Available online at www.ijpe-online.com

International Journal of Performability Engineering

Vol. 13, No. 4, July 2017, pp. 540-550
DOI: 10.23940/ijpe.17.04.p21.540550

LAPDK: A Novel Dynamic-programming-based Algorithm for the LAP-(D, k) Query Problem in Wireless Sensor Networks

Xingpo Ma[a,*], Yanli Li[a], Ran Li[a], Yin Li[a], Junbin Liang[b]

[a]*School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, P.R.China*
[b]*School of Computer and Electronics Information, Guangxi University, Nanning 530004, P.R.China*

Abstract

In wireless sensor networks, the *LAP*-(*D, k*) query problem can be seen as a special Top-*k* query problem with the constraint that the Euclidean distance between any two locations corresponding to the data items in the Top-*k* query results should not be smaller than the given value *D*. To solve this problem, a novel dynamic-programming-based heuristic algorithm named LAPDK is proposed. LAPDK firstly divides the sensing field into hexagonal cells using some geometry methods. Then, it finds the approximate solution of the *LAP*-(*D, k*) query problem based on parts of the data generated by the sensor nodes in some preferential cells using the dynamic programming technique. Finally, it further optimizes the solution based on the sensing data received by the Sink node. Simulation results show that LAPDK not only decreases the energy cost of WSNs but also obtains a better approximation ratio compared to the existing state-of-the-art scheme for the *LAP*-(*D, k*) query problem.

Keywords: Wireless Sensor Networks; Distance-constrained Top-k query; Dynamic programming; Hexagonal region partition; Approximation ratio optimization

(Submitted on February 24, 2017; Revised on April 26, 2017; Accepted on May 30, 2017)

1. Introduction

In WSNs (Wireless Sensor Networks) [1], the *LAP*-(*D, k*) query problem [2], which is also known as the Location Aware Peak Value Query problem, can be seen as a special Top-*k* query problem, because the generation location of any sensing data (or event data) in the data set of the Top-*k* query results should also be included in the same data set, Moreover, the Euclidean distance between any two locations in the data set should not be smaller than the given value *D*.

In a sense, it is much more meaningful to solve the *LAP*-(*D, k*) query problem than the traditional Top-*k* query problem. In traditional Top-*k* queries, the generation locations of the data are not considered. In fact, sensor nodes that are located near one another generate similar sensing data [3]. Thus, the sensing data in the traditional query results may be generated in a small local region, and the network users cannot know the situation of the sensor network field as a whole if the parameter *k* is not big enough. Figure 1 shows the sensor network field with an application of soil pollution monitoring. In such an application, if the traditional Top-*k* query processing methods are used, the sensing data in the query results must all come from region A when $k < 16$. Thus, the other three polluted regions (region B, region C, and region D) may be neglected. Although this problem can be overcome by increasing the value of the parameter *k*, the energy consumption of the sensor network must be increased greatly. However, if we require that the Euclidean distance between any couple of locations

* Corresponding author.
E-mail address: maxingpo@xynu.edu.cn.

corresponding two sensing data items in the query results respectively must be larger than the radius of region A, then a top-3 query may be enough to see the whole pollution situation of the soil in the monitored field.

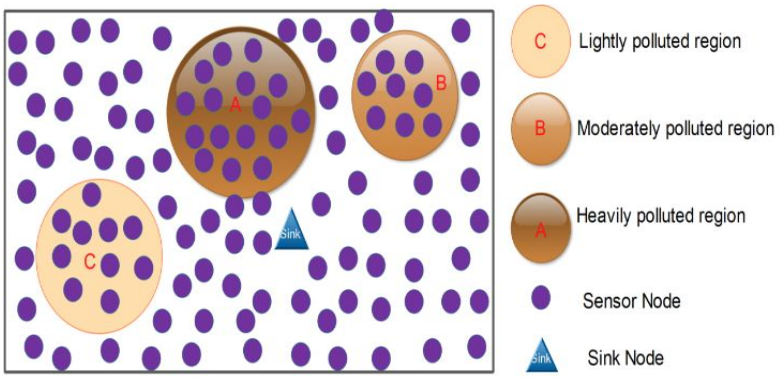

Figure 1. An application of WSNs on soil pollution monitoring

Although solving the *LAP*-(*D*, *k*) query problem is meaningful, it is not easy to do. In fact, it has been proven to be NP-hard in [2]. Although some solutions have been proposed in [2], there is still a great deal to improve the solutions. This paper tries to find a better way to solve the *LAP*-(*D*, *k*) query problem, and the two main contributions are shown as follows:

- A novel dynamic-programming-based Algorithm named LAPDK is proposed for the LAP-(D, k) query problem in WSNs. Many-times optimization is utilized in the algorithm to improve its performance.
- Extensive simulations are done in several scenes to illustrate the better performance of the proposed algorithm on energy efficiency and approximation ratio bound [2] in comparison to the state-of-the-art algorithm.

The rest of this paper is arranged as follows. Section 2 introduces the related work; Section 3 describes some notations and definitions used in this paper; Section 4 presents the novel dynamic-programming-based algorithm LAPDK; Section 5 shows the simulation results with detailed data analysis; Section 6 concludes this paper.

2. Related Work

In the past few years, a lot of schemes and algorithms trying to solve the Top-*k* query problem in WSNs have been proposed [4-16]. Among them, different techniques, such as clustering-based routing [4], random and directed walk [5], and filter-based optimization [6] are used to improve the accuracy [7] and the efficiency [8-11] of different kinds of Top-*k* queries, such as continuous multi-dimensional Top-*k* Queries [8] and the probabilistic Top-*k* Queries [12], or decrease the delay [13] of the Top-*k* query result delivery in different kinds of WSNs [14, 15]. However, the study on the *LAP*-(*D*, *k*) query problem is just at the beginning. Although there are lots of papers that focus on the Top-*k* query problem, most of them do not consider the location relationship among the data items in the Top-*k* query results.

As far as we know, there is only Reference [2] where the *LAP*-(*D*, *k*) query problem was first defined. In [2], the *LAP*-(*D*, *k*) query problem is proven to be NP-hard, and two distributed approximation algorithms are proposed. One is a greedy algorithm, and the other one is a region-partition-based algorithm that we call "*R-based*" algorithm. The ratio bound of the greedy algorithm is 5.8, while that of the "*R-based*" algorithm is 3. In the "*R-based*" algorithm, the whole sensor field is divided into many regular hexagon cells, and the cells are coloured with red, yellow and blue. Then, three query-result sets, each of which is the optimal query-result set of the *LAP*-(*D*, *k*) query problem for the cells with the same colour, are worked out, and the set with the largest sum value will be the final approximation solution. Thus, the "*R-based*" algorithm can achieve the best query result only if the elements in the optimal solution are all from the cells that colour the same. In theory, the probability that such a case happens is about 3/7. In other words, there is still great space to improve the approximation solution of the *LAP*-(*D*, *k*) query problem. Meanwhile, the energy efficiency of the *LAP*-(*D*, *k*) query algorithm can also be improved to maximize the network life [16] of WSNs.

3. Notations and Definitions

We assume N sensor nodes denoted by $\{S_1, S_2, S_3, \ldots, S_N\}$ are deployed randomly in a square field. The sensor nodes are static, and each of them can obtain its location using some localization algorithms that are designed for WSNs. Let l_i and v_{it} denote the location of node S_i ($1 \le i \le N$) and the sensing data generated by S_i at time t respectively. Then, l_i and v_{it} make up of a

tuple which is denoted as $e_i = (l_i, v_{it})$, and all tuples generated by all sensor nodes at time t can be denoted as $V_t=\{(l_1, v_{1t}), (l_2, v_{2t}),\ldots,(l_N, v_{Nt})\}=\{ e_1, e_2, e_3,\ldots, e_N\}$. For ease of presentation, we use v_i and V to take the place of v_{it} and V_t respectively in the following of this paper. For any couple of tuples e_i and e_j ($1\le i\le N$, $1\le j\le N$, $i \ne j$), the distance between them is denoted as $dis(e_i, e_j)$, which is the Euclidean distance between l_i and l_j.

Definition 1. *D-separated* and *D-closed*: Given any $D > 0$, any couple of tuples e_i and e_j ($1\le i\le N$, $1\le j\le N$, $i \ne j$) are *D-separated* if $dis(e_i, e_j) > D$, and are *D-closed* if $dis(e_i, e_j) \le D$.

Definition 2. *D-separated subset*: Given any $D > 0$ and any set U, U is the *D-separated subset* of set V if $U \subseteq V$ and any couple of tuples in U are *D-separated.*

Definition 3. *Weight of subset*: For any subset $U \subseteq V$, the weight of U is the sum of all the sensing data in all tuples in U. Let $w(*)$ denote the function of obtaining the weight of U. Then, we have

$$w(U)= \sum_{(l_i,v_i)\in U} v_i \tag{1}$$

Based on the definitions above, the *LAP*-(*D*, *k*) query problem can be described specifically as follows. Given set $V=\{(l_1, v_1), (l_2, v_2),\ldots,(l_n, v_n)\}$and another two parameters k and D， the *LAP*-(*D*, *k*) query problem is how to find the *D-separated subset* W of set V in the condition that the total number of elements in W is not more than k and that the weight of W is the biggest among the weights of all V's *D-separated subsets*, each of which contains no more than k elements.
In the following, we present a heuristic algorithm named LAPDK to solve the above-mentioned *LAP*-(*D*, *k*) query problem.

4. The LAP(D, k) Query Processing Algorithm LAPDK

4.1. The Main Frame of LAPDK

The main idea of LAPDK is to find the optimal *LAP*-(*D*, *k*) query results in local regions and then compute the approximation solution of the *LAP*-(*D*, *k*)query problem based on the selected optimal *LAP*-(*D*, *k*) query results in some regions using the dynamic-programming-based method.

The content of LAPDK is descripted in Algorithm 1. Generally speaking, Algorithm 1 can be mainly divided into four steps: (1) Segment the whole sensor field into regular hexagon cells; (2) Compute the optimal *LAP*-(*D*, *k*) query results in each cell; (3) Compute the first-stage approximation solution for the *LAP*-(*D*, *k*) query problem in the whole region based on the collected optimal *LAP*-(*D*, *k*) query results in some regions using the dynamic- programming-based method; (4) Optimize the first-stage approximation solution to get the final approximation solution. In Algorithm1, the optimal *LAP*-(*D*, *k*) query solution in each cell can be worked out by the exhaustive method, because the biggest element number of a *D-separated set* in a cell is 5 according to [2].

Algorithm 1. The *LAP*-(*D*, *k*) Query Processing Algorithm (LAPDK)

Input： Parameters k and D
Output： *AS*

/**AS* denotes an approximation solution of the *LAP*-(*D*, *k*) query problem */
1. Divide the whole sensor field into many regular hexagon cells with side length D;
2. Choose one sensor node as the head node in each cell dynamically according to the rest of the energy of the sensor nodes;
3. The head node in each cell collects the data tuples generated by the sensor nodes in its own cell;
4. Using the collected data tuples, the head node in each cell computes the optimal *LAP*-(*D*, 1) query result in its own cell, and sends the result to the Sink node;
5. After receiving all the optimal *LAP*-(*D*, 1) query results from all the head nodes, the Sink node first finds out the top k cells which have the largest weights of the *LAP*-(*D*, 1) query results, and then sends data-requesting messages to the selected cells;
6. Suppose the IDs of the selected cells are $x_1, x_2, x_3,\ldots,x_k$ respectively. Set $CID_{selected} =\{ x_1, x_2, x_3,\ldots,x_k \}$;
   /* $CID_{selected}$ denotes an ID set which includes the IDs of all the selected cells */
7. If receives a data-requesting message, any head node which is located in one of the selected cells will compute the optimal *LAP*-(*D*, 2) query results, the optimal *LAP*-(*D*, 3) query results, the optimal *LAP*-(*D*, 4) query results, and the optimal *LAP*-(*D*, 5) query results respectively in its own cell, and send them all to the Sink node;
8. Suppose $U_{selected} =\cup_{y\in CID_{selected}} \left\{LAP_y(D, 1)\cup LAP_y(D, 2)\cup\ldots \cup LAP_y(D, 5)\right\}$,Using the dynamic-programming-based method, the Sink node finds out a subset

$SU_{selected} \subseteq U_{selected}$, where $SU_{selected}$ satisfies the following two conditions:
(1) $|SU_{selected}| = k$, and
(2) for any subset $P \subseteq U_{selected}$ where $|P| = k$ and $P \neq SU_{selected}$, there is $w(P) \leq w(SU_{selected})$,
where $LAP_y(D, k)$ ($1 \leq k \leq 5$) denotes the optimal *LAP*-(D, k) query results in the cell with ID y

9. Sink deletes all the *D-closed* tuples in $SU_{selected}$ making $SU_{selected}$ be subset $SSU_{selected}$ using **Function 1** first, and then process $SSU_{selected}$ using **Function 2** to obtain the approximate solution R_{Dk} of the *LAP*(D, k) query problem for the whole sensor network;
10. If (($|R_{Dk}| = k$)||(CID- $CID_{selected} = \emptyset$)){
/* CID denotes a set which includes the IDs of all the cells in the sensor network field. */
11. return R_{Dk};
12. }else{
13. Sink finds out the cell which has the biggest weight of the *LAP*(D, 1) query result among all the cells whose IDs are not included in $CID_{selected}$. Suppose h denote the ID of such a cell. Then, Sink sends a data-requesting message to the cell with ID h;
14. $CID_{selected} = CID_{selected} \cup \{u\}$;
15. Go to line 7;
16. }

4.2. Dynamic Programming in LAPDK

The aim of using dynamicprogramming-based method in LAPDK is to find out a subset $SU_{selected} \subseteq U_{selected}$ so that the following conditions can be satisfied：

- $|SU_{selected}| = k$;
- For any subset $P \subseteq U_{selected}$ where $|P| = k$ and $P \neq SU_{selected}$, there is $w(P) \leq w(SU_{selected})$.

We then describe how the subset $SU_{selected}$ can be worked out using the dynamic-programming-based method as follows.

Let U_j denote the union set of j optimal *LAP*-(D, k) query results, each of which corresponds to one cell. Then we have

$$U_j = \cup_{y=1,2,...,j} \left\{ LAP_y(D, 1) \cup LAP_y(D, 2) \cup ... \cup LAP_y(D, 5) \right\} \tag{2}$$

Meanwhile, let $U_{j,h}$ denote one of U_j's subsets, which satisfies the following two conditions：

- $|U_{j,h}| = h$
- For any other U_j's subset P which satisfies the condition $|P|=h$， there is $w(P) \leq w(U_{j,h})$.

Then, $U_{j,h}$ can be worked out using the following dynamic programming function：

$$U_{j,h} = \begin{cases} \varnothing .....(h = 0 \,||\, j = 0) \\ U_{j\text{-}1,h\text{-}l_0} \cup LAP_j(D, l_0) .....(h > 0 \,\&\&\, j > 0) \end{cases} \tag{3}$$

In (3), $LAP_j(D, l_0)$ ($0 \leq l_0 \leq \min(h,5)$) denotes the optimal *LAP*-(D, l_0) query results in the j^{th} cell， and $LAP_j(D, l_0) = \emptyset$ if $l_0=0$; Moreover, l_0 in (3) must satisfy the following equation:

$$w\left(U_{j\text{-}1,h\text{-}l_0}\right) + w\left(LAP_j(D, l_0)\right) = \underset{l=0}{\overset{\min(h,5)}{MAX}} \left\{ w\left(U_{j\text{-}1,h\text{-}l}\right) + w\left(LAP_j(D, l)\right) \right\} \tag{4}$$

Supposing $j=|CID_{selected}|$ and $h=k$, we can then worked out $SU_{selected}$ using the above-mentioned dynamic-programming function.

4.3. The Contents of Function 1 and Function 2

The outcome of the dynamic-programming computation in line 8 in Algorithm 1 is the set $SU_{selected,}$ which may be a *D-closed set*. Because the expected solution is a *D-separated set*, it is needed to process $SU_{selected}$ to make the elements in it be all *D-separated* with one another. The way to process $SU_{selected}$ in LAPDK is to check every element in $SU_{selected}$ and find out whether there are some other elements which are *D-closed* with it in the same set or not. If there are some, delete all the elements that are *D-closed* with one another in $SU_{selected}$. This progress is shown in **Function 1**.

After being processed using **Function 1**, the number of the elements in $SSU_{selected}$ may be smaller than the given parameter k. However, using the tuples that were collected by the Sink node, we can add some more tuples to $SSU_{selected}$ and/or take place of some tuples in $SSU_{selected}$ so that the approximation ratio of the $SSU_{selected}$ can be improved further. The way to process $SSU_{selected}$ further is shown in **Function 2**.

Function 1. The Function of Processing Subset $SU_{selected}$

Input: Subset $SU_{selected}$
Output: A *D-separated* subset $SSU_{selected}$, where $SSU_{selected} \subseteq SU_{selected}$

1. $SSU_{selected} = \varnothing$;
2. while($SU_{selected} \neq \varnothing$){
3. get a tuple denoted as e_{tu} from $SU_{selected}$;
4. $SU_{selected} = SU_{selected} - \{ e_{tu} \}$;
5. if(there is no tuple which is *D-closed* with e_{tu} in $SU_{selected}$){
6. $SSU_{selected} = SSU_{selected} + \{ e_{tu} \}$;
7. }else{
8. Find out all the tuples which are *D-closed* with e_{tu} in $SU_{selected}$, and then delete them all from $SU_{selected}$;
9. }
10. }
11. Return $SSU_{selected}$;

Function 2**.** The Function of Processing Subset $SSU_{selected}$

Input: k, D, $SSU_{selected}$ and $V_{collected}$
/* $V_{collected}$ denotes a set which includes all the tuples received by the Sink node */
Output: The approximate solution R_{Dk}

1. $V_{collected} = V_{collected} - SSU_{selected}$;
2. while($V_{collected} != \varnothing$){
3. get any tuple denoted as tpl_0 from $V_{collected}$;
4. $V_{nbr} = \varnothing$, $R_1 = SSU_{selected}$;
// V_{nbr} and R_1 are two sets defined temporarily
5. while ($R_1 != \varnothing$){
6. Get any tuple denoted as tpl_1 from R_1;
7. if($dis(tpl_0, tpl_1) \leq D$){
8. $V_{nbr} = V_{nbr} \cup tpl_1$;
9. }
10. $R_1 = R_1 - tpl_1$;
11. }
/* Now, all the tuples which is no more than D far from tpl_0 in R_1 has been collected in V_{nbr} */
12. if ($V_{nbr} = \varnothing$){
13. if($|SSU_{selected}| == k$){
14. find out the tuple $tpl_{smallest}$ with the smallest weight in $SSU_{selected}$;
15. If($w(tpl_{smallest}) < w(tpl_0)$){
16. $SSU_{selected} = (SSU_{selected} - \{ tpl_{smallest} \}) \cup \{ tpl_0 \}$;
17. }
18. }else{
19. $SSU_{selected} = SSU_{selected} \cup \{ tpl_0 \}$;
20. }// end of if($|SSU_{selected}| == k$)
21. }else{
22. if($w(V_{nbr}) < w(tpl_0)$){
23. $SSU_{selected} = SSU_{selected} - V_{nbr}$;
24. $SSU_{selected} = SSU_{selected} \cup \{ tpl_0 \}$;
25. } // end of if($w(V_{nbr}) < w(tpl_0)$)
26. }
27. $V_{collected} = V_{collected} - tpl_0$;
28. } // end of the while circulation
29. return $SSU_{selected}$;

5. Simulations

The simulation tools used in this paper is OMNET++， and the comparative object of the LAPDK algorithm proposed in this paper is the "*R*-based" algorithm, which is proposed in [2]. The default parameter settings are shown in **Table 1**. *SZ* denotes the area size of the WSN field, *RDS* denotes the Communication radius of the sensor nodes, l_d denotes the length in bytes of one data item, l_{ID} denotes the length in bytes of a node's ID, l_{loc} denotes the length in bytes of the location information of a node, ζ_r denotes the consumed energy by receiving a byte of data, and ζ_s denotes the consumed energy by sending a byte of data.

Table 1. Parameter Settings

Parameter	Value	Parameter	Value
N	500	SZ	400×400m^2
k	10	RDS	40 m
D	20 m	l_d	8 bytes
l_{ID}	2 bytes	l_{loc}	16 bytes
ζ_r	0.0057mJ	ζ_s	0.0144mJ

In the following simulations, we assume that the data that are located in the numerical interval [0, 50] are normal， and the data that are located in [51, 100] are abnormal. Moreover, we call the area as the abnormal region if the data generated by all the sensor nodes in the area are all abnormal. To test the performance of the algorithm LAPDK sufficiently, we compare LAPDK with the "*R*-based" algorithm in the following three scenes:

Scene 1: There is no obvious abnormal region in the deployed sensing field, and each sensor node chooses one value from the numerical interval [0, 100] randomly as its sensing value.

Scene 2: There is only one abnormal region in the sensing field. The abnormal region in scene 2 is a round area, which takes the central point of the sensing field as its own center, with radius *L* (a parameter whose value can be adjusted in the simulation). In the abnormal region, each sensor node chooses one data from the numeric interval [51, 100] randomly as its own sensing data. While in the normal region, each sensor node chooses a data from the numeric interval [0, 50] randomly as its own sensing data. Scene 2 is shown in Figure 2.

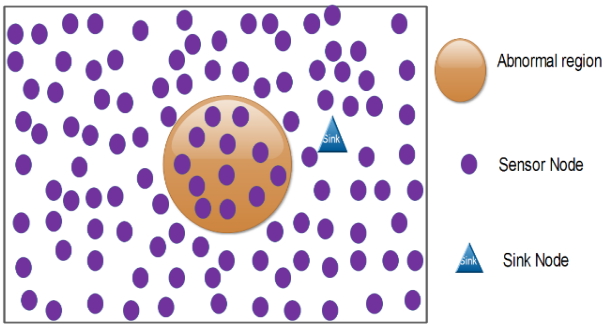

Figure 2. Scene 2

Scene 3: There are a total of four abnormal regions in the sensing field. The four abnormal regions are all square with central locations (200, 100), (200, 300), (100, 200), (300, 200) respectively. The side length of each abnormal region is *L*, which is an adjustable parameter. Moreover, the way for the sensor nodes both in and out of the abnormal regions in Scene 3 to get their own sensing data is the same as that in Scene 2. Scene 3 is shown in Figure 3.

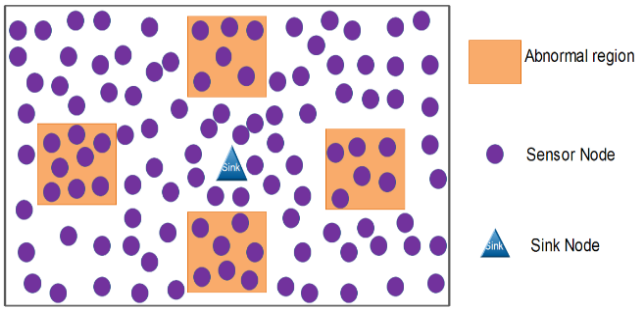

Figure 3. Scene 3

5.1. Simulation results in Scene 1

In Scene 1, there is no abnormal region, and we mainly test the affection of the parameters D and k on the total energy consumption and the approximate ratio of the *LAP*-(D, k) query solutions in both the LAPDK algorithm and the "*R*-based" algorithm. In fact, it is hard to compute the weight of the optimal *LAP*-(D, k) query solution. However, the weights of the optimal *LAP*-(D, k) query solutions for both of the schemes are the same. Thus, we can just test the weights of the *LAP*-(D, k) query solutions obtained according to the LAPDK algorithm and the "*R*-based" algorithm instead of testing the approximate ratios directly. It is clear that, the higher the weight of the *LAP*-(D, k) query solution, the better approximate ratio it achieves.

5.1.1 Simulation results with different values of D

The simulation results of the total energy consumption of all the sensor nodes with different values of D are shown in Figure 4. We can see that, as D increases, the total energy consumption brought by the two algorithms all decrease. The reason is that, the total number of cells in the sensing field decreases as D increases and the quantity of the data that are transmitted to the Sink node decreases. In Figure 4, we can also find that, although the total energy consumption brought by the LAPDK algorithm is a little higher than that brought by the "*R*-based" algorithm when D is small, as D increases, the total energy consumption brought by the LAPDK algorithm is obviously lower than that brought by the "*R*-based" algorithm. This reflects that, as the number of regular hexagon cells increases, the tuples which are transmitted to the Sink node in LAPDK is much fewer than those in the "*R*-based" algorithm.

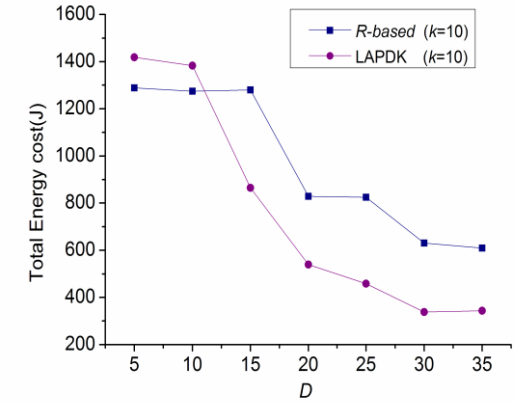

Figure 4. The total energy consumption in Scene 1 with different values of D

The simulation results of the weights of the *LAP*-(D, k) query results in the two algorithms with different D values are shown in Figure 5. It is clear that the weights of the *LAP*-(D, k) query result in LAPDK are higher than those in the "*R*-based" algorithm no matter what value D is set to. It implies that, in Scene 1, the *LAP*-(D, k) query result obtained by LAPDK has a higher approximate ratio than that obtained by the "*R*-based" algorithm no matter what sizes the cells in the sensing field are.

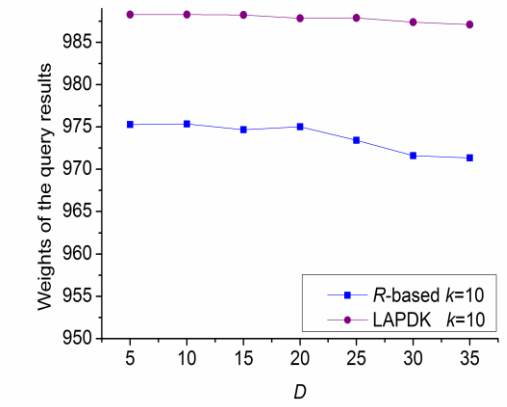

Figure 5. The weights of the LAP(D, k) query result in Scene 1 with different values of D

5.1.2 Simulation results with different values of k

The simulation results of the total energy consumption of all the sensor nodes with different values of *k* are shown in Figure 6. We can see that, as *k* increases, the total energy consumption brought by LAPDK increases slightly, while that brought by the "*R*-based" algorithm increases greatly. The reason is as follow. For the LAPDK algorithm, increasing *k* means only a few more cells need to transmit their own optimal *LAP*-(*D*, *k*) ($1 \leq k \leq 5$) query solutions to the Sink node. Meanwhile, as the parameter *k* increases, the amount of the data transmitted in the aggregation tree used in the "*R*-based" algorithm increases greatly so that the total energy consumption of the whole sensor network increases greatly. Moreover, Figure 6 also shows that the energy efficiency of LAPDK is also much better than that of the "*R*-based" algorithm.

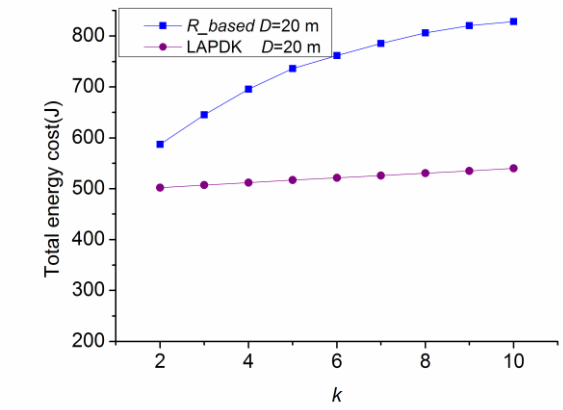

Figure 6. The total energy consumption in Scene 1 with different values of k

Figure 7 shows the simulation results of the weights of the *LAP*-(*D*, *k*) query results using the two algorithms respectively with different values of *k*. From Figure 7, we can conclude that approximate ratio of the *LAP*-(*D*, *k*) query result obtained according to the LAPDK algorithm is a little higher than that obtained according to the "*R*-based" algorithm.

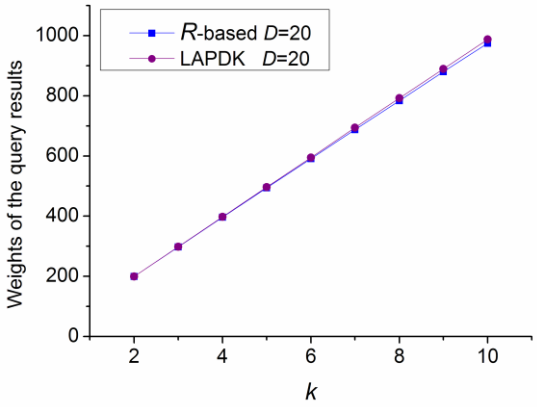

Figure 7. The weights of the LAP(D, k) query result in Scene 1 with different values of k

5.2. Simulation results in Scene 2

The simulation results of the total energy consumption of all the sensor nodes with different sizes of the abnormal region in Scene 2 are shown in Figure 8, where the abscissa denotes the area sizes of the abnormal region in the sensing field. We can see that, the total energy consumption brought by using the LAPDK algorithm is obviously less than that brought by using the "*R*-based" algorithm no matter what size the abnormal region is.

Figure 9 shows the simulation results of the weights of the *LAP*-(*D*, *k*) query results obtained according to the LAPDK algorithm and the "*R*-based" algorithm respectively with different abnormal region sizes. It is clear that, as the increasing of the size of the abnormal region, the weights of the *LAP*-(*D*, *k*) query results in both algorithms all increase. However, when the weights increase to a certain degree, they become relatively steady. This is because, at the beginning, the abnormal region

is small, and the tuple number in the *LAP*-(*D*, *k*) query results is smaller than *k*. As the size of the abnormal region increases, the tuple number in the *LAP*-(*D*, *k*) query results increases too, and each tuple in the *LAP*-(*D*, *k*) query results also have more chances to get a bigger weight. This leads to an increase in weight of the *LAP*-(*D*, *k*) query results at the beginning. When the tuple number in the *LAP*-(*D*, *k*) query result reaches *k* and the average weight of the tuples in the *LAP*-(*D*, *k*) query result becomes steady, the weight of the *LAP*-(*D*, *k*) query result becomes steady too. Whatever, the weights of the *LAP*-(*D*, *k*) query results obtained according to the LAPDK algorithm are all bigger than those obtained according to the "*R*-based" algorithm no matter what size the abnormal region is. This implies that using the LAPDK algorithm can achieve a better approximate ratio than using the "*R*-based" algorithm.

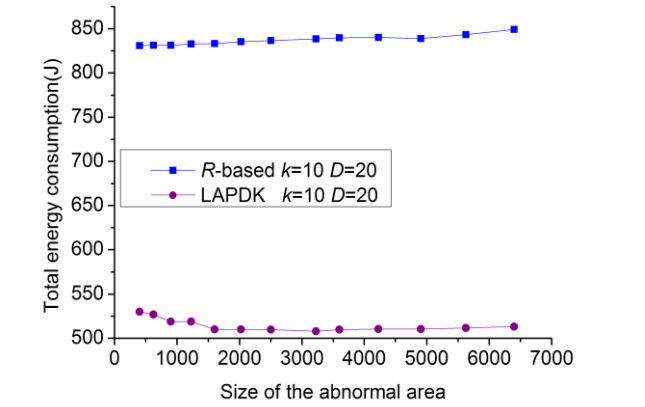

Figure 8. The total energy consumption in Scene 2 with different sizes of the abnormal region

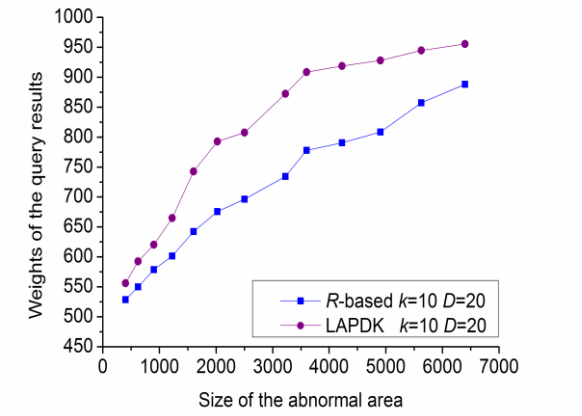

Figure 9. The weights of the LAP(D, k) query results in Scene 2 with different sizes of the abnormal region

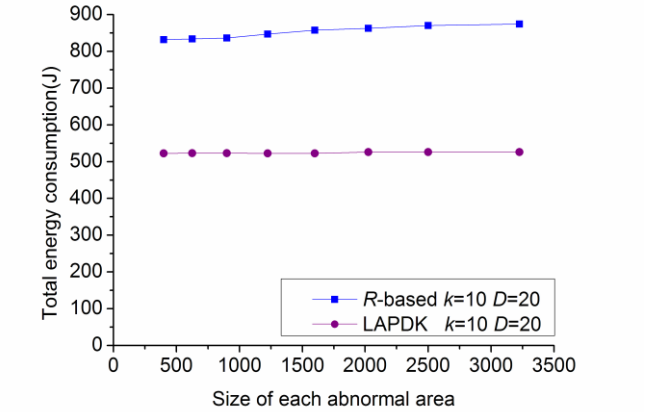

Figure 10. The total energy consumption in Scene 3 with different sizes of the abnormal region

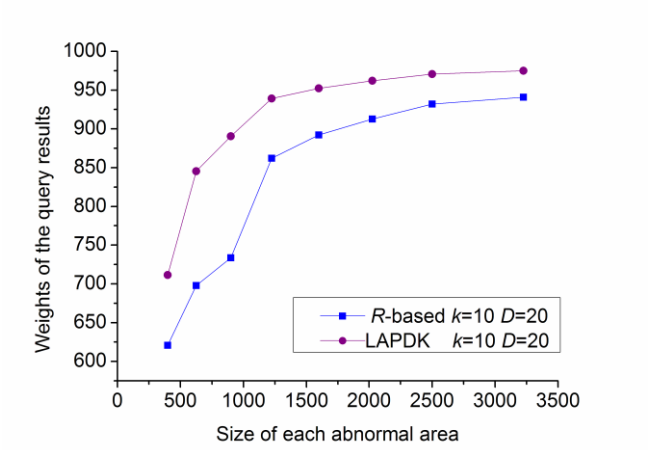

Figure 11. The weights of the LAP(D, k) query results in Scene 3 with different sizes of the abnormal region

5.3. Simulation results in Scene 3

The simulation results of the total network energy consumption and the weights of the *LAP*-(*D*, *k*) query results in Scene 3 obtained according to the LAPDK algorithm and the "*R*-based" algorithm with different sizes of the abnormal regions are shown in Figure 10 and Figure 11 respectively. From these simulation results we can find that the LAPDK algorithm outperforms the "*R*-based" algorithm by both the energy efficiency and the approximate ratio of the *LAP*-(*D*, *k*) query results in Scene 3.

6. Conclusion

The *LAP*-(*D*, *k*) query problem, which is derived from the traditional Top-*k* problem, is studied in this paper. It is a much more meaningful problem to study because the data items in the *LAP*-(*D*, *k*) query results will not be so concentrated in geography distribution and so the users of the network will have a better view of the monitored fields. However, it is not easy to solve such a problem because it has been proven to be NP-hard. We present a novel heuristic algorithm named LAPDK in this paper to solve the problem. In LAPDK, regular hexagon segmentation and dynamic-programming-based techniques are used to improve the approximate ratio of the *LAP*-(*D*, *k*) query result and the energy efficiency of the sensor network. The total network energy consumption is decreased by shrinking the amount of the data tuples to be transmitted to the Sink node without decreasing the approximate ratio of the *LAP*-(*D*, *k*) query result. To test the performance of the proposed algorithm, we performed simulations in three scenes and compared it to the "*R*-based" algorithm, which is the state-of-the-art solution on the *LAP*-(*D*, *k*) query problem. Almost all of the simulation results showed that LAPDK performs much better than the "*R*-based" algorithm on both energy efficiency and the approximate ratio of the *LAP*-(*D*, *k*) query result.

Acknowledgements

This research is supported by NSFC (Natural Science Foundation of P. R. China, No. 61501393, No. 61562005), the Natural Science Foundation of Henan Province of P. R. China (No. 162300410234), the Nanhu Scholars Program for Young Scholars of XYNU, and the supporting program of young backbone teachers in Xinyang Normal University, P. R. China.

References

1. P. Naik, N. Telkar, and K. Kotin, "Survey on Wireless Sensor Network with their remaining Challenges", *International Journal of Scientific Research in Science and Technology*, vol. 2, no. 6, pp.321-331, 2016
2. S. Cheng, J. Li, and L. Yu, "Location Aware Peak Value Queries in Sensor Networks", *Proceedings of IEEE INFOCOM*, pp.486-494, 2012
3. A. Jindal, and K. Psounis, "Modeling spatially correlated data in sensor networks", *ACM Transactions on Sensor Networks*, vol. 2, no.4, pp. 466-499, 2006
4. S. Mo, H. Chen, and Y. Li, "Clustering-based routing for top-k querying in wireless sensor networks", *Eurasip Journal on Wireless Communications & Networking*, issue 1, pp.1-13, 2011
5. J. Fu, and Y. Liu, "Random and Directed Walk-Based Top-k Queries in Wireless Sensor Networks", *Sensors*, vol.15, no.6, pp.12273-12298, 2015
6. H. Haiping H, Y. Qi, Q. Xiaolin, and W. Ruchuan, "A filter-based algorithm for optimizing top-k queries in wireless sensor

networks", *Journal of Systems Architecture*, vol.58, no.2, pp.73-85, 2012
7. Q. Pan, M. Li, M. Wu, and W. Shu, "Optimization of Accurate Top-k Query in Sensor Networks with Cached Data", *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference* (IWCMC) , pp.553 - 558, 2011
8. B. Chen，W. Liang，J. Yu, "Energy-efficient top-k query evaluation and maintenance in wireless sensor networks", *Wireless Networks*, vol.20, no.4, pp.591-610, 2014
9. D. Choi, and C. Chung, "REQUEST+: A framework for efficient processing of region-based queries in sensor networks", *Information Sciences*, vol.248 no.6, pp.151-167, 2013
10. B. Malhotra, M.Nascimento, and I. Nikolaidis, "Exact Top-k Queries in Wireless Sensor Networks," IEEE Transactions on Knowledge and Data Engineering, vol.23, no.10, pp.1513-1525, 2011
11. H. Jiang, J. Cheng, D. Wang, C. Wang, and G. Tan, "Continuous Multi-dimensional Top-k Query Processing in Sensor Networks", *Proceedings of IEEE INFOCOM*, pp.793-801, 2011
12. M. Ye, W. Lee, D. Lee, and X. Liu, "Distributed Processing of Probabilistic Top-k Queries in Wireless Sensor Networks", *IEEE Transactions on Knowledge and Data Engineering*, vol.25, no.1, pp.76-91, 2013
13. S. Tang, X. Mao, and X. Li, "Efficient and fast distributed top-k query protocol in wireless sensor networks", *Proceedings of International Conference on Network Protocols* (ICNP) , pp. 99-108, 2011
14. C. Zhu, L. Yang, L. Shu, and S. Nishio, "Insights of Top-k Query in Duty-Cycled Wireless Sensor Networks", *IEEE Transactions on Industrial Electronics*, vol.62, no.2, pp.1317-1328, 2015
15. H. Wang，Z. Guan，T. Yang，and Y. Xu, "Top-K Query Framework in Wireless Sensor Networks for Smart Grid", *China Communications*, vol.11, no.6, pp.89-98, 2014
16. H. Yetgin, K. Cheung, M. El-Hajjar, and L. Hanzo, "A Survey of Network Lifetime Maximization Techniques", *IEEE Communications Surveys & Tutorials*, 2017, DOI: 10.1109/COMST.2017.2650979