



A Novel Approach for Approximate Spatio-Textual Skyline Queries

Seyyed Hamid Aboutorabi, Nasser Ghadiri^(✉), and Mohammad Khodizadeh Nahari

Department of Electrical and Computer Engineering, Isfahan University of Technology,
84156-83111 Isfahan, Iran
{h.aboutorabi, m.khodizadeh}@ec.iut.ac.ir,
nghadiri@ec.iut.ac.ir

Abstract. The highly generation of spatio-textual data and the ever-increasing development of spatio-textual-based services have attracted the attention of researchers to retrieve desired points among the data. With the ability of returning all desired points which are not dominated by other points, skyline queries prune input data and make it easy to the user to make the final decision. A point will dominate another point if it is as good as the point in all dimensions and is better than it at least in one dimension. This type of query is very costly in terms of computation. Therefore, this paper provides an approximate method to solve spatio-textual skyline problem. It provides a trade-off between runtime and accuracy and improves the efficiency of the query. Experiment results show the acceptable accuracy and efficiency of the proposed method.

Keywords: Skyline query · Spatio-textual data · Approximated method

1 Introduction

The increased number of location-enabled devices such as smartphones and the development of social networks, including Twitter, Flickr, and Foursquare, have generated a massive amount of spatio-textual data in recent years. Also, location-based search services that use such data have been rapidly developed [1]. According to the statistics from a report published in 2015, about 20% of daily Google queries and 53% of smartphone users' searches are associated with spatial content [2, 3]. An accurate response to such queries, demands models that make the best possible trade-off between different features of data (here spatial and textual features) and facilitate decision-making process through dataset purification.

The skyline query was developed to cover such problems. It was first introduced by Börzsönyi et al. [4] in the database research field, and then received the attention of computer science researchers so that they offered methods to solve it [5, 6]. Consequently, its initial definition was gradually expanded and various issues were put forward based on the initial skyline. The spatio-textual skyline query was first introduced by Shi et al. [7]. It is only one of the various types of skyline queries with a set of data points and a set of query points; each point has spatial data and a set of textual descriptions. The spatio-textual distance between the data points and query points is measured using the integration of spatial distance and the textual relevance between these points. As the

output of the query, the skyline set returns all data points that are not spatio-textually dominated by other data points. A data point (p_2) is spatio-textually dominated by another data point (p_1), if spatio-textual distance of p_2 to all query points is not better than p_1 and spatio-textual distance of p_2 to at least one query point is worse than p_1 .

Spatio-textual skyline queries are among the newest and the most challenging queries of skyline problems. This is because skyline query has rarely been studied on spatio-textual data. Although there are studies on spatial skyline queries, the textual dimension is a new area of focus, which has been less studied. It should be noted that the top- k queries have previously been studied on spatio-textual data. However, they cannot be adopted in some applications because of the unknown weights of the different dimensions of data. On the other hand, the skyline queries are very costly from both processing time, and disk usage points of view [8, 9] and the addition of spatio-textual features to the queries will increase these costs. These challenges demand a new efficient approach. The primary aim of this research is to introduce an efficient approximate approach for spatio-textual skyline computation with acceptable accuracy and efficiency.

Our new approach does not necessitate the investigation of whole data space, and it shows, in many cases, a better performance than other approaches by making a trade-off between accuracy and efficiency. This is the main advantage of the proposed approximate method.

This paper has been organized as follows. In Sect. 2 we present related work. Section 3 discusses the proposed approach to executing spatio-textual skyline queries. In Sect. 4, the obtained results are analyzed and Sect. 5 concludes the paper and presents recommendations for future studies.

2 Background

From the time of its introduction by Börzsönyi et al., the skyline family of database queries has widely been studied. According to the classification shown in Fig. 1, three areas serve as the source of the diversity of the skyline challenges: data diversity, different execution environments, and different computation methods. Data diversity is the first influential factor. Handling deterministic data are the simplest case of this class where each data point has a set of features which are complete, fixed and reliable from the beginning of algorithm, and lacks any spatio-textual features. Some algorithms, including divide and conquer and bitmap skylines, are executed on such data [4, 10]. Although these algorithms are the basis of other methods, they have received less attention of researchers due to introducing new requirements.

A challenging class of skyline query focuses on spatial data. In this category, the Euclidean distance-based algorithms were first introduced in Sharifzadeh and Shahabi's work to solve spatial skyline problems [11]. The difference between the spatial skyline and spatio-textual skyline queries is that in the former, the data points and query points, lack textual descriptions. Lee et al. proposed a set of accurate and approximate approaches to solve this type of queries [12].

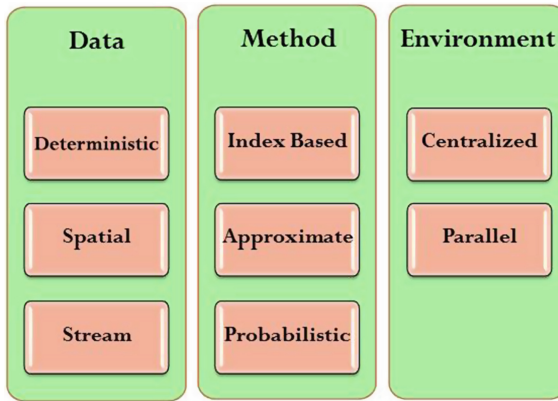


Fig. 1. Source of the diversity of the skyline challenges

Stream data are another type of data which is not available at the beginning of algorithm entirely. They are incomplete data; therefore, prior approaches are not applicable to them. There are different papers about spatial skyline query [13, 14] and this study concentrates on spatio-textual data.

Selection of the solution approach is another influential factor. It depends on the attitude of the researcher towards the skyline problem and his/her aim of solving the problem. The problem-solving methods can be divided into three categories: index-based, approximate and probabilistic methods. In the index-based methods, different index structures including B-tree, R-tree, ZB-tree, and Zinc are used to increase the efficiency of algorithms [15–17]. Although index-based methods do not necessitate the scanning of entire data space, they add additional storage overloads. Our proposed method uses simple database-ready built-in indexes to cope with the problem. In the approximate methods, such as those proposed by Chen and Lee [18], all or parts of skyline set along with the set of points which are closer to the skyline points may be returned to the user to satisfy the user or to increase algorithm efficiency. Our developed method is an approximate method that will be explained in the next section. Probabilistic methods are another class of skyline processing methods where the probability may be associated with data and users' preferences. Sacharidis et al. first presented an algorithm to introduce this method [19] followed by several algorithms. In such algorithms, the user defines a probability threshold and an error value. All responses returned as the skyline set should meet this threshold considering the defined error value. Our work considers non-probabilistic preferences.

The execution environment and hardware platform is another influential factor on which the problem is introduced and solved. There have been studies on the quality of the execution of skyline query on mobile-based networks, cloud platform, peer to peer networks, ad-hoc networks and wireless sensor networks [20]. These studies can be divided into two classes: centralized and parallel. Certain requirements should be taken into account in the design of the algorithms for each architecture. In the centralized methods, memory restriction is the only limitation of algorithm while parallel methods go with several challenges including concurrency, load balancing on CPU cores,

resource management, software and hardware failures and task scheduling. Different methods have been introduced to develop skyline parallel algorithms [21, 22]. Our hardware structure is a conventional centralized and standalone computer.

3 Method

3.1 Spatio-Textual Skyline Query Model

A spatio-textual skyline query gets a set of data points and query points as inputs, and returns those data points which are not spatio-textually dominated by other data points considering the location and preferences of query points. The spatio-textual skyline query contains two data sets: a set of data points (P) on which the query should be executed, and a set of query points from which the query is issued. Each member of the mentioned sets has a geographical coordinate shown as $p.\gamma$ for data points and as $q.\gamma$ for query points. In addition, each data point has a textual description shown as a set of keywords $p.\psi$. All query points have the same textual description indicating users' preferences and shown as $Q.\psi$. According to [7], a data point (p_i) will spatio-textually dominate another data point (p_j) if Eq. (1) holds:

$$\forall q_k \in Q: st(q_k, p_i) \leq st(q_k, p_j), \exists q_k \in Q: st(q_k, p_i) < st(q_k, p_j) \tag{1}$$

In Eq. (1), $st(q_k, p_i)$ is a model for integrating spatial distance and the textual relevance of data points and query points. The relationship between spatio-textual distance with spatial distance and textual relevance is direct and inverse, respectively:

$$st(q_k, p_i) = \frac{d(q_k, p_i)}{w(Q.\psi, p_i.\psi)} \tag{2}$$

In Eq. (2), $st(q_k, p_i)$ shows spatio-textual distance. The term $d(q_k, p_i)$ shows the spatial distance between data points and query points which is computed using Euclidean distance on the geographical latitude and longitude of the points. In addition, $w(Q.\psi, p_i.\psi)$ shows the textual relevance between the keywords of a data point and the keywords of query points. This relevance is calculated as shown in Eq. (3):

$$w(Q.\psi, p_i.\psi) = \prod_{t_j \in Q.\psi} (\widehat{w}(t_j, p_i.\psi)) \tag{3}$$

In Eq. (3), $\widehat{w}(t_j, p_i.\psi)$ indicates the weight of the keywords t_j of data point p_i . Equation (3) is interpreted as follows: To compute the relevance between the keywords set of query points $Q.\psi$ and the keywords set of data points $p_i.\psi$, the corresponding weights should be defined for each keyword Q available in $p_i.\psi$ and the product of the weights should be considered as the textual relevance between them. If only a few weights are zero, a reduction coefficient is considered for them, instead of considering their value as zero. This will prevent the value of the term to become zero. However, if all weights

are zero, no textual relevance is considered between the points. $\hat{w}(t_j, p_i, \psi)$ is computed using *tf-idf* method [23] as follows:

$$\hat{w}(t_j, p_i, \psi) = \frac{1}{|p_i \cdot \psi|} \times TF \times \log \frac{N}{|\{p_i \in P : t_j \in p_i \cdot \psi\}|} \tag{4}$$

In Eq. (4), TF is the total frequency of a keyword in the textual description of a data point and N is the total number of data points, (|P|). The whole term is divided by $|p_i \cdot \psi|$ in order to give more value to low frequent words.

The corresponding weight of the keywords of data points are computed before running the algorithm, and the outputs are stored in a MongoDB database. The MongoDB database is used because the corresponding weights of the keywords of data points are non-structured data and MongoDB is an optimized tool for managing such structure.

3.2 Propose Method

Initial assessments of spatio-textual skyline query results show that in most cases, the ideal points appear around the convex hull of query points. Theoretically, this may root in two reasons. The first one is that those data points which are in the vicinity of the convex hull of query points are more likely to be selected because their distance from query points is lower than that of other data points. The second reason is the accumulation of data points around the convex hull of query points. In this case, there are sufficient data points with a high textual relevance with query points. These points dominate other data points and are introduced as skyline points. The black solid line pentagon shown in Fig. 2 shows the convex hull of query points. As the area of the convex hull decreases/increases (dash line pentagons), algorithm accuracy decreases/increases. Also, the problem space becomes smaller/larger proportional to the changes. This significantly influences algorithm efficiency. If the area of whole data space is S and the area of circumscribed quadrilateral of the convex hull of query points is S_q , ΔS_q will be defined as follows:

$$\Delta S_q = S_{q_2} - S_{q_1} \tag{5}$$

In Eq. (5), S_{q_1} and S_{q_2} respectively are the area of the circumscribed quadrilateral of the convex hull of query point before and after its expansion, respectively.

Then, the α coefficient is defined as follows:

$$\alpha = \Delta S_q / S \tag{6}$$

Based on Eq. (6), α is the ratio of the changes in the area of the circumscribed quadrilateral of the convex hull of query points to the total area of data space. It belongs to [0,1) interval and varies by the variation of S_q . The optimal value of α is obtained by time-accuracy trade off. The effect of α on the accuracy and efficiency of the proposed method will be discussed in the experiments section.

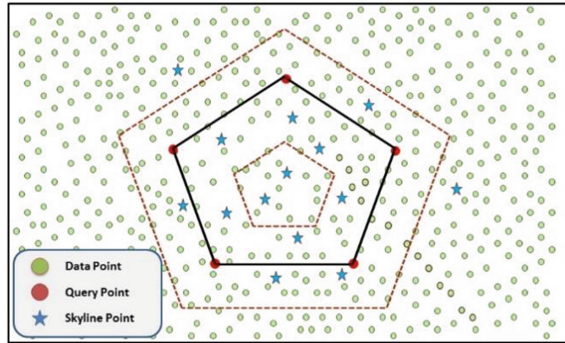


Fig. 2. The main idea of the proposed algorithm

Algorithm 1 shows the pseudo code of Approximate Centralized Method (ACM). It receives a set of data points and a set of query points as inputs. Then, it computes spatio-textual skyline for this query and returns the set of data points with the maximum match with users' requests. The convex hull of query points is first created (line 1), then the optimal value of alpha is computed based on the created convex (line 2) and accuracy required. The optimal value of alpha is predetermined with respect to query points and required accuracy. Then convex is expanded considering alpha (line 3).

Algorithm 1. Approximate spatio-textual skyline

Approximate_Spatio_Textual_Skyline (DataPoint P[], QueryPoint Q[])

1. CH = Create convex hull of Q
 2. α = Optimal Value of α based on created convex(CH) and accuracy required
 3. CHE = Expand (CH, α)
 4. **for** each $p_i \in CHE$
 5. TF = Read corresponding weights of p_i 's keywords from MongoDB Database
 6. TR = Compute textual relevance between $p_i.\psi$ and $Q.\psi$ using TF
 7. **if** (TR \neq 0)
 8. **for** each $q_j \in Q$
 9. SD_j = Compute spatial distance between $p_i.\gamma$ and $q_j.\gamma$
 10. $STD_j = SD_j / TR$
 11. skyline_set = Skyline ($p_i.id, STD[]$)
 12. **return** skyline_set
-

Then, the spatio-textual distance of each data point from query points is computed, and it is investigated that whether they can be present in the final skyline set. In line 5, the corresponding weights of the keywords of data points are retrieved from the MongoDB database. In Line 6, the textual relevance between the keywords of data points and query points is computed. If the relevance is zero, the corresponding data point is pruned without additional consideration because the denominator becomes zero in Eq. 2, making spatio-textual distance reach infinity. If the relevance is not zero, the spatial distance between query points and data points is computed (line 8 to 10).

Line 11 calls skyline algorithm and compares current data point with current members of skyline set. Then, the variable of *skyline_set* changes depending on the comparison result so that some members may be eliminated, some new members may be added, or the skyline set may remain unchanged. Following the evaluation of all current points of data set, *skyline_set*, which contains the final skyline set, is returned as the output.

3.3 Experimental Evaluation

This section assesses the proposed algorithm regarding accuracy and efficiency. Instead of monitoring whole data space, this approximate algorithm assesses only points in the vicinity of the convex hull of query points. Therefore, in some cases, it may fail to find all skyline points. The accuracy of this algorithm depends on alpha to a large extent so that by selecting a suitable alpha it can achieve an accuracy of 95%, and even higher. The next section measures the effect of alpha on the accuracy using sensitivity and specificity criteria:

$$Sensitivity = \frac{TP}{TP + FN} \quad (7)$$

$$Specificity = \frac{TN}{TN + FP} \quad (8)$$

In Eq. (7), TP is skyline points truly marked by the algorithm. FN is skyline points falsely marked by the algorithm as non-skyline points. In relation (8), TN is non-skyline points truly marked by the algorithm as non-skyline points, and FP is non-skyline points falsely marked by the algorithm as skyline points. Relying on more than thousands of tests, it can be argued that the specificity is always higher than 99.9%. Therefore, it is neglected and the accuracy of approximate algorithms is measured and declared only based on sensitivity.

The proposed algorithm is not a recursive algorithm, and the number of data points and query points is not endless. It is, therefore, naturally proven that the proposed algorithms are terminable. To assess the efficiency of proposed algorithms, we designed tests by changing the number of data points, the number of query points and the number of the keywords of query points, and measured algorithm runtime in all cases. The next section provides the results.

4 Experiments

In this section, we report on the results of the experiments. First we describe the evaluation platform and the datasets used, followed by results and their interpretation.

4.1 Test Specifications-Methods and Data Set

The tests were conducted on a system with Intel core i5 4460 CPU, 8 GB of RAM, and 1 TB hard disk. We developed the code for Approximate Centralized Method (ACM)

and Exact Centralized Method (ECM) to provide accurate assessments. They differ in that ECM evaluates all data points, instead of evaluating only data points in the vicinity of the convex hull of skyline points, to raise accuracy to 100%. The methods mentioned were compared with three methods presented in Shi et al. paper [7] i.e., ASTD, BSTD and BSTD-IR. Amongst them, ASTD shows a better efficiency. The comparison was made to show the improvements in the proposed methods.

The datasets were created in accordance with the standards indicated in Börzsönyi et al. paper [4] to evaluate different methods. The number of the keywords of each data point was variable and was 10 keywords on average. The variable parameters of the problem were alpha, number of data points (default = 400,000 points), number of query points (default = 10 points) and number of the keywords of query points (default = 10 points). The effect of these variables on the efficiency and accuracy of the studied methods were evaluated by conducting separate tests. The execution of the algorithm was measured from the moment that algorithm starts running until the final output is recorded. Each query was executed 100 times to avoid any deviation in results, and the average values of the obtained results were depicted.

4.2 Effect of Alpha on ACM Accuracy

This experiment was made to find the optimal alpha of ACM by which an acceptable trade-off is provided between the execution time and accuracy. To this end, separate tests were conducted with different S_q (from 0.1% to 12.8% of total data space) and different values of alpha (0% to 0.1% with an interval of 0.02). Figure 3 shows the results where the vertical and horizontal axes show the accuracy and runtime, respectively. Different values of S_q are shown by distinct colors. In addition, each group of bars shows results associated with a given alpha. For example, the farthest left group bar shows values associated with $\alpha = 0$ and, the farthest right one shows values associated with $\alpha = 0.1$. Observing the results of this test reveals that as alpha rises from zero to 0.1, the accuracy and execution time of the algorithm will increase. Surprisingly, as alpha exceeds a given value, the accuracy remains almost constant. This threshold value is defined as the optimal alpha. For example, the optimal alpha of $S_q = 0.01\%$ is defined to be 0.04. This figure shows another important point. As S_q rises, the optimal alpha increases proportionally. For example, for $S_q = 0.08$, the optimal alpha is defined to be 0.06.

In summary, for any given accuracy and S_q the optimal alpha can be determined using this figure. In the next experiments, the value of alpha is selected in a manner that the accuracy of algorithm always remained about 95%.

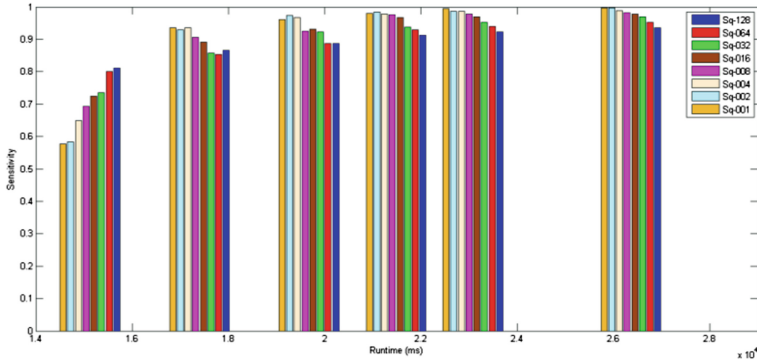


Fig. 3. The effect of the area of the convex hull of query points on the accuracy and efficiency of the studied approximate algorithm

4.3 Effect of the Number of Data Points on Efficiency

In this set of experiments, the number of data points, $|P|$ varied from 100000 to 1500000 and other parameters remained unchanged and were set at their default values. Figure 4 shows the execution time of three competitor algorithms and our two algorithms i.e. ECM (red curve) and ACM (green curve) with an accuracy of about 95%.

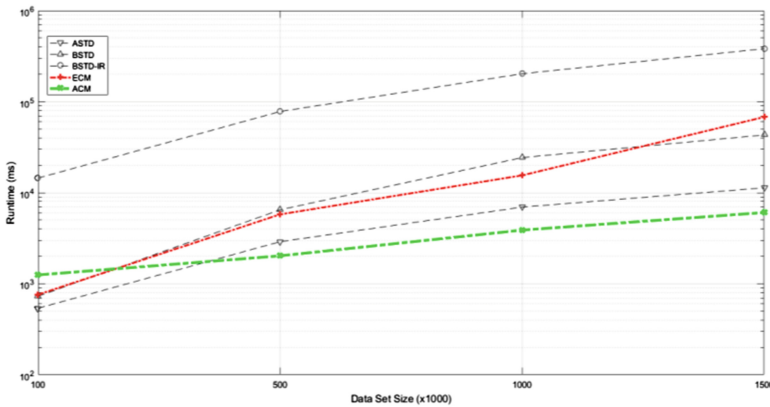


Fig. 4. The effect of data points on execution times of different algorithms

Deliberating in the results of this test reveals three crucial points. The first point is that as the number of data points increases, the execution time of all algorithm increases due to the enlargement of problem space. The algorithms differ with each other in their raising speed, i.e., their slopes. The second point is that the slope of ACM curve is lower than that of other methods. This implies the higher scalability of ACM compared with competitor algorithms. The third point is that in the test with 100000 data points, two competitor algorithms showed lower execution time compared to our ACM algorithm.

This may root in two reasons. First, they use modern index structures like IR-Tree which are optimized for such data. Apparently, the smaller the problem space is, the lower is search structures depth, and lower nodes should be explored to obtain target nodes. This, in turn, shortens algorithm execution time. However, the advantage of our algorithms is well demonstrated in the larger search space.

The second reason is that we tried to keep the accuracy of our algorithm in a constant level in all tests. This demanded accuracy vs execution trade-off. If we ignore the accuracy of 95% for ACM, better execution times could be achieved.

4.4 Effect of the Changes to Query Parameters on Efficiency

Two experiments were designed to show the non-significant effect of changes to query parameters on the proposed algorithm efficiency. In the first experiment, the number of query points $|Q|$ varied from 1 to 40 and in the second one, the number of the keywords of query points, $|Q.\psi|$, varied from 1 to 20. Other parameters were set in their default value and remained unchanged in all tests. According to Figs. 5 and 6, the changes of ACM resemble the changes of ASTD while the changes of ECM resemble the changes of BSTD. In addition, ACM is faster than ECM by 2 to 4 times.

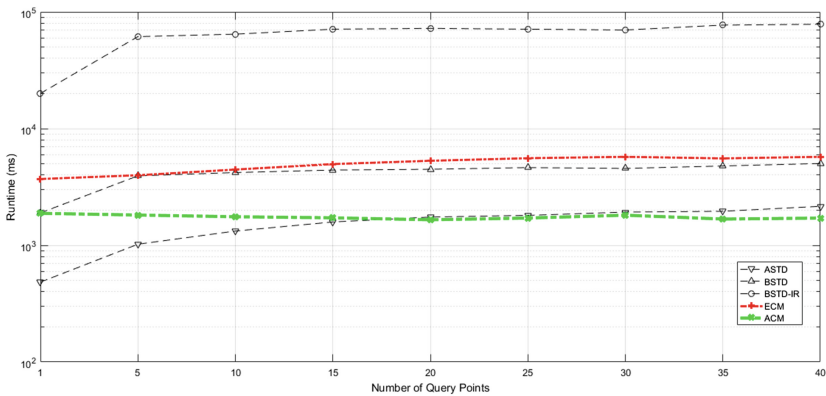


Fig. 5. The effect of *query points* on execution times of different algorithms

Focusing on the results of the tests reveals two crucial points. The first one is that the efficiency of our proposed methods is as good as that of competitors. The second point is that the execution time of algorithms arguably remained almost unchanged despite making a change in parameters. Therefore, any change in parameters has no significant effect on the efficiency of the proposed algorithms.

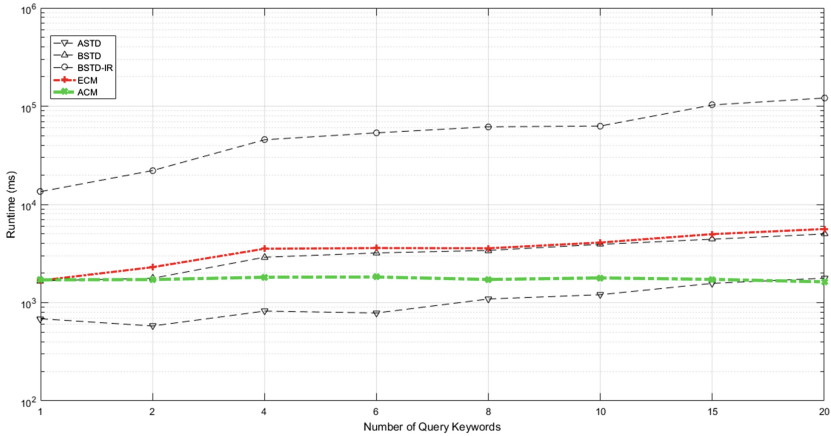


Fig. 6. Effect of the number of the keywords of query points on algorithms runtime

5 Conclusion

Spatio-textual skyline query returns all data points which are not spatio-textually dominated by other points. This paper provided an approximate method for spatio-textual skyline query, assessed its efficiency and compared it with exact version of the method and other competitor algorithms. According to the results of many experiments, the proposed method generally shows a higher efficiency than other methods of this field. If users do not need the accuracy of 100%, they can provide a trade-off between runtime and accuracy in order to neglect the accuracy of final result, to some extent, and get the result in a shorter time, instead. However, this method, and other competitor methods, fails to show good efficiency on large-scale input data. In future, we will provide a method to increase the scalability of spatio-textual skyline query using distributed methods. The implementation of spatio-textual index structures and consolidating them with ACM is another approach for improving the efficiency of the proposed method.

References

1. Bao, J., Mokbel, M.: GeoRank: an efficient location-aware news feed ranking system. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Orlando, Florida, USA, pp. 184–193. ACM, New York (2013)
2. Statistic Brain (2016). <http://www.statisticbrain.com/google-searches>. Accessed 10 Nov 2016
3. Chen, L., Cong, G., Cao, X.: An efficient query indexing mechanism for filtering geo-textual data. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, USA, pp. 749–760. ACM, New York (2013)

4. Börzsönyi, S., Kossmann, D., Stocker K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, pp. 421–430. IEEE Computer Society (2001)
5. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, China, pp. 275–286. VLDB Endowment (2002)
6. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, pp. 717–719. IEEE Computer Society (2003)
7. Shi, J., Wu, D., Mamoulis, N.: Textually relevant spatial skylines. *IEEE Trans. Knowl. Data Eng.* **28**, 224–237 (2016)
8. Mullesgaard, K., Pedersen, J., Lu, H., Zhou, Y.: Efficient skyline computation in MapReduce. In: Proceedings of the 17th International Conference on Extending Database Technology (EDBT), Athens, Greece, pp. 37–48. OpenProceedings.org (2014)
9. Woods, L., Alonso, G., Teubner, J.: Parallel computation of skyline queries. In: IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Seattle, Washington, USA, pp. 1–8. IEEE Press (2013)
10. Tan, K., Eng, P., Ooi, B.: Efficient progressive skyline computation. In: Proceedings of the 27th International Conference on Very Large Databases, Rome, Italy, pp. 301–310. Morgan Kaufmann Publishers Inc. (2001)
11. Sharifzadeh, M., Shahabi, C.: The spatial skyline queries. In: Proceedings of the 32nd International Conference on Very large Databases, Seoul, South Korea, pp. 751–762. VLDB Endowment (2006)
12. Lee, M., Son, W., Ahn, H., Hwang, S.: Spatial skyline queries: exact and approximation algorithms. *GeoInformatica* **15**, 665–697 (2011)
13. Lin, Q., Zhang, Y., Zhang, W., Li, A.: General spatial skyline operator. In: 17th International Conference Database Systems for Advanced Applications: DASFAA, Busan, South Korea, pp. 494–508. Springer (2012)
14. You, G., Lee, M., Im, H., Hwang, S.: The farthest spatial skyline queries. *Inf. Syst.* **38**, 286–301 (2013)
15. Lee, K., Zheng, B., Li, H., Lee, W.: Approaching the skyline in Z order. In: Proceedings of the 33rd International Conference on Very large Databases, Vienna, Austria, pp. 279–290. VLDB Endowment (2007)
16. Lee, K., Lee, W., Zheng, B., Li, H., Tian, Y.: Z-SKY: an efficient skyline query processing framework based on Z-order. *The VLDB J.* **19**, 333–362 (2010)
17. Liu, B., Chan, C.: ZINC: efficient indexing for skyline computation. *Proc. VLDB Endow.* **4**, 197–207 (2010)
18. Chen, Y., Lee, C.: The σ -neighborhood skyline queries. *Inf. Sci.* **322**, 92–114 (2015)
19. Sacharidis, D., Arvanitis, A., Sellis, T.: Probabilistic contextual skylines. In: IEEE 26th International Conference on Data Engineering (ICDE), Long Beach, CA, USA, pp. 273–284. IEEE Press (2010)
20. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. *VLDB J.* **21**, 359–384 (2012)
21. De Matteis, T., Di Girolamo, S., Mencagli, G.: A multicore parallelization of continuous skyline queries on data streams, In: Euro-Par 2015: Parallel Processing: 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, pp. 402–413. Springer (2015)

22. Liou, M., Shu, Y., Chen, W.: Parallel skyline queries on multi-core systems. In: Proceedings of the 14th International Conference on Parallel and Distributed Computing, Applications and Technologies, Taipei, Taiwan, pp. 287–292. IEEE Press (2013)
23. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)