The Journal of Supercomputing
https://doi.org/10.1007/s11227-018-2523-2

Fuzzy clustering-based skyline query preprocessing algorithm for large-scale flow data analysis

Yifu Zeng[1] · Yantao Zhou[1] · Xu Zhou[2] · Fei Zheng[1]

Abstract

In order to improve the effectiveness of the large-scale stream data skyline query preprocessing algorithm, a large-scale stream data skyline query preprocessing algorithm based on fuzzy clustering analysis (kdStreamSky) is proposed in the paper. Firstly, relevant concept of the large-scale stream data skyline query preprocessing algorithm was described; then, in order to solve the problem—curse of data dimensionality of the large-scale stream data skyline query preprocessing algorithm, the corresponding data subset was constructed and meanwhile MapReduce calculation model was combined to realize the mean value clustering center selection for the parallel mobile social network; meanwhile, in order to improve algorithm stability and facilitate the design of information forwarding scheme, a distributed hierarchical clustering method was adopted for the clustering analysis of individuals and the design of hierarchical forwarding scheme; and finally, the corresponding simulation experiment was implemented to verify algorithm effectiveness.

Keywords MapReduce parallel calculation · Hierarchical clustering · Large-scale · Stream data · Skyline query

1 Introduction

Because of the phenomenon—"spatial data explosion but knowledge insufficiency," it is increasingly important to adopt SDMKD (spatial data mining and knowledge discovery) to find the unknown potential useful spatial pattern from the spatial database. Various query methods in the spatial database have provided strong support for spatial

✉ Yifu Zeng
zengfuzi9@163.com

1 College of Electrical and Information Engineering, Hunan University, Changsha 410082, Hunan Province, China

2 College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, Hunan Province, China

Published online: 11 August 2018

data analysis, spatial data discovery, etc. [1–3]. Specifically, the spatial database query includes point query, window query, zone query, nearest neighbor query, clustering query and spatial skyline query. Among above spatial queries, the spatial skyline query as a query preferred by the users is greatly widely applied.

In fact, skyline query is widely applied in such fields as market analysis, decision making, data mining, knowledge discovery, data retrieval and econometrics. For example, the financial business is required to collect massive data, analyze these data, and discover the pattern and characteristics thereof in order to find the financial and commercial interests of individuals, consumer groups or organizations and predict the change tendency of the whole market. Such knowledge discovery process is exactly included in the application scope of skyline query for account analysis and credit evaluation in order to ensure the maximum profits and the minimum risk. In recent years, skyline query is further developed to inverse skyline query, *k* dominant skyline query, probability skyline query, top-*k* skyline query, spatial skyline query, etc.

Skyline query aims to search the data set not dominated by any other data object, and "domain" here can be understood as "superior." Traditional skyline query is mainly considered with non-spatial attribute. For example, if several persons at different positions of a city intend to dine together, then it is necessary for them to consider the price, the risk evaluation, etc., of the restaurant when they choose the restaurant, wherein above factors are the non-spatial attributes. However, in some cases, the most important thing is to consider the spatial attributes, and such requirement cannot be satisfied by traditional skyline query, so the spatial skyline query is needed. Specifically, the spatial skyline query is one of the important queries in the spatial database. For the spatial skyline query, Voronoi graph and *R* tree are combined in the literature [4] for data processing. The DSS (direction-based spatial skyline) query is researched in the literature [5]. A method for calculating common spatial skyline query is proposed in the literature [6], wherein ANN (all nearest neighbor) method is adopted to calculate the object-oriented query, and the INN (incremental nearest neighbors) method is adopted to calculate the facility-oriented query. Above queries are implemented in ideal European space, but some geographical condition constraints inevitably appear in practical application. For example, when we determine the distance between two cities, if a unbridgeable mountain exists between the two cities, the straight line between two points cannot be used for calculating the distance in order to avoid the significant error caused thereby, and we need to determine the shortest distance bypass the mountain, so it is necessary to consider the barrier factor when determining the shortest distance between two objects. For the query in a barrier space, a nearest neighbor query method is proposed in the literature [7] to obtain the actual nearest neighbor of the barrier through the maximum barrier distance query processing method based on line segments and the optimized anonymity zone query processing method. A clustering algorithm based on Voronoi graph in barrier space is proposed in the literature [8–12]. In a barrier space, the existing spatial query research only covers scope query, nearest neighbor query, clustering query, etc., without including any skyline query problem of the barrier space. In fact, the spatial skyline query problem of the barrier space is a new problem to be urgently solved [13].

In order to solve the spatial skyline query problem of the barrier space, based on the MapReduce hierarchical clustering, SOS query method (kdStreamSky) was proposed in the paper. The method can be used for solving the practical problems in knowledge discovery, including line failure troubleshooting for communication media, and the barrier processing during geographic data analysis in the field of national defense and military. Specifically, SOS query method includes two processes: filtering process and refining process [14, 15].

2 Relevant concepts

$A_1, A_2, \ldots, A_d$ are set as d totally ordered sets, with the ordering relation of $\leq$; $S = A_1 \times A_2 \times \cdots \times A_d$ is set as a d-dimensional data space, and $A_1, A_2, \ldots, A_d$ are called as the attributes or dimensionalities of space S; a set including n data points is set as $D = \{p_1, p_2, \ldots, p_n\}$, and an optional data point p_i is sourced from space S, and the value of p_i upon the attribute A_i is recorded as $p_i \cdot a_i$.

Definition 1 (*dominant*) For two optionally set data points $p, q \in D$: If $\forall A_i \in A$, $p \cdot a_i \leq q \cdot a_i$ and $\exists A_j \in A$ are true, then $p \cdot a_j \leq q \cdot a_j$ is true, and P dominates q, recorded as $p{<}q$; if p is not dominated by q, then it is recorded as $p! < q$.

Definition 2 (*skyline*) The set composed of the data points not dominated by any other data point in Set D is called as the skyline set in D; if SK(D) is set to represent the skyline set, then $\mathrm{SK}(D) = \{p \in D | \forall q \in D : q! < p\}$ is true.

Definition 3 (*effective data point*) The time stamp for the arrival of data point p at the system is recorded as $p.t_{arr}$; $p \cdot t_{\mathrm{arr}}$ is set as an integer gradually increased from 0; the size of the time window is set as W, and the present time stamp of the system is t_n; and when $p \cdot t_{\mathrm{arr}} \in [t_n - W, t_n]$ is true, p is regarded as an effective data point; or else, it is regarded as an overdue data point.

Definition 4 (*active data point*) If the effective data point p can meet the condition $p \cdot t_{\mathrm{arr}} \geq q \cdot t_{\mathrm{arr}}$, wherein $q \in \{q | q \in D, q < p\}$ is true, then p is called as an active data point; particularly, all data points in the skyline set are active data points.

Definition 5 (*influence time*) An optional active data point is set as $p \in D$, and the overdue time of the data point with maximum time stamp in the active data point set $\{q | q \in D, q < p\}$ is called as the influence time of F; particularly, if data point meets the condition $p \in$ skyline, then the influence time of p is namely the overdue time thereof.

3 Selection of MapReduce parallel *k*-mean sampling point

3.1 Parallel sampling point selection algorithm framework

In the big data analysis for the large-scale mobile social network, the following two problems are mainly concerned: (1) how to reduce the input and output data volumes

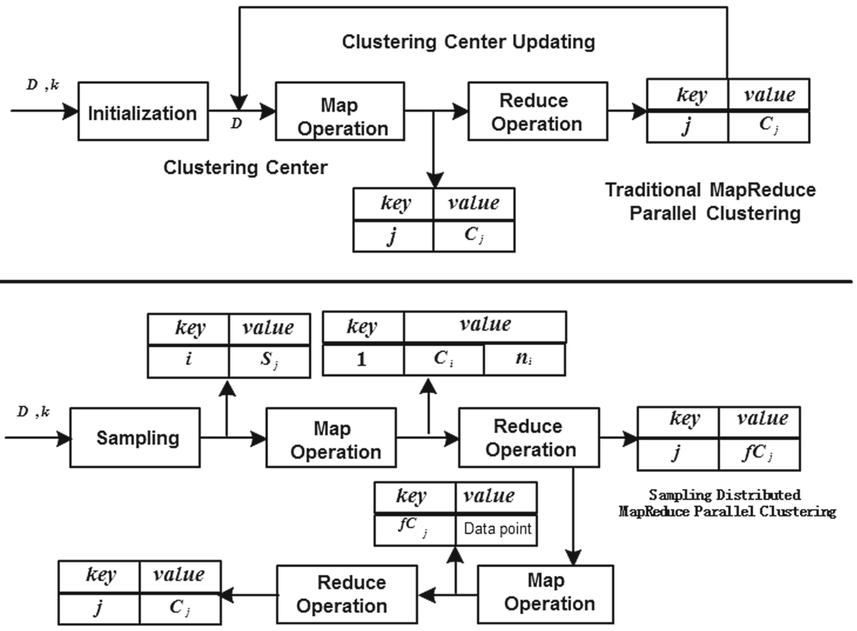

Fig. 1 Algorithm framework

of the algorithm, namely the dimensionality reduction problem; (2) how to maximally reduce the network cost for the inter-node information processing. In allusion to above two problems, based on the sampling thought, MapReduce parallel calculation method is adopted, as shown in Fig. 1.

In the traditional MapReduce parallel clustering process shown in the upper part of Fig. 1, MapReduce operation needs to repeatedly read the massive data stored in the magnetic disc to update the clustering. In this way, the algorithm needs to consume a lot of resources for calculation [16–18].

In order to solve this problem, the big data are sampled in the paper to obtain the subsets, and these subsets are processed to obtain the clustering center sets for clustering the original data set; then, the hierarchical clustering algorithm is adopted for clustering analysis. The distributed MapReduce parallel clustering algorithm framework is shown in the lower part of Fig. 1, and this process includes two-stage MapReduce operation [19–21].

The symbolic variables included in the figure and in the following paragraph are described as follows: D is the original data obtained after preliminary information exchange, $D = (N_i, H_i, \Gamma_i, S_i)$; k is the clustering category number; i and j are integers, $i = 1 \sim 2k$ and $j = 1 \sim k$; c_i is the center of the ith cluster; and C_i is the set of the clustering center c_i.

3.2 *k*-mean MapReduce probability sampling

k-mean algorithm is a common clustering method, and it is used in the paper for data point sampling. In a d-dimensional data space $\mathbb{R}^d$, for the given n-dimensional data $X = \{x_1, x_2, \ldots, x_n\}$, the clustering number of the algorithm is set as $k(1 < k < n)$, and a k-dimensional vector $\zeta = \{m_1, m_2, \ldots, m_n\}$ is set to represent k clustering centers, and k-mean clustering objective is to minimize the in-cluster variance, and the clustering centers are combined to allocate the data to k disjoint sets $\Re = \{\Re_1, \Re_2, \ldots, \Re_k\}$, namely: the two conditions—$\cup_{k=1}^{K} \Re_k = X$ and $\Re_i \cap \Re_j = \varnothing (i \neq j)$—are met. The center of the kth cluster is [4]:

$$m_k = \left(\frac{1}{n_k}\right) \sum_{i=1}^{n_k} x_i^{(k)} \tag{1}$$

In Formula (1), n_k is the data volume of the kth category, $n_k = |\Re_k|$; $x_i^{(k)}$ is the data point in the kth category, $x_i^{(k)} \in \Re_k$. The error can be defined by the quadratic sum of the norm $\|\cdot\|$. For example, the input and the Euclidean distance of the nearest gravity are adopted for description. Then, k-mean clustering objective function is as follows:

$$E(M) = \sum_{i=1}^{n} \min_{k=1\sim k} \|x_i - m_k\|^2 = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_i^{(k)} - m_k \right\|^2 \tag{2}$$

Formula (2) is the common clustering objective, and includes the clustering separability and the measurement homogeneity, thus objectively presenting the clustering effect [22].

The first task of MapReduce operation is the sample selection, and samples are taken from the original data $D = (N_i, H_i, \Gamma_i, S_i)$ on the basis of integer k and probability value $p_x = 1\big/\varepsilon^2 N$, wherein $\varepsilon \in (0, 1)$ is adopted to control the sampling scale. N is the quantity of the data points, the data points sufficient for the full-load operation in a host machine are taken as the samples, and $\varepsilon = 0.005$ is set in the experiment. The probability sampling process is as shown in pseudo-code 1.

When the objective number is set as n, k is empirically set as $n \approx 2k^2$. Therefore, $2k$ small-scale samples are firstly generated; then, k is adopted to cluster each sample to obtain $2k^2$ center points, and these center points are further adopted to generate the final k center points. Notably, if k is an excessively large value, $2k^2$ points cannot be processed in one host machine, so it is necessary to properly reduce the sample selection quantity.

Pseudo-code 1: Probability sampling and sample combination

```
1. Input: $k$ , $D$ ;
2. Output: $c_i$
3. Map( $k_1, v_1$ )
4. for each mapper, for $i$=1:2$k$ do
5. Select the sampling points in $D$ according to probability value $p_x = 1/\varepsilon^2 N$ ;
6. Output ($i$, $x$)
7. Reduce( $k_2, v_2$ )
8. for each $v \in value$ do
9.    $C_i.add(v)$ ;
10. Output $(k_i, C_i)$
11. Map( $k_1, v_1$ )
12. Adopt $k$ -mean algorithm for sampling clustering;
13. Output $(1,(C_i, N_i, P_i))$ ;
14. Reduce( $k_2, v_2$ )
15. Combine sequence $(C_i, N_i, P_i)$ (please refer to pseudo-code 3);
16. Output $(c_1, \cdots, c_{2k^2})$
```

In the figure, $(c_1, \ldots, c_{2k^2})$ are the probability sampling points running in one machine.

4 Experimental analysis

In order to ensure the accuracy of the calculation result, the synchronous response mode is adopted for the algorithm discussed in the paper, namely: After each query submission, the system needs to firstly completely process the data points in the present time window in the cache and then responds to the query request and finally returns the calculation result. In fact, the updating time of the new arriving data points can directly influence the query response time, and the response time is also an important index for reflecting the real-time system performance. Therefore, the query response time is adopted as the standard for measuring the practical performance of the algorithm in the paper.

The experiment environment is as follows: Intel Core i5 2.8-GHz processor, 2-GB internal memory, 260 GB disk space, and C entOS6.2 operating system. The three algorithms—Eager, StreamSubsky and kdStreamSky—are implemented in above experimental environment, and the experimental determination results are compared and analyzed. The experimental data include the standard data and the actual traffic data. The standard data generating tool mentioned in the literature [5] is adopted to generate the standard data; the actual traffic data are sourced from Beijing practical road network and totally include 433,391 entries of road segment information and 171,504 top points; 30,000 fixed objects are randomly generated in the road network, and a mobile object is set to move along the road network to record the fixed objects within the radius range of 500 m and the distance thereto as well as the 3D non-spatial

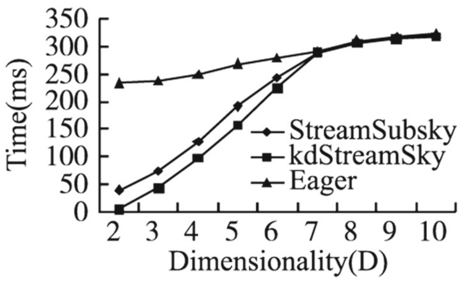

Fig. 2 Query response time of three algorithms under different data dimensionalities

attribute values randomly generated thereby, thus forming a four-dimensional pointset data stream. The quantity of the fixed objects within the radium range of the mobile object is changed along with the movement of the mobile object, so the corresponding data stream presents time-dependent distribution density.

In the experiment, the standard data stream rate is constant; in practical traffic data stream, the movement speed of the mobile object is constant; under the same query response time, the mean value of 10 measurement results is taken; and the performance of StreamSubsky algorithm is related to the mesh width, so the mesh width is set in the experiment according to the optimization rule in the literature [5]. As for the standard data, under different data dimensionalities, data scales and data stream rates, the response time of the three algorithms is measured and determined.

Figure 2 shows the query response time of the three algorithms under different data dimensionalities, wherein the data scale is 300 K, and the data stream rate is 300 points/s. Due to the adoption of efficient index structure, compared with Eager algorithm, StreamSubsky algorithm and kdStreamSky algorithm have obvious performance advantages under different data dimensionalities; further compared with StreamSubsky algorithm, kdStreamSky algorithm has better performance under low dimensionality; along with the increase in data dimensionality, the performance of the three algorithms is gradually approximate and stable, because under constant data scale, the index structures of the three algorithms are gradually failed when the data dimensionality is increased to a certain value and the dominant detection time is approximate to the point-by-point detection time.

Figure 3 shows the query response time of the three algorithms under different data scales, wherein the data dimensionality is set as $d = 4$, and the data stream rate is set as 300 points/s. Among the three algorithms, compared with Eager algorithm, StreamSubsky algorithm and kdStreamSky algorithm have significant performance advantages under different data scales; further compared with StreamSubsky algorithm, kdStreamSky algorithm needs to take more time to maintain the clustering algorithm index under small data scale and accordingly presents slightly poor performance; but along with the increase in the data scale, the increment in the query response time becomes relatively mild and presents obvious advantages.

For the practical traffic data, under the condition of keeping the mobile object at the speed of 5 m/s, the query operation is implemented every 5 s to measure and determine the query response time of the three algorithms; meanwhile, the average query response time of the three algorithm is measured and determined under different moving speeds of the mobile object.

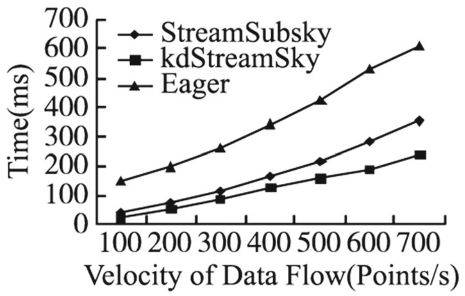

Fig. 3 Query response time of three algorithms under different data stream rates

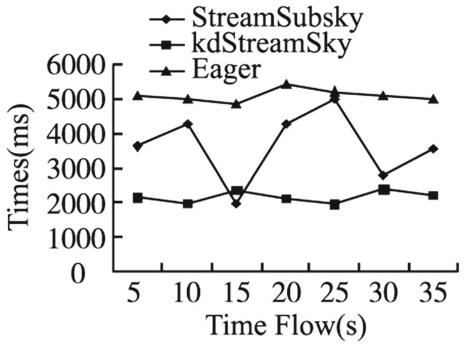

Fig. 4 Query response time of three algorithms under constant movement of mobile object

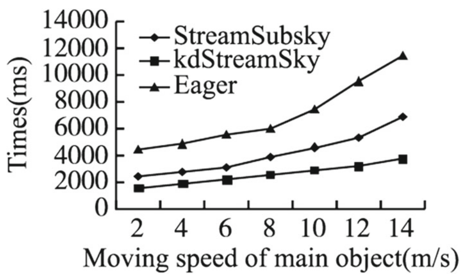

Fig. 5 Mean query response time of three algorithms under different movement speed of mobile object

Figure 4 shows the query response time of the three algorithms under the condition of keeping the mobile object at a constant speed. Among the three algorithms, compared with other two algorithms, kdStreamSky algorithm has better query response time; StreamSubsky algorithm has significant response time fluctuation, because the index of StreamSubsky algorithm has fixed mesh width and the query performance thereof is significantly influenced by the data point distribution density change; in comparison, kdStreamSky algorithm presents good self-adaption, and the query response time is basically stable.

Figure 5 shows the mean query response time of the three algorithms under different movement speeds of the mobile object. The quantity of the data points in the time window is increased along with the increase in the movement speed of the mobile object, so the query response time of the three algorithms is increased along with the increase in the movement speed of the mobile object. Compared with other two algorithms, kdStreamSky algorithm has significant advantages.

In conclusion, compared with StreamSubsky algorithm and Eager algorithm, kdStreamSky algorithm has better performance in standard data environment and has good self-adaption and obvious performance advantages in practical traffic data environment, so it is more suitable for the skyline query task of the traffic data stream with significant distribution density change.

5 Conclusion

A data stream skyline incremental updating algorithm based on clustering algorithm, namely, kdStreamSky algorithm, is proposed in the paper. The algorithm aims to pre-calculate the state transition of the incremental data through event chain mechanism and accordingly update the state of the data point through event processing. Meanwhile, a new index structure clustering algorithm is adopted to index the data points, there is no need to designate any initial parameter, and the index can be adaptively adjusted according to the data point distribution in the data stream. Furthermore, multiple pruning rules are proposed structurally for the clustering algorithm to narrow the search field and improve the search efficiency. The algorithm can quickly respond to the query requests of the users and return the skyline point set of the data stream in any time window to the users. For large-scale and high-speed data stream, kdStreamSky algorithm has good performance and can better adapt to the distribution density change for data stream analysis, so it is suitable for the data stream skyline query task in the intelligent traffic system.

Compared with other similar algorithms based on index structure, kdStreamSky algorithm may have index failure when processing high-dimensional data, so this algorithm has limited uniprocessing capacity and is inevitably required to be parallelized. Meanwhile, the data stream in the intelligent traffic system is usually the distributed data stream, so the next work will be focused on expanding the algorithm to the distributed data stream environment for the parallel expansion of the algorithm in order to adapt to the real-time analysis requirement of the data stream in the intelligent traffic system.

Acknowledgements National Natural Science Foundation of China (Grant No. 61472126).

References

1. Turcan S, Rohle D, Goenka A et al (2016) IDH1 mutation is sufficient to establish the glioma hypermethylator phenotype. Nature 483(7390):479–483
2. Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Discov 29(3):626–688
3. Doulkeridis C, Nørvåg K (2014) A survey of large-scale analytical query processing in MapReduce. VLDB J 23(3):355–380
4. Mohammed MA, Ghani MKA, Arunkumar N, Hamed RI, Mostafa SA, Abdullah MK, Burhanuddin MA (2018) Decision support system for nasopharyngeal carcinoma discrimination from endoscopic images using artificial neural network. J Supercomput. https://doi.org/10.1007/s11227-018-2495-2
5. Mohammed MA, Ghani MKA, Arunkumar N, Hamed RI, Abdullah MK, Burhanuddin MA (2018) A real time computer aided object detection of nasopharyngeal carcinoma using genetic algorithm and

artificial neural network based on Haar feature fear. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2018.07.022

6. Al-Bashir A, Al-Abed M, Amari H, Al-Rousan F, Bashmaf O, Abdulhay E, Al Abdi R, Arunkumar N, Tapas Bapu BR, Al-Basheer A (2018) Computer-based Cobb angle measurement using deflection points in adolescence idiopathic scoliosis from radiographic images. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3614-y
7. Khanna A, Jain S, Aggarwal T, Arunkumar N, Gupta D, Julka A, Albuquerque V (2018) Optimized cuttlefish algorithm for diagnosis of Parkinson's disease. Cogn Syst Res 52:36–48
8. Hussein AF, Arunkumar N, Ramirez-Gonzalez G, Abdulhay E, Tavares JMR, de Albuquerque VHC (2018) A medical records managing and securing blockchain based system supported by a genetic algorithm and discrete wavelet transform. Cogn Syst Res 52:1–11. https://doi.org/10.1016/j.cogsys.2018.05.004
9. Wei J, Meng F, Arunkumar N (2018) A personalized authoritative user-based recommendation for social tagging. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2018.03.048
10. Ashokkumar P, Arunkumar N, Don S (2018) Intelligent optimal route recommendation among heterogeneous objects with keywords. Comput Electr Eng 68:526–535
11. Hussein AF, Arunkumar N, Burbano-Fernandez M, Ramirez-Gonzalez G, Abdulhay E, de Albuquerque VHC (2018) An automated remote cloud-based heart rate variability monitoring system. IEEE Access. https://doi.org/10.1109/access.2018.2831209
12. Sarvaghad-Moghaddam M, Orouji AA, Ramezani Z, Elhoseny M, Farouk A, Arunkumar N (2018) Modelling the spice parameters of SOI MOSFET using a combinational algorithm. Clust Comput. https://doi.org/10.1007/s10586-018-2289-6
13. Elhoseny M, Ramírez-González G, Abu-Elnasr OM, Shawkat SA, Arunkumar N, Farouk A (2018) Secure medical data transmission model for IoT-based healthcare systems. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2817615
14. Vardhana M, Arunkumar N, Abdulhay E, Ramirez-Gonzalez G (2018) Convolutional neural network for bio-medical image segmentation with hardware acceleration. Cogn Syst Res 50:10–14
15. Tharwat A, Elhoseny M, Hassanien AE, Gabel T, Arunkumar N (2018) Intelligent Bézier curve-based path planning model using chaotic particle swarm optimization algorithm. Clust Comput. https://doi.org/10.1007/s10586-018-2360-3
16. El-Dahshan ESA, Mohsen HM, Revett K et al (2014) Computer-aided diagnosis of human brain tumor through MRI: a survey and a new algorithm. Expert Syst Appl 41(11):5526–5545
17. Dobre C, Xhafa F (2014) Intelligent services for big data science. Future Gener Comput Syst 37(2):267–281
18. Gedik B, Schneider S, Hirzel M et al (2014) Elastic scaling for data stream processing. IEEE Trans Parallel Distrib Syst 25(6):1447–1463
19. Sujay RN, Deka PC (2014) Support vector machine applications in the field of hydrology: a review. Appl Soft Comput J 19(6):372–386
20. Lee J, Turner Y, Popa L et al (2014) Application-driven bandwidth guarantees in datacenters. In: ACM Conference on SIGCOMM. ACM, pp 467–478
21. Moshref M, Yu M, Govindan R et al (2014) DREAM: dynamic resource allocation for software-defined measurement. In: ACM Conference on SIGCOMM. ACM, pp 419–430
22. Han J, Zhou P, Zhang D et al (2014) Efficient, simultaneous detection of multi-class geospatial targets based on visual saliency modeling and discriminative learning of sparse coding. ISPRS J Photogramm Remote Sens 89(1):37–48