International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 27241-27248
© Research India Publications
http://www.ripublication.com

Targeting Enemies Using Reverse Skyline Query in Spatial Game†

Jongwan Kim

Department of Computer Engineering, Sungkyul University, Korea
Email: kimj@sungkyul.edu

Dukshin Oh[1]

Department of Management Information Systems, Sahmyook University, Korea
Email: ohds@syu.ac.kr

Abstract

With the continued advances of network technologies, mobile devices with positioning technologies are leading online games in imitation of the real world. Such gamesprovide us with new experiences and challenges for winning games where the game space is set similar to the real world (i.e., spatial games). Searching enemies (or sets) is important to achieve victory inthese games. We show how to identify enemies using reverse skyline (RS)query and suggest ideas for winning spatial games. Conventional schemes compute the distance from a gamer to the enemies using Euclidean distance, and compare all objects to find the nearest enemies. However, RS prunes many objects and decreasescomputation costs. Decreasing time is required for fast game processing and support services to users. We simulate this scheme based on various data sets, and show that RSquery is better than conventional distance schemesfor searchingthe nearest enemies.

Keywords: Skyline, Reverse Skyline, Reverse Nearest Neighbor, Spatial Game

---

†This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2014R1A1A2059902).
[1]Corresponding author: Dukshin Oh.

1 Introduction

The spatial objects of games are in locations that use spatial coordinates such as 2D, 3D, or GPS data (synthetic or real). In spatial games, finding enemies is a shortcut to victory. Most games search the nearest objects or enemies using Euclidean distance; however, all objects are compared, which causes computation costs to be high. Guaranteeing that the nearest objects found through Euclidean distance are those nearest from a gamer is difficultbecause the same objects can be nearest to another gamer.

It is possible to identify enemies around a gamer by computing the Euclidean distance from the gamer to the enemies. However, comparing objects is computationally expensive,whereas skyline query prunes many objects in a given space. Skyline query [1] searches objects in a database area, and provides good performance. The objects arethose nearest to the gamer (query point) side, but the gamer might not be the one nearest to the objects.

The basic idea of reverse skyline (RS) query is to find objects in a given data set whose objectsare given query points based on skyline query. The concept of reverse query is of particular interest in a wide range of applications, such as marketing or decision-making systems. In spatial games, RS is more attractive for identifying other gamers, who might be the enemy. To the best of our knowledge, no one has applied this scheme to spatial gamesat this point. We believe that this is the first approach for targeting enemies using RSquery on spatial game objects.

Typically, spatial games use Euclidean distance to allow gamers to search the nearest game objects in order to avoid or meet them. However, this scheme requires a high computation cost because a gamer is compared with others to find the appropriate objects. High computation costs may result in lazy performance. On the contrary, RS offers gamers good performance by pruning objects according to their feature.

The rest of the paper is organized as follows. Section 2 provides related works, such as traditional, global, and reverse skyline queries.Section 3 describes the background and motivation of our idea, and Section 4 introduces the process oftargeting the nearest enemies using RS query.Section 5 shows simulations and discusses the results. Finally, the conclusion is presented in Section 6.

2 Related Works

We apply RSquery to the nearest enemy set in spatial games, but applying this scheme directly is not sufficient. We briefly survey the traditional, global, and reverse skylines in order to demonstrate their performance in finding enemies sets on skylines. The two former skylines are the basis for understanding the latter.

2.1 Traditional skyline (TS)

The increasing data size in various applications has required supporting complex queries in order to search targeted objects in such data. Skyline query has received considerable attention and been applied in many applications, such as uncertain data query [2], sensor network [3], and mining [4] areas since its introductionin 2001. For example, skyline showsthe relationship between distances near a beach and price of hotels,as shown in Figure 1.

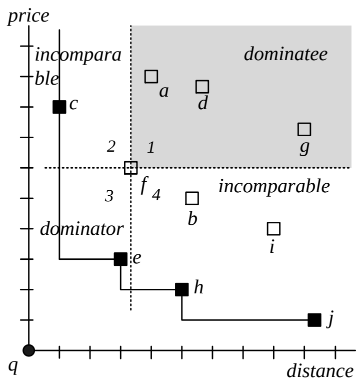


Fig. 1. Traditional skyline

Skyline query has been used to search the attributes of data objects. There are dominators and dominatees that a data object discards from the data space because they are not required for a user to make a choice. In Figure 1, data object *f* dominates all the objects in the first quadrant(depicted in gray). The second and fourth quadrantsare incomparable at this time, but they should be comparable in the skyline later. Skyline is defined as follows:

Definition1. Let a query point be *q*, and the dataset be *P*. When a data object *p1∈P* dominates *p2∈P*, *p1* is an object on the skyline. That is, *p1* dominates *p2* if and only if it is not worse than *p2* for all dimensions *i*, and *p1* is better than *p2* for at least one dimension *j*. The notations *i* and *j* are applied with the same meaning in this section.

$$i \in \{1, \dots, d\}: p_1^i \leq p_2^i \; and \; j \in \{1, \dots, d\}: p_1^j \leq p_2^j \qquad (1)$$

In spatial games, TS cannot be used efficiently to search the objects nearest to a gamer because it is computed from the gamer in the first quadrant only in practice. The nearest gameobjects are around a gamer,which is critical in the case of enemies.

2.2 Global skyline (GS)

From Figure 1, it can be seen that there are objects from all four quadrants around the querypoint that could be the nearest objects. However,TS only searches objects in quadrant 1. The basic idea of GS is to search skyline objects on each quadrant and aggregate them in one result (Figure 2) [5].

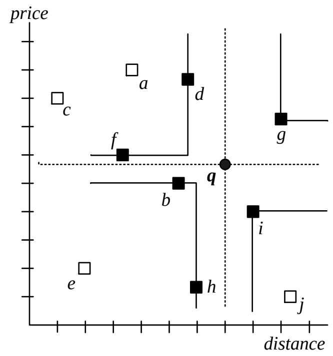


Fig. 2. Global skyline

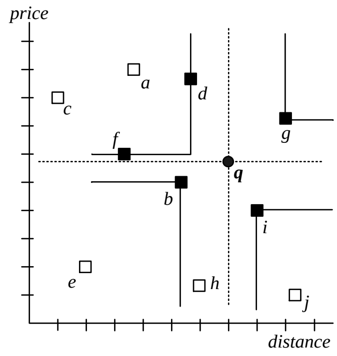


Fig. 3. Reverse skyline

Definition2. When $p1 \in P$ and $p2 \in P$for query point q, $p1$ dominates $p2$under the following conditions:

(1) $i \in \{1, \dots, d\}: (p_1^i - q^i)(p_2^i - q^i) > 0;$

(2) $i \in \{1, \dots, d\}: |p_1^i - q^i| \leq |p_2^i - q^i|$($i$=dimension); and

(3) at least in one dimension j, $j \in \{1, \dots, d\}: |p_1^j - q^j| \leq |p_2^j - q^j|$.

That is, the result set for GS(q) includes GS objects not dominated globally by any objects.

2.3 Reverse skyline (RS)

RS occurs when a query point q has a data point p as its nearest neighbor (NN), and q is the NN of p in reverse. For example,assume that there are three points $a<b<c$ placed on a straight line according to their distance. In this case, b is the NN of a, and similarly,c is that of b. However, $a$cannot be guaranteed to be the NN of b, nor $b$that of c. RSquery was proposed by [5], which retrieves data objects in the database.

Definition3. Let P be data in 2D. The RSquery of a query point q retrieves all data objects in $p1 \in P$, and it satisfies the following equation:

$$i \in \{1, \dots, d\}: |p_2^i - p_1^i| \leq |q^i - p_1^i| and\ j \in \{1, \dots, d\}: |p_{2-}^j p_1^j| < |q^j - p_1^j| \qquad (2)$$

3Background and Motivation

Computing the distance between a gamer and its enemies can influence the game's score by allowing the gamer to fight or avoid the surrounding enemies. Conventional spatial games use the Euclidean distance between objects in a space. Here, a gamer faces an object by calculating the distance and preparing to battle such object, but this particular object is not concerned witha gamer that might not be anNN. For example, in Figure 4, the NN of query point q is c, but q is not the NN of c because c is nearer to q'.

In this situation, *c*is a target of *q'*, or *c* knows *q'*more than *q* in its distance. Therefore, it is difficult to identify enemy setsthrough Euclidean distance.

Another limitation of the conventional scheme is that query points should be compared with all those objects around a gamer in order to search the nearest object, which results in high computation costs.

To overcome the aforementioned limitations, first, we decrease the number of objects to be compared using skyline in advance. Then, if a gamer identifies a unique enemy set, he/shecan obtain a better score in the game level.Skyline discards more objects than the conventional NN query using Euclidean distance, as shown in Figure 1. Next, weapply RSquery to overcome two limitations as shown before. RS helps to find the nearest enemies between a gamer and enemies, and supports fast computation.

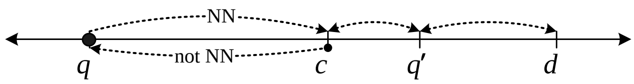


Fig. 4. NN with Euclidean distance

4How to Target Nearest Enemies with RS Query

In this section, we present our scheme for targeting the nearest enemies in a spatial game. We show that GS can identify those objects that surround the gamer, and then provide a method fordetermining the reverse objects from a query point in a given space. We make some assumptions that query point *q* is the gamer's position, and that the other letters in the alphabet represent the enemies. The terms *object* and *enemy*have the same meaning, and we use them interchangeably in this paper.Also, identifying our troops among game objects is not considered in this paper because it is straightforward in implementation.

4.1 Aggregate objects on GS

Figure 1 shows the result of TS(*q*)={*c, e, h, j*}, and this result is not appropriate for a gamer because the gamer can only identify the objects in quadrant 1. That is, when the query point moves to another location, such as Figure 2, TS identifies game object*g*, with the exception of quadrants 2 to 4.

In Figure 2, GS shows the objects around a gamer, and then TS(*q.1*) = {*g*}, TS(*q.2*) = {*d, f*}, TS(*q.3*) = {*b, h*}, and TS(*q.4*) = {*i*} in each quadrant, respectively. GS(*q*) aggregates these TS in a set, and therefore, there is no doubt that by the definition of GS,this is GS(*q*)={*g, d, f, b, h, i*}.

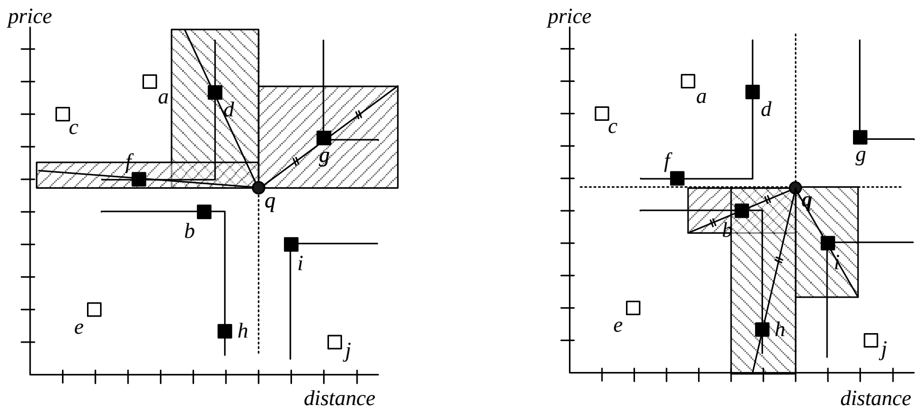
Computing GS is simple, and similar to skyline query, it provides us all the objects around the query point, which is different from the TS query. This decreases computation costs compared withthe conventional NNquery that compares all objects because the objects are pruned through the relationship between objects,such as dominators or dominatees.

4.2 Identify reverse objects

GS provides us a method where the gamer searches the surrounding objects. However, one such object might not be the NN object by the definition of reverse NN.

Figure 5 depicts the process for identifying objects in four quadrants. Figure 5(a) shows the drawing of a box from query point *q* to the endpoint of*g*, which is twice the distance from *q* to *g*. If there is no other object aside from *g*, no object is in the box. Quadrant 1 has one object, and it is the reverse nearest object of *q*.Figure 5(a) shows that quadrant 2 has two GS objects. We draw a box with *d*at the center, and compare all objects with the query point to identify reverse NNs in the box. If there are no objects in the box, *q* is anNN of *d*, and their reverse relationship is the same. Remember that reverse NN is where the objects are NNs of each other in a region. Two objects *d* and *f* arethe nearest objects to *q*, and they are also RS objects. There is no further comparison because there are no objects in the boxes.

Figures5(b) shows two and one GS objects, respectively. In quadrant 3, the box for *h* includes object *b*, and therefore,*h*is not the reverse object of *q* because there is anearer object *b*than *h*. However,*b* is the nearest object to *q*, that is, RS(*q*)={*g, d, f, b, i*} in Figure 5.



(a) Quadrant 1 and 2 **(b) Quadrant3 and 4**

Fig.5. Identifying reverse skyline objects

RS has two phases for searching the reverse nearest objects. One is to prune objects using skyline in all quadrants, and the other is to filter the objects to obtain those in a reverse relationship. All the remaining objects are only those nearest to *q*.

4.3 Targeting enemies in spatial games

Victory is important in games. Gamers have to calculate their enemies' locationsin order to avoid or fight them. An object'slocation is presented in 2D through latitude and longitude (or (x, y) in 2D-graph). In general, the distance between a gamer and its objects is computed using the Euclidean distance. This scheme compares all data to find the nearest enemies,which increases the computation time,thus resulting in delayed game speeds. For this reason, the Euclidean distance is not appropriate for targeting enemies.

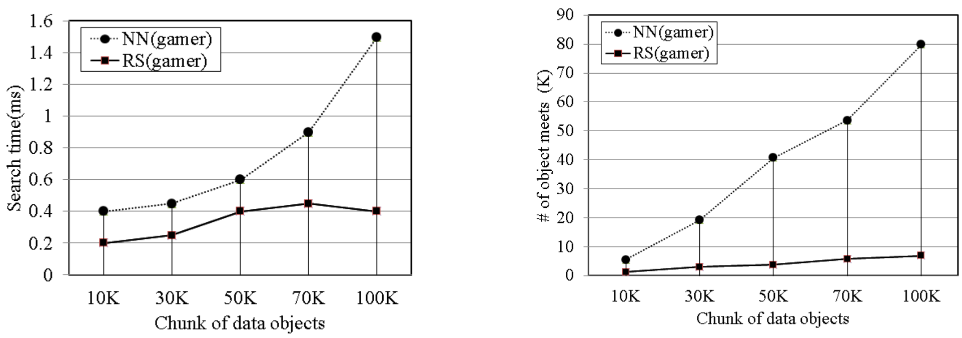
Targeting enemies means making a set of enemiesthat are dangerous to the gamer because they arethe obstaclesfor obtaining victory in a game. As Figure 5 shows,RS supportsan efficient method for searching enemies and indicates a way for winning in spatial games.

5Simulation and Discussion

We implemented NN and RS, andthen compared the results with various options. A simulation was processed on a server with Windows Server 2012, 2 CPU, and RAM of 16GB; we also generated 100 K tuple objects using a synthetic data generator. The data objects were indexed inR-tree [6] with location and other properties.The data set includes uniform and skewed distribution ofthe generated synthetic data, and it is separated into five data chunks, such as 10 K, 30 K, 50 K, 70 K, and 100 K. We focus on searching time and times of comparison to obtain RS objects in the NN query.

The results of the simulation are shown in Figure 6(a) for search time, and Figure 6(b) for the number of object meets. First, we identified the RS reverse objects, and then attempted to search the same objects with NN in order to compare the search times. In Figure 6(a), the time for NN increases according to chunk size because NN compares more objects than RS to determine those that are nearest to the gamer.

Figure 6(b) shows the number of objects that NN and RS meet in the simulation. NN increases up to 100 K objects, but RS changes insignificantly compared with NN in every chunk. The reason is that RS prunes many objects in the comparison step; therefore, the target objects to be compared are fewer than for NN.



(a) Search time for reverse objects **(b) Number of meets**

Fig. 6. NN and RS performance comparison

6Conclusion

In this paper, we resolved the problem of the high computing costs of NN for reverse objects, and showedwhy RS objects are important in spatial games.We proposed applyingRSquery to spatial gamesto search enemies and achieve victory promptly. To the best of our knowledge, this work is the first whereskyline is applied to games, but this is necessary work in order to satisfy users' needsfor accelerating the game.In the simulation, RS showed better performance than NN query for searching enemies in a reverse manner. This means that applying RS to spatial games guarantees game speed, and avoids enemy setsto allow safer gaming.

References

[1] S. Borzsonyi, D. Kossmann, K. Stocker,"The Skyline Operator," In ICDE, (2001), 421–430.
[2] X. Li, Y. Wang, X. Li, G. Wang, "Skyline Query Processing on Interval Uncertain Data," Intl. Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, (2012), 87–92.
[3] B. Yin, Y. Lin, J. Yu, Q. Luo, "Energy-efficient Filtering for Skyline Queries in Cluster-based Sensor Networks," Computers and Electrical Engineering, Elsevier, 40, (2014), 350–366.
[4] Z. Huang, S. Sun, W. Wang, "Efficient Mining of Skyline Objects in Subspaces over Data Streams," Knowl. Inf. Syst., 22, (2010), 159–183.
[5] E. Dellis, B. Seeger, "Efficient Computation of Reverse Skyline Queries," VLDB, (2007), 291–301.
[6] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching,"ACMSIGMOD Int. Conf. on Management of Data, (1984), 47–57.