamic skyline query specifies an 'm' dimension function $f_1,f_2,\ldots,f_m$, such that each function f_i $(1<i<m)$ takes as parameters the coordinates of the data points along a subset of d axes. These queries return the skyline in the new data space with dimensions defined by $f_1,f_2,\ldots,f_m$. For example, suppose a data base stores for each hotel its X and Y coordinates and its price i.e, the database contains three dimensions and suppose a user specifies his or her current location (l_x, l_y) and wants to know the most interesting hotels with respect to Euclidean distance and the price of the hotel. In this case, each point P with coordinate (p_x, p_y, p_z) in the original 3D space is transformed to a point p' in the 2D space with coordinates $(f_1(p_x, p_y), f_2(p_z))$ where the dimension functions f_1, f_2 are defined as:

$$f_1(p_x, p_y) = (p_x - u_x)2 + (p_y - u_y)2 \text{ and } f_2(p_z) = p_z\,.$$

Enumerating Skyline Queries: For each skyline point 'p' in the data set, the number of points dominated by 'p' are found by enumerating the queries. This information may be relevant in applications where the goodness of a skyline point is determined by how many other point it dominates. This is done by defining the spatial bound for the region where a skyline point dominates. Then, all points in the data sets are scanned and checked against the spatial extent for each of the skyline points. The total number of point-region intersection gives the required count for all skyline points.

K-dominant Skyline queries: A k-dominant skyline contains all the points that cannot be k-dominated by any other points. A point 'p' is said to k-dominate another point 'q' if there are k dimensions in which 'p' is better than or equal to 'q' and better than 'q' in at least one dimension. For example, in Table 1, $\{p_1,p_2,p_3,p_5\}$ form a free (6-dominant) skyline, $\{p_1,p_2,p_3\}$ form a 5-dominant skyline, while $\{p_1,p_2\}$ form a 4-dominant skyline. p_4 is not a part of the 5-dominant skyline as it is dominated by other points in all dimensions. p_5 is not a part of the 5-dominant skyline as it dominated by other points in all 5-dimensions not involving d4.

Skyline Queries, Table 1 An example data set showing k-dominance

	d1	d2	d3	d4	d5	d6
P1	2	2	2	4	4	4
P2	4	4	4	2	2	2
P3	3	3	3	5	3	3
P4	4	4	4	3	3	3
P5	5	5	5	1	5	5

Spatial Skyline queries: Given a set of data points P and a set of query points Q, each data point has a number of derived spatial attributes, each of which is the point's distance to a query point. An SSQ retrieves those points of P which are not dominated by any other point in P, considering their derived spatial attributes. The main difference with the regular skyline query is that this spatial domination depends on the location of the query points Q. SSQ has application in several domains such as emergency response and online maps [3].

Key Applications

Skyline queries and their variations are used in many applications involving multi-criteria decision-making, mobile applications, streaming applications, spatial applications and others.

Multi-criteria Decision Making

This area can again be divided into applications with static data points and those with dynamic data points. Examples

with static data points are cell phone finder and movie rating, whereas location based services with moving objects are examples of decision-making with dynamic data.

Cell Phone Finder

A person looking to buy a cell phone at a website may care about a large number of features like weight, size, talk time, standby time, screen size, screen resolution, camera quality, color, brand and other things specific to his/her choice. There will be too many phones for a person to examine manually. In such cases, computing the skyline over the desired cell phone features may remove a large number of cell phones whose features are not better than those in the skyline, thus leaving a manageable number of phones for evaluation.

Movie Rating

A person looking for top ranked movies based on the ranking given by other users on a particular movie website might be interested in only top ranking movies. In such cases, the rating of each user corresponds to a dimension in the data set, and due to a large number of users, the data set obviously becomes a high-dimensional one. The skyline of the data set will contain only top rated movies while the low ranked queries will be ignored and not become part of the skyline. Note that in both the above cases, ranking can be done by providing some preference functions and requesting users to provide some weight assignments for their preferred features or more trusted users in the latter case. However, providing such weight assignments for a large number of dimensions is not always easy without any initial knowledge about the data. For example, it is not clear how weight assignments can be provided to aggregate the talk time and camera quality of a phone into one grade. Computing skylines in high-dimensional data sets is challenging because of the large number of skyline points. On the movie ranking website, for example, it is nearly impossible to find a movie which is ranked lower than another movie by all the users. Such a blowup in the answer set not only renders the skyline operator worthless (with respect to the desired pruning of candidates), but it also results in high computational complexity for both index and non-index methods as many pairwise comparisons are performed without affecting any pruning.

Location Aware Mobile Services

Suppose that a multidisciplinary task force team located at different and fixed offices wants to put together a list of restaurants for their weekly lunch meetings. These meeting locations must be interesting in terms of traveling distances for all the team members; for each restaurant r in the list, no other restaurant is closer to all members than r. Also, considering that the team members are mobile and change location over time, the computation of skyline becomes complex. In this case, the restaurants distance attributes based on which the domination is defined are dynamically calculated based on the user's query or based on the location of the team members.

Road Network Services

A person looking for a gas station may prefer to minimize detours rather than distance, while a user searching for an emergency room is likely to be interested in minimizing the distance and is insensitive to the detour. In such cases, the skyline operator should return larger result sets from which the user can choose the route. The skyline mechanism returns a result if no other result exists that is better with respect to all the criteria considered. This is useful when a total ordering cannot be defined on the space of all criteria.

Future Directions

Progress is on-going with regards to work in skyline computations and important avenues of research in the future include:

- Alternative optimal and progressive algorithms for high-dimensional spaces where R trees are inefficient
- Fast retrieval of approximate skyline points, i.e, points that do not belong to a skyline but are very close
- Challenges and application of spatial skyline queries in metric spaces such as road networks.

Cross References

- ► Aggregation Query, Spatial
- ► R-Trees – A Dynamic Index Structure for Spatial Searching
- ► Vector Data

Recommended Reading

1. Baorzsaonyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. ICDE (2001) pp. 421–430, Heidelberg, Germany
2. Stojmenovic, I., Miyakawa, M.: An optimal parallel algorithm for solving the Maximal Elements Problem in the Plane. Parallel Computing (1988) **7**(2):249–251
3. Matousek, J.: Computing dominance in E^n. Information processing letters, Vol. 38 pp. 277–278 (1991)
4. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest Neighbour Queries. SIGMOD (1995)
5. Hjaltason, G., Samet, H.: Distance Browsing in Spatial Databases. ACM TODS **24**(2):265–318 (1999)

6. Kossmann, D., Ramsak, F., Rost, S.: Shooting Stars in the Sky: An Online algorithm for Skyline Queries. In: Proceedings of VLDB'02, pp. 275–286 Hong Kong, China (2002)
7. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for Skyline Queries. In: ACM SIGMOD, San Diego, CA, USA (2003)
8. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: proceedings of ICDE'03, IEEE Computer Society, pp. 717–816 Bangalore, India (2003)
9. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the Sky: Efficient Skyline Computation over Sliding Windows. In: Proceedings of ICDE'05, IEEE Computer Society, pp. 502–513 Tokyo, Japan (2005)
10. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Trans. Database Syst. **30**(1),41–82 (2005)
11. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient Progressive Skyline Computation. In: Proceedings of VLDB'01, pp. 301–310 (2001)

Smallworld GIS

► Smallwold Software Suite

Smallworld Software Suite

ROBERT P. LAUDATI
Marketing Program Manager, GE Energy,
Englewood, CO, USA

Synonyms

Smallworld GIS; GE smallworld; Software; Object-oriented; Components; Reuse; Integration; Database management; Manifolds; Manifold rules; Version managed datastore; Version management; Versioned B-tree; B-tree, versioned; Locking; Check-out; Conflict resolution; Distributed databases; Multiple worlds; Worlds, multiple; Magik, smallworld; Service oriented architecture; OGC standards, ODBC; COM/OLE; Oracle spatial

Definition

Smallworld Core Spatial technology is an object-oriented, database-driven product that provides a powerful, consistent architecture for geospatial applications such as those used for planning electric, gas and water distribution systems, designing telecommunications networks and evaluating strategic market opportunities. Encapsulated, reusable components enable rapid development with reduced custom code, driving down maintenance and upgrade costs-build once, use many times. Smallworld technology has been proven to deliver solutions for thousands of users managing terabytes of data across complex, distributed operations. The software integrates with other products that require spatial information, including systems for customer relationship management, market analysis, network and work management.

Historical Background

Smallworld Systems Limited was founded in 1988 by ten pioneers with extensive experience in computer graphics and large databases obtained during careers at the CAD-Centre in Cambridge, England, and Cambridge Interactive Systems (CIS), the producer of Medusa, which became the leading CAD product in Europe.

During its first year, Smallworld was commissioned by the world's leading technology vendors to research and assess the needs of the GIS marketplace in Europe, with a focus on utility companies. More than 650 organizations in ten countries were interviewed, including transportation, cartographic, and government agencies. This wealth of information provided Smallworld with revealing insights into the real needs of the marketplace and an in-depth understanding of the GIS user community.

In April 1990, the Smallworld GIS product was first launched at the European GIS conference (EGIS) in Amsterdam, and by August 1990, the company had signed its first customer. In 1991, sales and support offices were opened in the Netherlands and Sweden, and the company had installed systems in utility companies in the United Kingdom, Germany, Switzerland, the Netherlands, and Sweden. By the end of 1992, Smallworld Systems had over 60 customers in Europe.

Smallworld technology was introduced to the North American market in 1993 at the Automated Mapping/Facilities Management (AM/FM) show in Orlando, Florida. Soon after, Smallworld established a U.S. subsidiary in Colorado. By July of 1993, Smallworld had its first success in North America, winning a contract with Centra Gas in Manitoba, Canada. This was quickly followed by contracts with Public Service Company of Colorado, Entergy Services Corporation, Commonwealth Edison, and Wisconsin Gas.

In October of 2000, Smallworld Systems became a business unit of GE Energy Management Services, a part of GE Energy. GE Energy (http://www.geenergy.com) is one of the world's leading suppliers of power technology, energy services and information management systems with 2005 revenues of $16.5 billion, serving customers in over 100 countries for over a century.

Scientific Fundamentals

Smallworld Core Spatial Technology ("Core") is based on database management foundations rather than file or CAD