Asian Journal of Information Technology 13 (3): 106-111, 2014
ISSN: 1682-3915

Evaluation of Multiple Choice Queries on the Basis of Stable-Marriage Problem

K. Padmapriya and S. Sridhar
Sathyabama University, Chennai, India

Abstract: Consider multiple users searching for an apartment, based on size, cost, distance from the airport, etc. Users may have different choices on the attributes of the searched objects. Even though every option queries can be evaluated by selecting the object in the database with highest score, in the case of multiple requests at the same time, multiple objects can be assigned to multiple users. We have to identify a suitable one to one matching between the queries and a subset of the objects. The proposed algorithm finds the query-object pair satisfying maximum options and removes it from the problem by performing several iterations. It can be achieved by maintaining and matching the skyline of the remaining objects with the remaining queries at each iteration. The effectiveness of the proposed solution is verified through extensive experiments.

Key words: Size, cost, searched objects, query-object pair, skyline

INTRODUCTION

Consider an apartment system where users search and choose objects or services based on their own choice. Typically different users may have different choices on the attributes of the searched objects. For a single user, the system returns the best objects with respect to their options. When different users choose the same object at the same time then the problem will be aroused. For example, a particular apartment could be the highest choice of many users while it can be assigned to only one of them. As a result, the system must look for a suitable one to one matching between the queries and the objects.

Since, nearest neighbour search is also essential to be discussed in the proposed methodology, researchers used indexing method adopted from De Berg *et al.* (2000) which consumes O(n) space. Chan (2006) achieved the lower query cost by increasing the update time. Another important algorithm used here is to support NN retrieval on data of any dimensionality and can be executed on access methods (e.g., B and R-trees (Beckmann *et al.*, 1990) available in a commercial database. Among the existing solutions, the branch and bound (Hjaltason and Samet, 1998, 1999) and best-first algorithms are the fastest for low-dimensional data and iDistance (Jagadish *et al.*, 2005) is a popular solution in high dimensional spaces. Gowda and Krishna (1979) introduce the notion of mutual nearest neighbor. Specifically, two objects p and pf in the same dataset S are mutual Nns, if p is the NN of pf in S-{pf} and pf is the NN of p in S-{p}. Finding mutual NNs is useful in data mining and several solutions (Brito *et al.*, 1997; Ding and He, 2004; Gowda and Krishna, 1978, 1979; Jagadish *et al.*, 2005) have been developed. Those solutions all stick to the original definition by Gowda and Krishna, i.e., both mutual NNs come from the same dataset. As clarified shortly, the proposed technique requires a closely relevant concept but with respect to two datasets.

Suitable one to one assignments can be done based on the classic stable marriage problem (Wong *et al.*, 2007). To compute a suitable assignment between a set of choices C (i.e., queries) and a set of Objects O, the pair (c, o) in CXO with the highest score is identified and established. The process is repeated by eliminating c and o from C and O, respectively, till C and O becomes empty. This has been adopted by earlier research on spatial assignment problems (Tao *et al.*, 2007; Bohm and Krebs, 2002). But researchers replaced the progressive nearest neighbor search by incremental top-k search. Assume a set of objects O is denoted by a data structure called R-tree (Beckmann *et al.*, 1990; Chan, 2006), researchers can use an incremental top-k search for each function in C to find out their best suitable objects. This choice-object pair (c, o) with the highest score is constant. But it can be suffered by the large number of top-k queries performed on the entire objects.

Researchers propose another algorithm for the objects in the skyline (Papadias *et al.*, 2005) of O need to be considered in each iteration of the assignment process.

The skyline of O contains all objects in O with no equivalent or better object in O with respect to all

Corresponding Author: K. Padmapriya, Sathyabama University, Chennai, India

attributes. In this way, researchers can maintain the skyline of O by avoiding accession end examination of objects unnecessarily and iteratively matching it with the function set C. The algorithm proposes an efficient skyline maintenance module and a speedy method for identifying the matching pairs between the skyline of O and C.

PROBLEM STATEMENT

Researchers assume a set of user choices C over a set of multidimensional objects O. Each object o in O is defined by n values o_1, ..., o_n. Every function c in C is represented over these n values and maps objects o ⌋ O to a numeric score c(o). C may have any monotone function, i.e., if for two objects o, of ⌋ O, ∣ i ⌋ [1, n] then, c (o) = c (of), ∣ c ⌋ C. Also, researchers use linear functions specify n options $c.\alpha_1$, ..., $c.\alpha_i$, one for each dimension. The options are normalized to:

$$\sum_{i=1}^{n} c.l = 1$$

It gives assurance that any function is not favoured over another. Given an object o ∣ O, its score with respect to c ⌋ C is Eq. 1:

$$c(o)=\sum_{i=1}^{n}(c.\alpha i+o_i) \quad (1)$$

The aim is to identify a constant one to one matching (Wong *et al*., 2007) between C and O with respect to the function c prefers o to of if c(o)>c(of) and simultaneously, object o prefers c to cf, if c(o)>cf(o). Like SMP, the matching can be done by reporting the (c, o) pair iteratively with the highest score in C×O and eliminating c and o from C and O, respectively. Thus, the matching pair in the order of.

Property 1: A choice-object pair (c, o) in CXO is constant if no function is cf ⌋ C, cf ≠ c, cf(o)>c(o) and no object is of ⌋ O, of ≠ o, c(of)>c(o) where C and O are the sets of the remaining choices and objects.

ALGORITHMS

Here, researchers explain the naive solution and then the proposed approach. Both are progressive as constant as choice-object pairs are output at once they are identified. Consider C is kept in main memory while O is represented by an R-tree on the disk. The idea behind is to apply this to other indexes and alternative storage configurations.

Brute Force Method: The solution for an assignment problem can be achieved by identifying and removing the firm pair by performing iteration, according to property 1. However, not like identifying nearest pairs in the SMP, it needs substantial effort to identify firm choice-object pairs. A brute force method is to provide top one queries against O, one for every choice in C. It will give |C| pairs. The pair (c, o) with the highest value c(o) must be constant due to o is the top one choice of c and cf(o) can be lesser than c(o) for any choice cf ≠ c.

In this method, enormous top-one queries has to be initiated, one for each choice in C. Researchers can consider that O is indexed by an R-tree Ro and identically these queries can be implemented to nearest neighbour queries as shown by Borzsonyi *et al*. (2001) and Bohm and Krebs (2002). Along with that after the pair (c, o) with the highest score c(o) is getting added with the query result, o must be deleted from Ro and if o was the top-1 object for any other choice after identifying each firm pair this method needs O(|C|) removals from Ro and O($|C|^2$) top-1 searches in Ro. Eliminations and top-1 searches have logarithmic costs. Researchers now explain another algorithm for this choice-object assignment problem.

Skyline related method: It has been noticed that if c has only monotone choices then the top-1 objects of all options must be in the skyline of O. Remembering that the skyline Osky of O is the maximum subset of O which has only objects and are not dominated by any other object. It can be said that for any o ⌋ O, unless o is not in the skyline then there does not exist an object of in Osky, such that any choice c ⌋ C would select of over o.

Based on this notification, researchers form an algorithm to work out and keep the skyline of Osky while firm choice-object pairs between Osky and C are identified and output. Following algorithm 1 is a pseudo code for this skyline related method. First of all, researchers have to find the skyline Osky of the complete set O and then while there are remaining choices, the choice-object pair (c, o) with highest value c(o) is identified and c and o are deleted from C and O, respectively and Osky is modified by deleting o from O only.

Algorithm 1: Skyline related firm assignment:

Input: Set C, R-tree Ro
Procedure:
Initialize Osky as null.
Repeat the iteration till |C|>0
    If Osky is equivalent to null then
        Assign the value returned by computeSkyline (Ro) to Osky
    Otherwise
        UpdateSkyline (Osky, o, Ro)

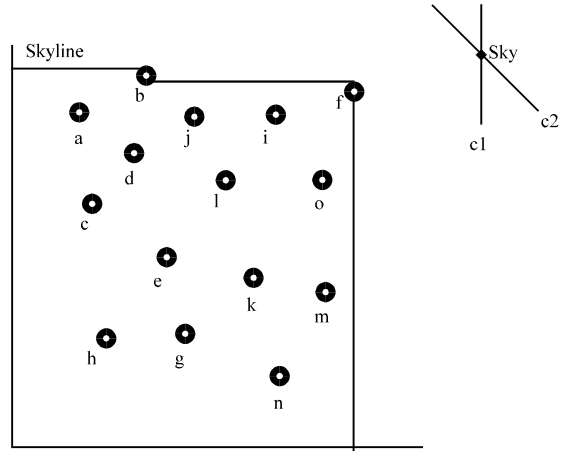

Fig. 1: Original skyline

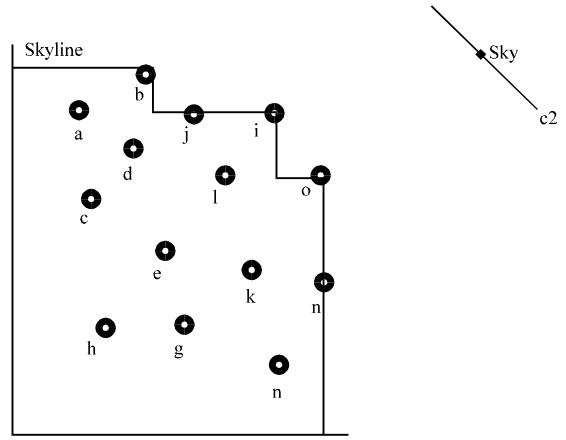

Fig. 2: Updated skyline

Find out the BestPair (C, Osky) and assign it to (c, o)
Print the result as (c, o)
Delete c and o from C and O, respectively
Remove o from Osky and assign it to Osky
Output: the closest pair (c, o)

Researchers prove the skyline related algorithm with an example. In the following Fig. 1, researchers have two choices denoted by lines and 15 objects denoted by two dimensional points. The top-1 object of each choice is the first one to be recognized if researchers remove the corresponding line from the best possible object (i.e., from top-right corner of the space) towards the worst possible object (i.e., up to the origin of the axes). In the Fig. 1, researchers can identify that 'f' is the top-1 object for both the choices.

At first the algorithm finds the skyline of O: Osky = {b, f}. From this, researchers can understand that b and f will be the top-1 objects for c1 and c2. Hence, it is essential to compare 4 choice-object pairs instead of 30 in order to identify the highest value c(o). Here, (c1, f) is the first firm pair reported by this algorithm. Osky is again modified as Osky = {b, j, i, o, m} as shown in the Fig. 2. After printing the top one firm pair, repeat the steps to update the value of C, O and Osky to identify the next highest value pair (c2, i). This pair is reported as firm pair and the algorithm ends.

The competence of the algorithm based on proper implementations of the BestPair and UpdateSkyline functions. So, researchers propose efficient methods along with it to enhance the SB algorithm to yield many firm pairs at each iteration.

IMPLEMENTATION OF SKYLINE RELATED ALGORITHM

Searching for best pair: At each iteration, the Skyline related algorithm looks for the best pair in the cross product C X Osky. An implementation of Brute Force of this process is not sufficient, since it needs |C|.|Osky| comparisons. So, the number will be decreased by indexing either C or Osky. Researchers prefer index C because it reduces only one in each iteration and at the mean time many new objects entered into Osky after each deletion. This set is not correlated, so organising the choice coefficients with multidimensional index is not sufficient. Hence, researchers prefer to index the choices as sorted list by assigning one for each coefficient. Then, researchers have to use reverse top-one search on the list for each object in Osky where the functions of objects and choices are interchanged by applying the threshold algorithm on it. Let us assume D ordered lists $L_1, L_2, ..., L_D$ where D is the dimensionality. The list L_i contains the $(c.\alpha_i, c)$ pairs of all choices $c \in C$ in decreasing order of $c.\alpha_i$ where $c.\alpha_i$ is the ith coefficient of c.

Consider a situation that researchers are looking for the best choice for an object $o \in Osky$ by calling the members of the sorted lists in a Round-Robin Method. By maintaining the choice cbest with the highest value, researchers calculate c(o) for each visited choice c. Consider the last values calculated in the lists are $\{l_1, l_2, ..., l_D\}$ which is in sorted order. Then, the threshold value T is computed as:

$$\sum_{i=1}^{D} l_{i.o_i}$$

However:

$$\sum_{i=1}^{D} li$$

should not be <1 since the assumption is that the sum of coefficients of the choices should be 1. Hence, the aim is to calculate a closer threshold value Tclose with the set of coefficients β such that:

$$\sum_{i=1}^{D} \beta i = 1 \text{ and } \beta i \,|li, |i|[l,D]$$

The threshold value is computed as:

$$Tclose = \sum_{i=i}^{D} \beta i.o_i$$

At first, researchers sort the dimensions in decreasing order according to o. Then, researchers consider each and every dimension i in this same order. With the value of B = 1, researchers set βi = min {B, li}, recompute B = B-βi and move to the next dimension. Researchers follow the same process till all βi values are computed. If at some situation B becomes 0 then researchers consider the remaining βi is 0 and terminate the process. It is easily noted that the Tclose obtained is the upper bound value for all functions.

Maintenance of incremental skyline: The next aim is to maintain the skyline after assigning an object to a function. It is not possible to recompute the skyline from scratch since it is very costly. If all |C| pairs are set up, researchers have to execute a skyline algorithm for Ro |C| times.

As described by Fagin *et al.* (2003), researchers can maintain the incremental skyline by executing the exact skyline R-tree traversal algorithm by eliminating the entries whose MBRS are dominated by current skyline objects. However, it minimizes the cost of maintenance to access only the fraction of the tree which is not dominated by the present skyline. But it needs tree traversal each time after updating the skyline. If researchers do so non-leaf entries and some objects will be accessed many times. To overcome this problem, researchers have to keep track of the pruned objects and entries during the first execution of the skyline computation algorithm. For every R-tree entry E, if E has been dominated by an object o, prune it and add it to the pruned list o.plist of o. Hence, each skyline object contains a list of dominating entries. Each pruned entry E is stored in the plist of only one skyline object o, even though E would be dominated by more than one skyline object. This will reduce the memory required.

After deleting the skyline object o, researchers then scan o.plist for each entry E whether it is dominated by any other skyline object of. If so, move E to of.plist. Else move E to skyline candidate set Scandidate which is dominated by the removed skyline object o. According to the distance to the best corner of the search space, Scandidate is structured in heap. By taking Scandidate and existing skyline objects as input, researchers execute the skyline computation algorithm.

Finding of multiple pairs per each iteration: At each iteration skyline related algorithm discovers the best choice in C for each object in Osky. After obtaining the best pair of choice-object (c, o), it has been reported and o is deleted from Osky for maintaining the skyline. If the algorithm returns many firm object-choice pairs in each iteration, researcherscan reduce the number of iterations by using property 1. If o is the best object for c and the c is the best choice for an object o then (c, o) should be firm. In each iteration, assume Cbest ⊂ C which contains every object o ⊂ Osky, the function o.cbest that maximizes c (o). For each c ∈ Cbest, researchers confirm the object c.obest ∈ Osky which maximizes c(o). Then, researchers find out all the pairs which satisfy property 1. Especially, researchers can scan Cbest for each c, researchers make sure whether (c.obest).cbest = c. If it is true (c, c.obest) is a firm pair and the corresponding object and choice is removed from O and C.

And researchers assure that at least one pair is reported with the highest value and the object is removed from Osky. If many pairs are reported then many skyline objects are removed from Osky which does not make any changes in the execution of the UpdateSkyline module in Skyline related algorithm. All entries in the plist of these objects are stored either in the plist or it has been enheaped and then the process is carried over by maintaining the incremental skyline.

EXPERIMENTAL RESULTS

Researchers assess the performance of the algorithm by comparing it with Brute Force Method. Now researchers are proving it with the help of dataset. All methods are implemented in Dot Net and experiments are done on Intel Core2Duo 2.66 GHz CPU machine with 4 GB memory.

Researchers used two different types of dataset, one is independent dataset where the values are generated independently and uniformly and another dataset is anti-correlated where the objects are good in one dimension but not in others. These datasets are usual benchmarks for preference based queries. The experimental database has D dimensions. www.99acres.com is a website provides all informations about the real estate. It has four attributes: property type, location, number of bedrooms and price.

Researchers indexed the dataset with the help of R-tree data structure. Researchers follow LRU memory buffer with 2% of tree size. The preferable choices are generated independently with weights as linear.

Researchers compare the algorithms with datasets of uniform and anti-correlated objects with D dimensions. In Fig. 3, researchers can see SR algorithm proves that it is better than Brute-Force. It can be achieved by its skyline maintenance of the SR algorithm. The I/O

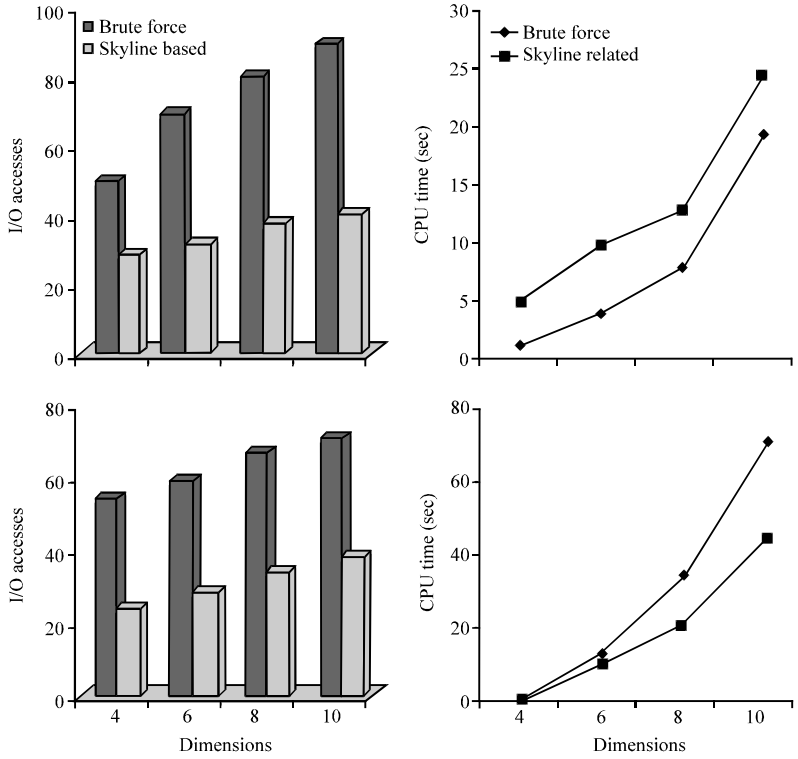

Fig. 3: Effects of D dimensionality. a) I/O cost (Independent); b) I/O cost (Anti-correlated); c) CPU time (Independent) and d) CPU time (Anti-Correlated)

cost increases with the Dimensions D due to the object R-tree degrades. In terms of CPU cost also, the SR algorithm performs well.

CONCLUSION

In this study, researchers have taken stable marriage problem between a set of objects O and a preference choices C. The choices specify their requirements and according to it, the algorithm selects a suitable object as one to one assignment. the algorithm computes firm pairs objects in skyline of O. Researchers have implemented incremental maintenance technique to update the skyline of the objects efficiently. the results show that the proposed algorithm outperforms well when compared with the existing methodologies.

REFERENCES

Beckmann, N., H.P. Kriegel, R. Schneider and B. Seeger, 1990. The R-tree: An efficient and robust access method for points and rectangles. Proceedings of the ACM-SIGMOD International Confrence on Management of Data, 1990, University of British Columbia, pp: 322-331.

Bohm, C. and F. Krebs, 2002. High performance data mining using the nearest neighbor join. Proceedings of the IEEE International Conference on Data Mining, December 9-12, 2002, Maebashi City, Japan, pp: 43-50.

Borzsonyi, S., D. Kossmann and K. Stocker, 2001. The skyline operator. Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, pp: 421-430.

Brito, M., E. Chaves, A. Quiroz and J. Yukich, 1997. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Stat. Probab. Lett., 35: 33-42.

Chan, T.M., 2006. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, January 22-26, 2006, Miami, Florida, USA.

De Berg, M., M. van Kreveld, M. Overmars and O. Schwarzkopf, 2000. Computational Geometry: Algorithms and Applications. 2nd Edn., Springer Verlag, USA.

Ding, C. and X. He, 2004. K-nearest-neighbor consistency in data clustering: Incorporating local information into global optimization. Proceedings of the ACM Symposium on Applied Computing, March 14-17, 2004, Nicosia, Cyprus, New York, pp: 584-589.

Fagin, R., A. Lotem and M. Naor, 2003. Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci., 66: 614-656.

Gowda, K. and G. Krishna, 1978. Agglomerative clustering using the concept of mutual nearest neighborhood. Patt. Recog., 10: 105-112.

Gowda, K. and G. Krishna, 1979. The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (Corresp.). IEEE Trans. Inf. Theor., 25: 488-490.

Hjaltason, G.R. and H. Samet, 1998. Incremental distance join algorithms for spatial databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA., pp: 237-248.

Hjaltason, G.R. and H. Samet, 1999. Distance browsing in spatial databases. ACM Trans. Database Syst., 24: 265-318.

Jagadish, H.V., B.C. Ooi, K.L. Tan, C. Yu and R. Zhang, 2005. Distance: An adaptive B^+-tree based indexing method for nearest neighbor search. TODS, 30: 364-397.

Papadias, D., Y. Tao, G. Fu, and B. Seeger, 2005. Progressive skyline computation in database systems. ACM Trans. Database Syst., 30: 41-82.

Tao, Y., V. Hristidis, D. Papadias and Y. Papakonstantinou, 2007. Branch-and bound processing of ranked queries. Inform. Syst., 32: 424-445.

Wong, R.C.W., Y. Tao, A.W.C. Fu and X. Xiao, 2007. On efficient spatial matching. Proccedings of the VLDB, September 23-28, 2007, Vienna, Austria, pp: 579-590.