Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 6434083, 15 pages
https://doi.org/10.1155/2018/6434083

Research Article

A Node Selection Paradigm for Crowdsourcing Service Based on Region Feature in Crowd Sensing

Zhenlong Peng,[1,2,3,4] Xiaolin Gui,[1,2,4] Jian An,[2] Dong Liao,[1,4] Ningchao Cai,[4,5] and Ruowei Gui[1,4]

[1]*School of Electronics and Information Engineering, Xi'an Jiaotong University, No. 28, Xianning West Road, Xi'an 710049, China*
[2]*Xi'an Jiaotong University Shenzhen Research School, High-Tech Zone, Shenzhen 518057, China*
[3]*Chen Shouren Business School, Quanzhou Normal University, Donghai Street, Quanzhou 362000, China*
[4]*Shaanxi Province Key Laboratory of Computer Network, No. 28, Xianning West Road, Xi'an 710049, China*
[5]*The Fu Foundation School of Engineering and Applied Science, Columbia University, Manhattan, NY, USA*

Correspondence should be addressed to Jian An; anjian@mail.xjtu.edu.cn

Received 24 May 2018; Revised 9 September 2018; Accepted 28 October 2018; Published 14 November 2018

Academic Editor: Sara Montagna

Crowd sensing is a human-centered sensing model. Through the cooperation of multiple nodes, an entire sensing task is completed. To improve the efficiency of sensing missions, a cost-effective set of service nodes, which is easy to fit in performing different tasks, is needed. In this paper, we propose a low-cost service node selection method based on region features, which builds on the relationship between task requirements and geographical locations. The method uses Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm to cluster service nodes and calculate the center point of each cluster. The area then is divided into regions according to rules of Voronoi diagrams. Local feature vectors are constructed according to the historical records in each divided region. When a particular sensing task arrives, Analytic Hierarchy Process (AHP) is used to match the feature vector of each region to mission requirements to get a certain number of service nodes satisfying the characteristics. To get a lower cost output, a revised Greedy Algorithm is designed to filter the exported service nodes to get the required low-cost service nodes. Experimental results suggest that the proposed method shows promise in improving service node selection accuracy and the timeliness of finishing tasks.

1. Introduction

With a growing number of smart mobile devices and an increasing number of embedded sensors, crowd sensing is becoming an effective way of completing large-scale sensing tasks in a sensing area. Because of mobile Internet technology and the continuous development of sensor technology, people increasingly use mobile intelligent devices in their lives. In this trend, crowd sensing computing is gradually becoming the core of the mobile computing stage [1]. The core idea of mobile crowd sensing is to distribute tasks to unspecified public (service nodes). After completing the tasks, these nodes return the results to the sensing system, which then processes the results (such as removing interference information, integrating information, etc.), and finally get the results we need. Crowd sensing presents a new sensing paradigm based on the capacities of mobile devices and the interactions of a person or a group [2]. It uses mobile devices owned by ordinary users (i.e., mobile phones and tablet PCs) as primary sensing units and then achieves sensing task distribution and sensing data collection through explicit or implicit mobile Internet collaborations. In addition, crowd sensing has become a promising paradigm for cross-space and large-scale sensing [3]. Task requestors can ordinarily use the corresponding crowd sensing platform to distribute the sensing tasks to many mobile nodes instead of allocating many fixed sensors to collect data.

During the crowd sensing process, the selection of service nodes is important to the quality of the crowd sensing system. In general, most of the crowd sensing systems sent tasks to all participants. However, there are several problems: (1) some participants are only trying to get rewards without performing tasks seriously. That is, there are some deceivers; (2) the data of these unqualified participants will increase the difficulty of the subsequent processing of the system. (3) If the unqualified participants cannot be effectively identified, the crowd sensing system will cost more. Therefore, it is very important to choose qualified nodes to participate in crowd sensing to save time and money. However, there remain some drawbacks of existing methods.

First, these methods do not emphasize the importance of relationships between the distribution of sensing nodes and the geographic area division, which can lead to the inaccuracy of the analysis of geographical characteristics. For many current methods, a commonly used approach is called Grid Division, which is used to ensure data integrity and completeness of the collected information of mobile devices. In [4–7], Grid Division is used to process a specific sensing area to conduct further analysis on collected data. By using this method, the entire sensing area is divided into multiple equally sized regions for node selection based on some criteria such as longitude and latitude. However, since some requested tasks might relate to the patterns of population, we cannot recklessly use Grid Division for every task. What is more, even though we have a better method to partition the sensing area into different regions, we should not select nodes in some random regions. Matching one or several regions with a high degree of task correlation, selecting service nodes from these regions and performing tasks on selected service nodes can facilitate the effective and efficient accomplishment of the sensing task. The key to the realization of this idea is to find a good method to partition a specific sensing area and match some divided regions to specific tasks, that is, to construct eigenvectors of each region.

Example 1. A sensing task requires analyzing people's interests and their attention to locations of advertising settings in a specific suburb area. For this area, the density population distribution varies in different parts. So the task requestors should first find out some ways to partition the area reasonably for analysis. And then, they should think about how to efficiently solve the problem of evaluating the regions whether they are good for collecting data for advertising settings. Therefore, they can pick out some characteristics, like location and liveliness, to assess the regions.

At the same time, many methods do not take the cost of each selected node, which is caused by some regional features, into account in the process of completing tasks. In crowd sensing, task requestors need to recruit specific numbers of users that are selected by some incentive mechanisms to participate in their tasks. Currently, there are many platforms that give requestors opportunities to recruit participants, such as Ohmage [8], Hive [9], and Prism [10]. Although these platforms can provide convenience, they have a problematic shortcoming. By adopting certain approaches, like having promotions or using social media in a system, we can have a large number of users. However, task requestors might find some difficulties in completing the requested task cost-efficiently. They not only require the selected service nodes to provide high-quality data, but they need to minimize the cost of each user during collection process for the budget cut. Therefore, if task requestors could reduce costs while considering region features, the reliability and feasibility of the data can be guaranteed. In addition, this could reduce costs for information gathering. To achieve this scenario, the region features should be considered in the incentive mechanisms.

Example 2. Under the scenario from Example 1, the task then asks each sensing node to provide pictures or videos to show the place they are at. In this situation, participants are required to take initiative to participate in the tasks of x sensing data collection, which means that the existence of wireless network access in the region can be crucial. Users need to pay for their cellular consumption without wireless network access, which will affect the users to participate actively. The incentive system then needs to provide higher compensation. Therefore, when choosing the appropriate set of service nodes for the sensing task, this characteristic, namely wireless network coverage in this region, should be considered.

Aiming at addressing the problems discussed above, this paper proposes a low-cost service node selection method that requires active user participation based on regional features for sensing tasks. For a particular sensing task, this method is first used to discover activity-intensive clusters based on movement traces of service nodes and obtain the center points of these clusters. Then, the given sensing area is divided according to the Voronoi diagram partition rules and each calculated center point becomes the center point of a corresponding Voronoi diagram. This partition rule has broad applications in many fields, such as visual imaging of invisible hazardous substances [11], uncertain data analysis [12] and privacy preserving [13]. It ensures that the distances from user locations in each Voronoi diagram to the center of the diagram are shorter than the distances to centers of other Voronoi diagrams in a certain area. The rule is more natural for the regional division in accordance with the activities of nodes, rather than the previous indiscriminate division. After that, the importance of each factor in the task is analyzed by using AHP method, and we can obtain the local feature vectors of each partitioned Voronoi diagram. AHP is a practical multischeme or multiobjective decision-making method proposed by American operation researcher Professor TL Saaty in the 1970s. It is a combination of qualitative and quantitative decision analysis methods. It is often used in multiobjective, multicriteria, multielement, multilevel unstructured complex decision-making problems, especially strategic decision-making issues, it has a wide range of practicality [14, 15]. When the system releases a sensing task, the matching degrees of the task and the local feature vector of each region are calculated to obtain a certain number of services nodes with higher matching degrees. Greedy Algorithm is then used to calculate the costs incurred by each selected node. Finally, some appropriate nodes in the area are selected to perform the sensing task. The whole process improves the accuracy, timeliness, and efficiency of node selection for completing the task.

The main contributions of this paper are as follows. First, this paper proposes a service node selection method that combines task requirements with region characteristics. A decision model is used to preferentially match the data sampling regions according to the feature vectors. Second, to reflect the activity density differences between regions, this paper proposes an area partitioning method based on the Voronoi diagram. Finally, the selected nodes are filtered through a simple algorithm to reduce the costs of completing tasks by selected service nodes.

The rest of this paper is organized as follows. Section 2 introduces the related work on node selection methods in crowd sensing networks at home and outdoors. Section 3 presents the implementation process and the algorithmic complexity of the method in detail. Section 4 applies the method to a real data set to evaluate the accuracy and speed by experimental comparisons. Section 5 summarizes this work and discusses the feasibility of this algorithm, as well as its shortcomings and prospects.

2. Related Work

Compared with the traditional wireless sensor network (WSN), the crowd sensing also uses GPS, camera, Wi-Fi, and other sensing devices, but the sensors of WSN are fixed; sometimes, there is a "sensing hole" problem; that is, where no sensors are deployed, we are unable to get the corresponding sensing information. However, in mobile crowd sensing, all sensors are mobile (because people are mobile), which is the expansion and promotion of WSN. In crowd sensing service, in order to reduce cost of data collection and obtain large-scale sensing data efficiently, some previous researchers provide many good solutions to node selection in crowdsourcing service.

Hachem et al. proposed a node selection algorithm based on a probabilistic configuration in a large-scale mobile participant sensing system. They proposed an algorithm for probabilistic allocation that allows users to decide whether to participate in sensing tasks based on the probabilities that users appear in the expected location by the mobile model of real-life human movement [16]. The algorithm can solve the problem of the node set being too large in the process of crowd sensing, and it relies on the intersections of users' paths in a dense network; different user devices can be substituted for each other in the acquisition of sensing data, and the number of nodes participating in sensing tasks is successfully reduced without reducing the requirements of regional coverage.

Reddy et al. proposed a participatory sensing algorithm by using context-aware mobile reference files to select nodes to perform sensing tasks [17]. They focused on establishing a service that can select an appropriate task node set for participatory sensing. This algorithm can make the creation, management, and organization of data collection more convenient for task requestors so that service nodes can easily find and participate in a sensing task in which they are interested.

Yu et al. proposed a node prediction model by implementing a Markov chain (O2MM) and then introduced a prediction improvement based on social relations (SMLP) [18]. The social relationship used in [18] is mainly by collecting the times and the length of contacts between the user and his friends within a certain sampling period (e.g., within one day). And [18] uses this indicator to measure the closeness of the relationship with friends. Then the system predicts the next position by an algorithm that integrates the user's historical trajectory and the location of his friend's current location. SMLP has higher accuracy than using historical trajectory data only. The Markov chain prediction model takes nodes' locations as states of the Markov process and extracts the residence time of each position state. It establishes a Markov chain forecasting model based on these positions and times to gather closely related groups into a community while taking into account the social relationships between people in the real world.

There are some other purposed algorithms and frameworks for node selections in crowd sensing. Xiong et al. showed their purposed framework to select sensing nodes with optimal cost and incentives in the whole sensing area in their paper by using prediction results, the estimated k-depth coverage and incentive cost [19]. Guo et al. purposed a framework called *ActiveCrowd* to select sensing nodes for tasks by using different greedy-enhanced genetic algorithms in the framework with respect to time-sensitivity [20]. Mendez et al. purposed a data interpolation method to get a representative sample in a group of nodes, which are selected by the matching degree of a specific sensing task, from each partitioned region [21].

Then some previous research focuses on task assignment for a group of selected nodes. Tong et al. purposed a two-step framework to assign tasks to a group of selected workers online or offline. The way of using maxflow in the job assignment can help the system have a general idea of which worker should get assigned a specific task [22]. Mendez et al. considered using density map as a tool to determine the distribution of sensing nodes. In this way, the system can have the quality of estimation the popularity in various regions [23]. Zhang et al. gave methods to optimize the process of assigning tasks to a selected group of workers. The methods basically require the users to give the confidence of finishing a specific sensing task in [24]. Then in [25], the authors focused on assigning tasks to different workers to maximize sensing coverage, and they use information entropy as the criteria to select a group of workers.

The above algorithms are relatively representative node selection algorithms in crowd sensing. However, these algorithms do not care about the node distribution pattern in the chosen sensing area. They use traditional division methods, some of which use other tools to help to partition, to produce divided regions. Moreover, attributes and features of locations or regions are closely related to sensing tasks in many cases and have a significant reference value for the accomplishment of the sensing tasks. Most of the above algorithms, nevertheless, do not consider these factors, which makes them not very useful for some tasks with specific requirements. Last but not least, some of the above algorithms do not consider the impact of node selection on the costs of the whole data acquisition process with some underlying geographic features.

3. ReLSNS Paradigm

To overcome the shortcomings of the algorithms discussed above, this paper proposes a low-cost service node selection method (ReLSNS). This method first clusters the tasks, and then divides the regions based on the Voronoi diagram, so that the appropriate participating nodes in the sensing region can be more accurately found. In turn, it saves costs and improves efficiency. Before introducing this method, we first define the area from which service nodes will be selected. We define the whole sensing area as A, and the scale of A is determined by the sensing task requestors according to the task type and task granularity. For example, if an institution needs to study the points of interest of students on the campus, A can represent the campus.

The participant set that is used in a particular sensing task is defined as U. The number of participants must be sufficient; otherwise, the quality of coverage may be degraded. There are many ways to recruit participants. One is to recruit volunteers who are interested in sensing tasks and are willing to participate in the sensing task for free. The other is to attract a larger group of people through a small group of selected people, by allowing them to tell friends and family what organizations they take part in and what they should do in this task. Third, task requestors can attract participants through a positive incentive mechanism: when participants provide data of high quality, task requestors provide money, coupons, and other rewards related to their own personal interests. The third method is now the most common since in crowd sourcing platforms, the typical extrinsic motivational factors are reputation, status, peer pressure, fame, community identification and fun [26].

3.1. Architecture of ReLSNS. As shown in Figure 1, the specific steps of this method include three parts: area partitioning based on the Voronoi diagram, region matching based on feature vectors, and the selection of nodes by using Greedy Algorithm. And some steps, such as step 2 and 3, are given some illustrations of each step in the sensing area. In the different parts, there are some separate support modules, as shown in Figure 1. The selection method proposed in this paper will be described in detail below.

3.2. Voronoi Diagram-Based Area Partition. Compared with the traditional Grid Division method, Voronoi diagram-based area partitioning uses the characteristic of local similarity of the geographic area feature to divide the whole sensing region into multiple regions with irregular shape but different characteristics. Since crowd sensing is a sensing mode that is centered on human beings, the regional characteristics are determined by the region with highest sensory node activity. Therefore, we first use the historical moving trajectories of human-induced sensing nodes to discover active node clusters. Here, the recording pattern for each user's history in the present method is first proposed.

3.2.1. Prerequisite Knowledge

Definition 1. Track Point Set Td consists of all track point data of mobile nodes in U in area A, which can be represented as follows:

$$Td = \{S_n \mid \forall u \in U, (x, y) \in A\} \tag{1}$$

In this equation, S_n represents as a track point, which is a trajectory point that represents the location of the mobile node u ($u \in U$) at a time t, and is represented by a four-element tuple,

$$S_n = (u, x, y, t), \tag{2}$$

where t denotes time, x denotes the latitude of the node u at t, and y denotes the longitude of the node.

If trajectory points of the mobile node are extracted in the same time interval, the connected trajectory points can approximate activities performed by the node. To differentiate among nodes, unique numbers or MAC addresses of user devices can be used to represent those nodes.

To discover clusters, high-activity regions in area A should be found by clustering regions with high track point density. In this paper, we use the revised DBSCAN algorithm for clustering to fit in each sensing task. The basic sense of this algorithm is to group together sensing points that are closely located together in a certain region. Different tasks might have different values of radius of clusters. The pseudocode of this algorithm is shown in Algorithm 1.

In this algorithm, Tdc is a set of mobile node track points for region clustering and can be determined according to the following definition.

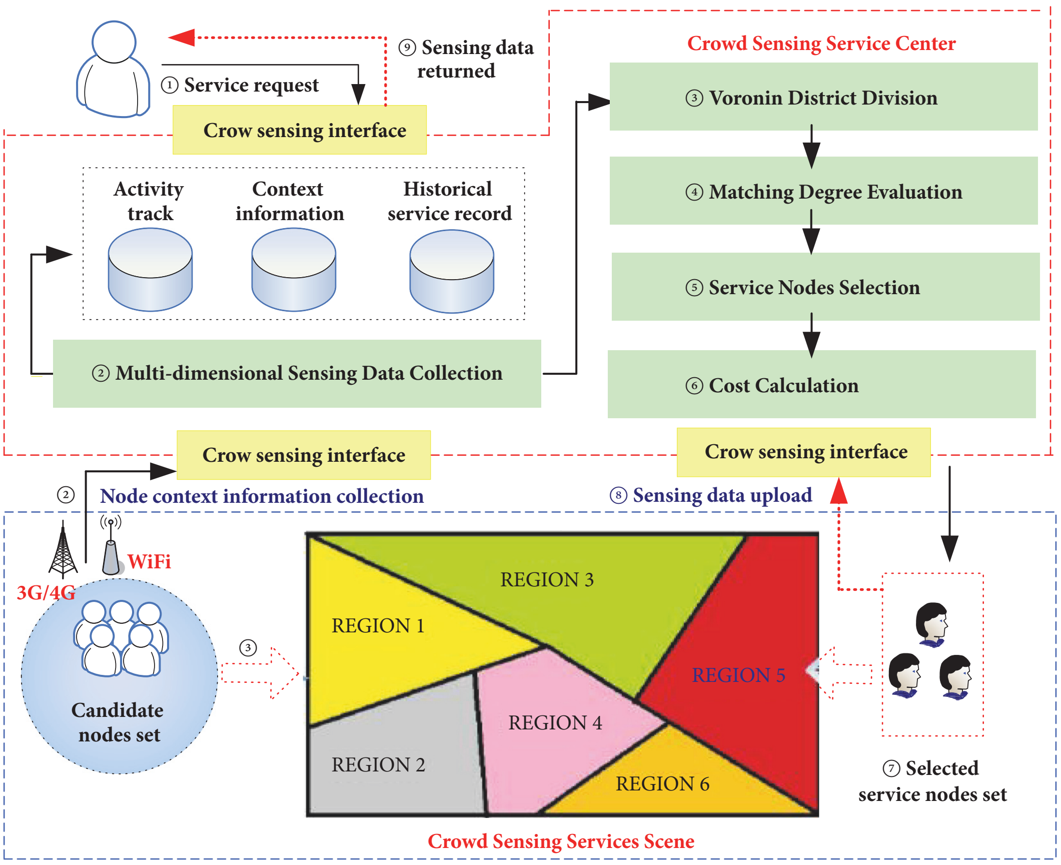
Definition 2. Specific Track Point Set Tdc contains some specific elements, as defined below.

$$\begin{gathered} Tdc \subseteq Td, \\ p \in Tdc, \\ p(t) \in [t_1, t_2] \\ t_1, t_2 \in T \end{gathered} \tag{3}$$

where Tdc belongs to Td and the time attribute t for each track point $p_i (p_i \in Tdc)$ in Tdc satisfies $t \in [t_1, t_2]$, in which t_1 and t_2 are determined according to the specific requirements of the sensing task.

For example, if the task should be completed from 11:00 to 12:00, then t_1 and t_2 will be set to eleven and twelve, respectively. After that, by using the existing track data in the system, track points sensed between eleven and twelve o'clock can be filtered to obtain a set of specific discrete points Tdc for clustering.

In addition, the clustering radius ε and the density threshold $MinPts$ are determined by the specific sensing tasks. The primary method for determining these two parameters is the distance (k-distance) from the observation point to its k-nearest neighbors [27]. For a point belonging to a cluster, if k is not larger than the cluster size, then the k-distance will be minuscule. And for points that are not in any clusters, the k-distance can be relatively large. Therefore, for a particular value of k, we should calculate the k-distances of all points and sort them in ascending order. In addition, when we plot the sorted values, it is expected that a sharp change



① Service request
⑨ Sensing data returned
Crow sensing interface
Crowd Sensing Service Center
③ Voronin District Division
④ Matching Degree Evaluation
⑤ Service Nodes Selection
⑥ Cost Calculation
Activity track
Context information
Historical service record
② Multi-dimensional Sensing Data Collection
Crow sensing interface
Crow sensing interface
② Node context information collection
⑧ Sensing data upload
WiFi
3G/4G
Candidate nodes set
③
REGION 1
REGION 2
REGION 3
REGION 4
REGION 5
REGION 6
⑦ Selected service nodes set
Crowd Sensing Services Scene

Figure 1: The architecture of ReLSNS.

in k-distance will be seen. According to the change in k-distance, which corresponds to the appropriate value of ε, the parameter ε can be chosen, and k can be taken as the $MinPts$ parameter. The above method can theoretically select the optimal ε and $MinPts$ values. In practice, since the number of sensing users in the network and their distribution are relatively fixed, the values of ε and $MinPts$ can be fixed as the network eigenvalues and do not need to be changed every time. If the error is significant, these parameters can be adjusted over time. If we consider the weight of each node, we can use the fuzzy k-nearest neighbor method that was proposed by James Keller to set ε and $MinPts$ [28].

Through Algorithm 1, we can obtain several sets of discrete points, with each set representing a region with a high density of discrete data points; that is, each discrete point set C_i represents an aggregation region. Finally, a set of perceived node clusters $C = \{C_1, C_2, C_3, \ldots, C_n\}$ can be obtained.

Consider that it is difficult to describe the characteristics and state of each geographical region by continuous geographic location coordinates, it is necessary to divide the sensing area and discretize the geographic location. After obtaining the clustering set C, we use coordinate values of all track points in each C_i $(C_i \in C)$ and calculate the coordinate set of center points of all clusters $O_c = \{O_{c_1}, O_{c_2}, O_{c_3}, \ldots, O_{c_n}\}$. The center points O_{c_i} can be generated using the following definition.

3.2.2. The Generation Process of Voronoi Diagram

Definition 3. Clustering center point O_{c_i} can be generated by the track points in the clustering region C_i from the previous DBSCAN algorithm and can be represented by the following four-element tuple:

$$O_{c_i} = \{i, x, y, [t_1, t_2]\}. \tag{4}$$

Since the time interval was filtered in the previous step, the original time parameters of sensing nodes will be used to represent the time interval. Each center point can be numbered by the index of the clustering region i. Its latitude and longitude, namely, x and y, can be generated by each sensing node p using the following formulas.

$$O_{c_i}(x) = \frac{1}{s}\sum_{j=1}^{s} C_i\left(p_j(x)\right) \tag{5}$$

$$O_{c_i}(y) = \frac{1}{s}\sum_{j=1}^{s} C_i\left(p_j(y)\right) \tag{6}$$

where S denotes the number of sensing nodes in a clustering region C_i.

Then, based on the center points that were calculated above, we use the Voronoi diagram partition method [29] to

```
Input: Trace dataset $Tdc = \{p_1, p_2, p_3, ..., p_n\}$,
Radius $\varepsilon$, Density Threshold $MinPts$
Output: Density-based trace dataset
1:  mark all the nodes in $Tdc$ unvisited
2:  while not all the nodes are visited do
3:    select an unvisited node $p$ from $Tdc$
4:    mark node $p$ visited
5:    define the $\varepsilon$-domain of $p$ as the area with radius $\varepsilon$ centered on $p$
6:    if $\varepsilon$-domain of $p$ has at least $MinPts$ nodes then
7:      create a new cluster $C$ and add $p$ into $C$
8:      make $C'$ as a set of the $\varepsilon$-domain of $p$
9:      for each $p \in C'$ do
10:       if $p$ is unvisited then
11:         mark $p$ as visited
12:         if $\varepsilon$-domain of $p$ has at least $MinPts$ nodes then
13:           put the nodes into $C'$
14:         end if
15:         if $p$ is not the member of any clusters then
16:           put $p$ into $C$
17:         end if
18:       end if
19:     end for
20:     output $C$
21:   else
22:     mark $p$ as noise
23:   end if
24: end while
```

Algorithm 1: Density-based spatial clustering of applications with noise (DBSCAN).

divide the whole sensing area into irregular polygon regions. In this paper, the Delaunay triangulation [30] is used to divide the area, and the process of Delaunay triangulation can be completed by using Convex Hull Interpolation algorithm [31].

The black dots in Figure 2(a) are the center points obtained by clustering the tasks. The red dot in Figure 2(b) is the center of the circumscribed circle of each triangle. Connect these centers around the node and finally form a Voronoi diagram, as shown in Figure 2(e).

When the process of Delaunay triangulation has been completed, there are likely to be some center points that are located on boundary lines when delineating boundaries. To generate the triangulation, we define the right-hand method in the process of division, which can help task requestors avoid this problem without having a significant impact on the overall vector generation process.

Definition 4 (right-hand method). When a region is divided, a vector is generated that passes through a dividing line, namely, the boundary line of adjacent regions. We define the line direction as the direction from the side of the node with the smaller label to the side with the larger label.

Consider a sensing region from South to North on a map. The point on the boundary is moved vertically by a unit distance (defined by the map) in the right-hand direction, and the new coordinates are assigned to the node.

Using the methods described in this section, we can obtain some regions in a specific sensing area. Next, we generate feature vectors in these regions to help us select service nodes.

3.3. Eigenvector-Based Regions Matching

After partitioning area A to obtain some irregular Voronoi polygon regions, we extract the eigenvectors of each region according to a historical task set, which we call regional feature vectors; these vectors are defined as follows.

3.3.1. Generation Process of Task Feature Vector

Definition 5 (task feature vector (TFV)). A TFV is a vector of elements that describe the characteristics of a region. Each element of the vector reflects one characteristic of the region. For each Voronoi polygon region a_i ($a_i \in A$), the task feature vector is denoted as $TFV(a_i)$.

In this paper, we also abstracted out four task-related regional characteristics:

Location Information (Li). Local information, such as restaurants, hotels, shopping malls, banks, stadiums, and libraries, can be obtained by comparing maps from applications such as Amap and Google Maps.

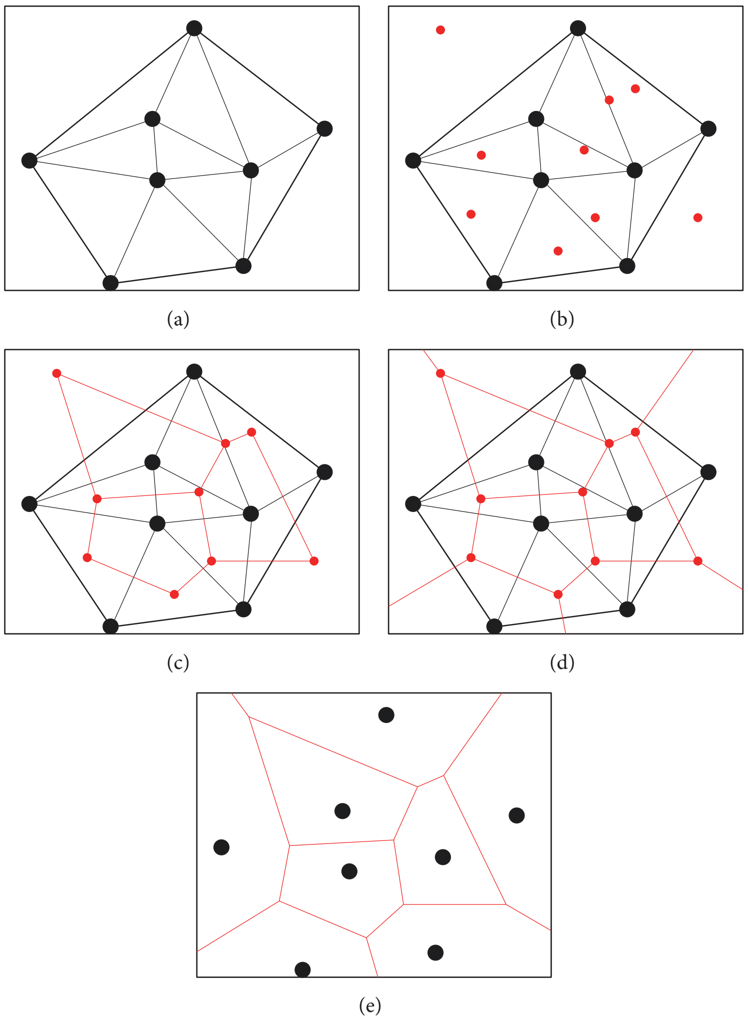


Figure 2: Generating a Voronoi diagram from Delaunay triangulation.

Timeliness (Tl). This characteristic can be calculated by learning from historical data, namely, by calculating the average time spent from receiving tasks to uploading data by the nodes in the region.

Liveliness (Ln). Through the study of historical data, the total amount of effective task data that has been uploaded by nodes in an area can be obtained, which we call liveliness.

Wi-Fi Coverage Information (Wi). By learning historical data, we can determine whether nodes upload historical task data through Wi-Fi, mobile 3G/4G network or other means in an area. Then, the ratio of Wi-Fi uploading to total uploading can be calculated as Wi.

The above four features are characterized by high correlation between regional location and task, and all of them are universally applicable. This $TFV(a_i)$ can also be used in the given examples in Section 1. In addition, $TFV(a_i)$ can add new elements according to the specific sensing scenarios or task requirements. Next, we take these four factors as an example to explain how $TFV(a_i)$ can be expressed in more detail as follows:

$$TFV(a_i) = (Li(a_i), Tl(a_i), Ln(a_i), Wi(a_i)), \qquad (7)$$

where Li can be obtained by querying electronic map matching, and the resting three characteristics are obtained by analyzing historical effective task data. The detailed process is shown in Figure 3.

In this flowchart, the method of calculating the four features in $TFV(a_i)$ is illustrated. The timeliness of task execution in the region is calculated by computing the average time spent on a particular task by service nodes in the region. Since the algorithm in this section needs to be performed by the nodes in regions with high matching degree, only nodes that accept the task in the region are selected in the calculation of the task timeliness in the region. We do not consider nodes that are passing the region and acquiring the sensing data in the region when the sensing task is performed. For calculating the liveliness in a region performing the task, the activity degree of the acquisition of the sensing data in the region is taken into account. If the mobility activity of nodes and activity of receiving data of nodes are high in a region that otherwise has a relatively low degree of data acquisition, the region should not be considered an important

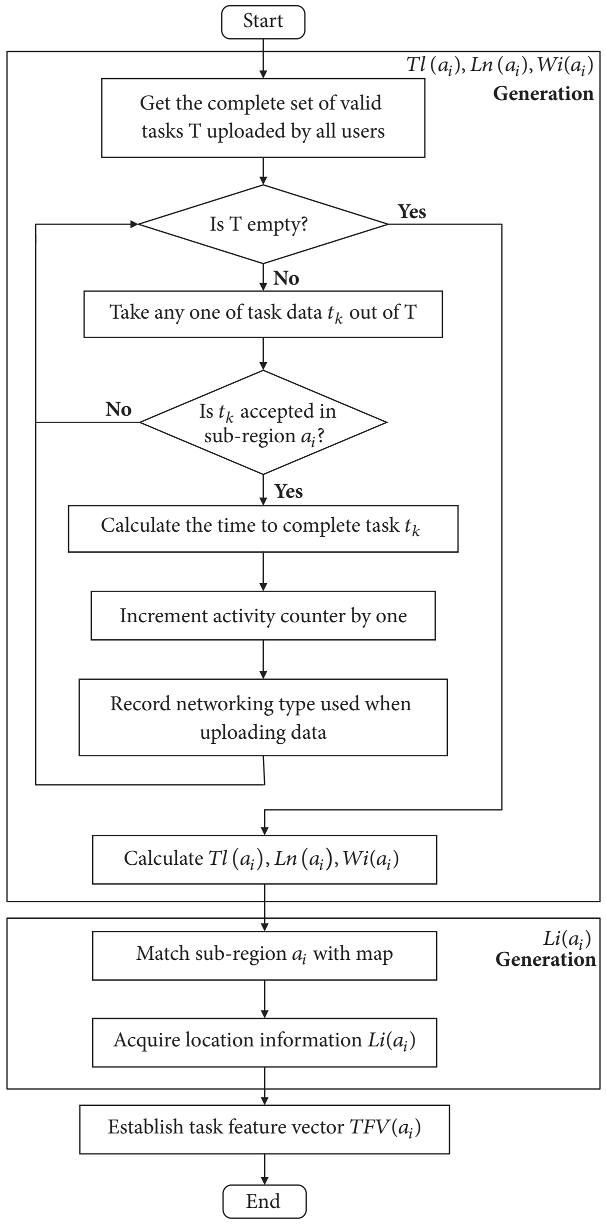

Start
$Tl(a_i), Ln(a_i), Wi(a_i)$ Generation
Get the complete set of valid tasks T uploaded by all users
Is T empty?
Yes
No
Take any one of task data t_k out of T
No
Is t_k accepted in sub-region a_i?
Yes
Calculate the time to complete task t_k
Increment activity counter by one
Record networking type used when uploading data
Calculate $Tl(a_i), Ln(a_i), Wi(a_i)$
$Li(a_i)$ Generation
Match sub-region a_i with map
Acquire location information $Li(a_i)$
Establish task feature vector $TFV(a_i)$
End

FIGURE 3: Task feature vector generation process.

source of perceived data. Moreover, the liveliness in a region is obtained by calculating the number of nodes in the region that are acquiring the sensing data. For the Wi-Fi coverage information, when user cellular data consumption in the process of uploading data is considered, task requestors should be aware that the main actors in crowd sensing are human beings, and people's smart handheld terminals, when used to obtain data, consume cellular data. In contrast, people are not charged when uploading data in Wi-Fi coverage areas. Therefore, the Wi-Fi coverage information is calculated as the ratio of the number of times the Wi-Fi users upload task data to the total number of sensing data uploads in the area.

3.3.2. Normalization of Feature Vector

Definition 6 (regional functional requirement set L_{ref}). This set is used to describe the collection of region functional attributes, such as dining area, culture, and art area. L_{ref} can contain one or more regional functional attribute parameters. If the sensing task needs to gather information about the athletic activities of students in a certain school, L_{ref} can contain only one functional attribute, which represents sports venues in this region. If the sensing task needs to collect traffic congestion conditions in a region, since there is no regional functional requirement, all regional function attributes of the region can be added to L_{ref}.

The four elements in $TFV(a_i)$, namely, position, length of time, number of times, and proportion, have different meanings. To facilitate the calculation, the four elements of $TFV(a_i)$ should be normalized to between 0 and 1. Tl, Ln, and Wi in $TFV(a_i)$ are represented by numbers, which are normalized to map eigenvalues to numbers between 0 and 1; Li in $TFV(a_i)$ is obtained directly by matching $Li(a_i)$ with L_{ref}, which is given by the task requestors. The normalization method is presented in Table 1.

After calculating an eigenvector for each region, we need to evaluate the preferences for each factor based on the assigned task and find a specific vector to represent the characteristics of the corresponding task. Therefore, two definitions are introduced here.

After calculating an eigenvector for each region, we need to evaluate the preferences for each factor based on the assigned task and find a specific vector to represent the characteristics of the corresponding task. Therefore, two definitions are introduced here.

Definition 7 (task reference vector M_{ref}). This vector is used to describe the degree of reference of the task to each task feature factor. It is also denoted as follows:

$$M_{ref} = \left(Li_{ref}, Tl_{ref}, Ln_{ref}, Wi_{ref}\right). \tag{8}$$

Each component in this vector represents how important the corresponding factor is to this task, where $Li_{ref} + Tl_{ref} + Ln_{ref} + Wi_{ref} = 1$. Take Location information Li as an example. While it is straightforward to obtain numerical values for the other factors, a numerical value must be defined for Li:

$$Li_{ref} \begin{cases} = 0, & \textit{when the task doesn't refer to this factor} \\ \in (0,1], & \textit{when the task refers to this factor} \end{cases} \tag{9}$$

3.3.3. Feature Vector Processing Based on AHP

Definition 8 (region matching degree vector ζ). Each component of this vector is used to measure the degree of matching between the task and a region feature. Each component in ζ is a value between 0 and 1. The larger the component value is, the higher the matching degree of the task and the selected region feature is, and the greater the probability that service nodes in the region can perform the sensing task as required.

Thus, based on the definitions given above, Analytic Hierarchy Process (AHP) is used to obtain the above task

Table 1: Normalization method.

Region Feature	Normalization Method
Location information (Li)	For every region a_i in area A, match $Li(a_i)$ with L_{ref}. If $Li(a_i)$ is contained in L_{ref}, set $Li'(a_i)$ as 1. Otherwise, set $Li'(a_i)$ as 0.
Timeliness (Tl)	Find the smallest Tl in area A, which is denoted as $a_{\min}$. Set $Tl'(a_{\min})$ as 1, and compare the value of $Tl(a_{\min})$ with the $Tl(a_i)$ values of all the regions in area A. Let $Tl'(a_i)$ denote the resulting ratio.
Liveliness (Ln)	Find the largest Ln in area A, which is denoted as $a_{\max}$. Set $Ln'(a_{\max})$ as 1, and compare the value of $Ln(a_{\max})$ to the $Ln(a_i)$ values of all the regions in area A. Let $Ln'(a_i)$ denote the resulting ratio.
Wi-Fi coverage information (Wi)	Find the largest Wi in area A, which is denoted as $a'_{\max}$. Set $Wi'(a'_{\max})$ as 1, and compare the value of $Wi(a'_{\max})$ with the $Wi(a_{\max})$ values of all the regions in area $\boldsymbol{A}$. Let $Wi'(a_{\max})$ denote the resulting ratio.

Table 2: The value of $k_{i,j}$ and its meaning.

Value of $k_{i,j}$	meaning
1	k_i and k_j are equally important
3	k_i is slightly more important than k_j
5	k_i is more important than k_j
7	k_i is much more important than k_j
9	k_i is extremely important than k_j
2, 4, 6, 8	The importance of the k_i to $\mathrm{K_j}$ is between the two adjacent levels

reference vector M_{ref}. When AHP is used, it is measured using a generalized pairwise preference scale to produce a corresponding 4×4 preference matrix, as shown below:

$$Ref = \left(\begin{array}{c|cccc} & k_1 & k_2 & k_3 & k_4 \\ \hline k_1 & 1 & k_{1,2} & k_{1,3} & k_{1,4} \\ k_2 & k_{2,1} & 1 & k_{2,3} & k_{2,4} \\ k_3 & k_{3,1} & k_{3,2} & 1 & k_{3,4} \\ k_4 & k_{4,1} & k_{4,2} & k_{4,3} & 1 \end{array}\right) \tag{10}$$

where $k_{i,j}$ represents the ratio of the importance of k_i to k_j, $k_{i,j} * k_{j,i}$=1, and $\forall k_{i,j}$ $(i, j \in [1,4] \land i, j \in \mathbb{Z})$, $k_{i,j} \in [1,9] \land k_{i,j} \in \mathbb{Z} \land i, j \in \mathbb{Z}$.

$k_i, i \in [1,4] \land k_i \in \mathbb{Z}$ mean the four decision factors which are shown in Table 1. For example, we can set $k_1 = Li$, $k_2 = Tl$,$k_3 = Ln$, $k_4 = Wi$.

According to [14, 15], the value of $k_{i,j}$ and its meaning are as Table 2.

The matrix (10) indicates the importance of each factor relative to other factors. For example, the columns of the matrix represent the importance of location information, timeliness, liveliness, and Wi-Fi coverage information from the perspective of task requestors, from left to right, with rows representing location information, timeliness, liveliness, and Wi-Fi coverage information importance from top to bottom. Then, for task requestors, if location information is crucial comparing to Wi-Fi coverage information in the task, the corresponding position in the matrix, namely, $k_{1,4}$, can be set as 7. For the selected area A, the value of Wi-Fi coverage information comparing to location information, namely, $K_{4,1}$, must be set as $1/7$.

Finally, by using AHP in the matrix processing steps, we can finally obtain a 4×1 task reference vector M_{ref}. We combine all normalized task feature vectors $TFV'(a_i)$ to form a feature vector matrix of size $n \times 4$, namely, TFM (Task Feature Matrix). The following formula is used to calculate an $n \times 1$ region matching degree vector ζ:

$$\zeta = TFM \cdot M_{ref}. \tag{11}$$

The components of vector ζ represent the reference values of the respective regions under the relevant task characteristics. The larger the value, the better the feature in the corresponding region matches the task.

3.4. Selection of Service Nodes Based on Greedy Algorithm.

After obtaining Region Matching Vector ζ, all regions are sorted in descending order according to the values of the elements in the vector. That is, the regions with high matching degrees are arranged in the front of a priority queue. Nodes in the regions at the front of the queue perform the task with priority higher than the nodes in the other regions. The number of nodes that are required to perform a task is determined by the task requestors when the task is published, which is denoted as Q. Therefore, to improve the performance of the proposed algorithm, a certain number of regions are selected, which contain nearly 2Q nodes. We denote the number of selected regions as N and the number of selected nodes as K. These regions possess characteristics that match task characteristics. However, the distance cost is not considered. Therefore, further processing is required after selection.

Then, the Greedy Algorithm is used to obtain low-cost regions to achieve a better balance between possessing task characteristics and the cost of task accomplishment. Finally, the necessary service nodes are obtained. Before describing the algorithm, define the node score vector.

Definition 9 (node score vector δ_k). This vector is evaluated using an estimate of the current situation of each service node, which is used to measure the degree of matching of

```
Input: Task-featured Region Dataset $C = \{C_1, C_2, C_3, \ldots, C_N\}$, and Location of the specific task $O_T = (x_T, y_T)$
Output: Scores of labor cost of each region
1:      create a distance vector $D = \{D_1, D_2, D_3, \ldots, D_N\}$
2:      for each $C_i \in C$ do
3:        while not all nodes in $C_i$ have been visited do
4:          select one unvisited node from $C_i$
5:          set it as visited
6:          calculate Manhattan distance from this node to $O_T$
7:          save the calculating result in $D_k$ with the corresponding node identified number
8:        end while
9:      end for
10:   for each $D_k \in D$ do
11:   compare the value of $D_k$ with other elements in the vector and rank
12:   end for
13:   Based on the order in $D$, score each region with a rule that a shorter calculated distance can have a higher score
```

Algorithm 2: Score the regions based on labor costs.

regional features and the cost-effectiveness of a node. δ_k is defined as follows.

$$\delta_k = \left(u_i, Score_i\left(M_{ref}\right), Score_k\left(Cost\right)\right) \qquad (12)$$

In this vector, u_i represents the number given to a specific region. $Score_i(M_{ref})$ represents the score obtained from the process of evaluating the matching degree of each region, and the cost score $Score_k(Cost)$ represents the score obtained from the calculated distance cost. $Score_k(Cost)$ is simply obtained from the distance between the service node and the task. The shorter the distance, the higher the cost score. In Algorithms 2 and 3, service nodes can be selected according to these two values. In practice, costs include many factors such as the distance, the motivation of the task, the difficulty of the task, and so on. However, in our system, in order to simplify the problem, we only considered the simplest factor "distance", and we will involve injecting more complex cost functions with other more factors in the follow-up study.

We can compute the scores in the node score vector δ_k according to the following rules. The values of $Score_i(M_{ref})$ can be scores from N to 1 in descending order, based on the order obtained by using the method that was described in the previous section, with each score decremented by one. The values of $Score_k(Cost)$ in the region score vector δ_k can be calculated according to the corresponding task latitude x_T and longitude y_T using Algorithm 2.

In Algorithm 2, the distance of each node is calculated and the corresponding Manhattan distance is saved in a distance vector D. Therefore, by using Algorithm 2, the distance costs of all the service nodes of each region can be considered. Finally, we can obtain the corresponding $Score_k(Cost)$ by using the rule that a node with a shorter distance to the target place has a higher cost score. After completing the steps mentioned in Algorithm 2, the service nodes can be selected by using the revised Greedy Algorithm, which is presented in Algorithm 3.

Ω is a set of candidate nodes and is the final result we need. In Algorithm 3, since the positions of the tasks may change at any time, the process is dynamic. Moreover, the final score of each service node set task characteristic and the distance cost have the same weight, which can be changed based on the tasks. The Greedy Algorithm described in Algorithm 3 obtains a locally optimal solution, which is more convenient and suitable for low-cost service node selection.

3.5. Algorithm Complexity Analysis. In the process of Voronoi diagram partitioning, the first step is to extract the centers of the discovered clusters. If the number of nodes that meet the conditions in sensing area A is N, then the time complexity is $O(N\log N)$. Then the time complexity of Delaunay triangulation is $O(1)$, and the time complexity of this step will not exceed $O(N)$ for extracting center points from clusters. Based on an analysis of the process of Voronoi diagram division, if the number of center points is r_0, the time complexities of convex hull generation, convex hull split, discrete point interpolation, and Voronoi diagram division are $O(r_0)$, which is smaller than $O(N)$. Therefore, the time complexity of Voronoi diagram partitioning is $O(N\log N)$.

In the process of eigenvector-based region matching, the establishment of task feature vectors is based on historical data for the task, which is already in the system. Then it is assumed that the quantity of the above-mentioned historical data is N_0. To create a task feature vector, the historical data only need to be accessed once, so the time complexity is $O(N_0)$. When normalizing some of the elements in the task feature vectors, namely, $Tl(a_1)$, $Ln(a_1)$ and $Wi(a_1)$, the operations are performed only once, and then, data processing begins. Therefore, the time complexity of normalization is approximately $O(r_0)$. For calculating degrees of matching, the AHP method has the time complexity of $O(r_0)$. Finally, we can use Algorithm 2 to obtain the scores of each element. Even if a specific sorting process has been selected, the time complexity can be reduced to $O(r_0\log r_0)$ or less by using an appropriate sorting algorithm. For the revised Greedy Algorithm, this process can produce a time complexity of $O(2KQ)$. Therefore, the time complexity of the whole step is $\max\{O(r_0), O(r_0\log r_0), O(N_0), O(2KQ)\}$. Thus, the entire

```
Input: Region-scored Vector Set $\Delta = \{\delta_1, \delta_2, \delta_3, \ldots, \delta_K\}$
Output: Selected Node Set $\Omega$
1:    $k = 1$
2:    while $\Delta \neq \varnothing$ & $k \neq Q$ do
3:      calculate the final score of each element by adding $Score_i(M_{ref})$ and $Score_k(Cost)$
4:      select the element having the highest score
5:      add the number of node selected to $\Omega$
6:      $k = k + 1$
7:      eliminate the selected vector from $\Delta$
8:    end while
```

Algorithm 3: Revised Greedy Algorithm.

process of eigenvector-based region matching has a time complexity of $\max\{O(r_0), O(r_0 \log r_0), O(N_0), O(2KQ)\}$.

4. Experimental Comparison

The experimental results of this method are compared with those of the SMLP model based on social relations and the subset selection algorithm in O2MM. Since this method is task-centered and selects nodes by region matching, it is very different from other node selection algorithms. Therefore, direct comparison with other algorithms is not convincing. Considering that SMLP and O2MM are trace prediction models of participatory crowd sensing, this paper simulates the node selection algorithm based on their trajectory predictions to carry out the comparison with the method of this paper. In the O2 MM model, a location set of the sensing area is used to simulate several task locations that need to acquire sensing data in the task publishing stage. The locations obtained in the model are sorted according to their distances to the targeted location. Then, it selects nodes with shorter distances in the set to perform the task. If the number of nodes is insufficient to satisfy the need, the probability of other nodes moving to the targeted location is predicted, and the nodes with the highest probabilities are selected until the number of nodes meets the requirement. The SMLP model optimizes the O2MM model by incorporating the social relation attributes of the nodes. In this section, the simulation of the node selection algorithm of SMLP is basically same as that of O2MM, but the social relation attribute is considered in the process of predicting the nodes. The method proposed in this paper, namely, ReLSNS, considers only two elements in TFV: location information and timeliness. It is not possible to consider other features when using the same parameters as in SMLP and O2MM. Moreover, we assumed that task characteristics are much more vital than distance costs of service nodes.

The test environment is based on our cloud computing center, with 107 blade servers and 200T storage space. The testing data(the data used in the study can be accessed: UCSD Wireless Topology Discovery Project. http://www.sys-net.ucsd.edu/wtd/wtd.html) is from the Wireless Topology Discovery (WTD) [21], which contains data from 275 PDA users on the campus of the University of California, San Diego, for 11 weeks. Because the WTD dataset is too large,

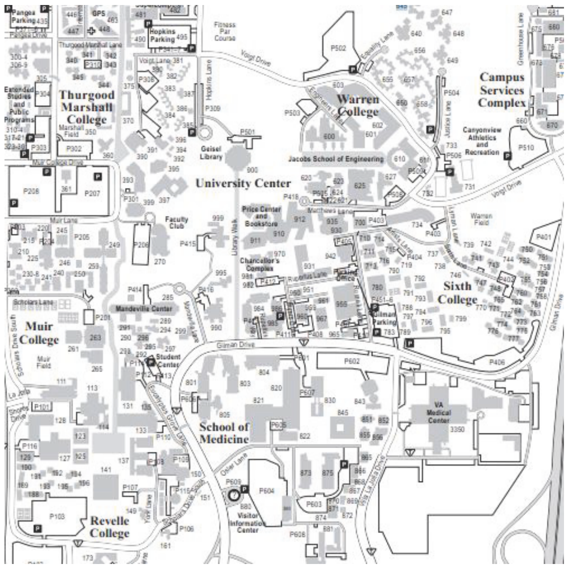


Figure 4: Map of University of California, San Diego.

we choose data (1,106,308 records) from one week from six colleges of the University of California, San Diego campus center, to simulate the above algorithms and method. There are 208 mobile nodes in the dataset. The actual map corresponding to the active range of these mobile nodes is shown in Figure 4.

Moreover, since the data acquired in WTD dataset has set a signal strength parameter, we can use the following equation to calculate the distances of sensing devices:

$$Lbs\,(dB) = 32.45 + 20\lg F\,(MHz) + 20\lg D\,(km) \qquad (13)$$

where Lbs represents the transmission loss, which can be calculated when we know transmission power and sensitivity of the devices. F denotes the working frequency. Here, we set $F = 2.4GHz$, which is a normal working frequency in many places. Finally, we can obtain the distances from the service nodes to the destination.

4.1. Results of Area Partition. First, the trajectory of the 1,106,308 data instances is analyzed, and the area covered by all the nodes is partitioned. According to the above two

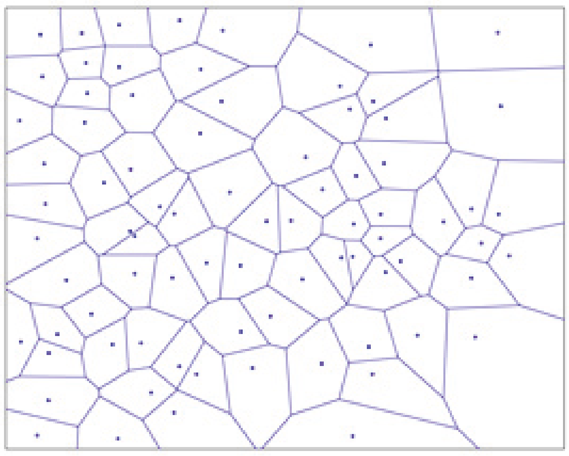


Figure 5: Partition result.

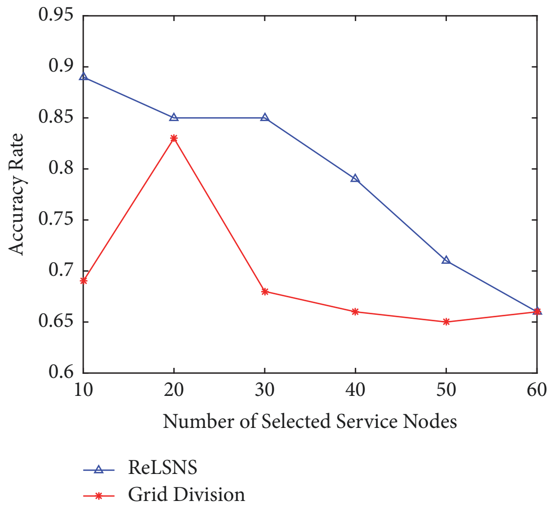


Figure 6: Accuracy of different division methods.

parameters, the area can be divided into 79 regions. We set the clustering radius of the selected dataset to 100 meters and the minimum number of nodes in each cluster to 5000. Finally, the area division result is shown in Figure 5, for comparison with the map mentioned in Figure 4.

The result is basically in accordance with the density of buildings and the population of each region in the map. The sparse area of human flow in the original area is evenly divided into the Voronoi polygons where the nearest center points are located. The regions with a high density of center points are mainly concentrated in the six colleges in the original area, so the area partition result reflects each region's important geographical features.

4.2. Evaluation of Service Node Selection Accuracy. We use the method proposed in this paper to select service nodes and set the requirements of the selected nodes differently each time we simulate a task. The numbers of service nodes of tasks that were required for the simulation are 10, 20, 30, 40, 50, and 60. For each requirement, we perform four tests, varying the selection of attributes and we count the numbers of nodes that satisfy the requirements of the task. We first calculate the distance of the filtered service node to a specific task. When the distance is less than a certain threshold (set by experience, in the experiment, we set it to 100m), we believe that the node will accept the perceived task. Finally, we calculate the ratio of the number of nodes that will accept the task to the number of service nodes selected by the entire algorithm. This ratio is defined as the accuracy.

The results of four experiments are shown in Table 3. Experimental results showed that ReLSNS could provide task requestors with relatively stable service node selection results, while the results of traditional Grid Division are not guaranteed to be stable. The reason is that Grid depends on the location of a particular sensing task, which can result in completely different accuracy rates due to the geographical location of the simulated sensing tasks. This means that if there is a sensing task that is far away from all of the service nodes, this task may not be able to complete. Our method is relatively stable, because the nature of the Voronoi diagram, the service node can be well divided into a polygon based on geographic features. Even though the Grid Division process obtains a good node selection in some tests, the fluctuation is undesirable in the process of service node selection in practice.

From the results presented in Table 3, the average numbers of nodes that are selected by the method in this paper are obtained, under the same requirements. Then, we calculate the accuracy rates of service node selection of our method, which are shown in Table 4, while the average accuracy rate comparison between the proposed ReLSNS and traditional Grid Division is shown in Figure 6.

ReLSNS, which was proposed in this paper, is compared with two other participatory crowd sensing algorithms, namely, the social relation-based prediction model (SMLP) and the second-order Markov model (O2MM). The accuracy rate comparison of service node selection is shown in Figure 7. The accuracy rates of selected service nodes decrease as the required number of selected nodes increases. Because ReLSNS always chooses the service node with the highest matching degree, the more nodes that are selected, the lower the average matching degree of the selected service nodes, which results in a decrease in the accuracy of the selected set of nodes. According to the figure, when the number of service nodes in the selected node set is less than 20, the performance of ReLSNS in this paper is slightly worse than that of SMLP, which nonetheless still has nearly 90% accuracy. When the number of selected service nodes is more than 30, the accuracy of the proposed method is obviously higher than those of the other algorithms. In general, the method proposed in this paper has a higher accuracy rate.

4.3. Evaluation of Timeliness. In crowd sensing, the timeliness of task execution is an important criterion for measuring task execution quality. By obtaining the distance between the location of the node in the selected service node set and the nearest position where effective sensing data can be obtained, we can obtain the time taken by each node to

Table 3: Comparison between ReLSNS and Grid on nodes selection.

SelectedNodes	Experimental Result							
	1st Time		2nd Time		3rd Time		4th Time	
	ReLSNS	Grid	ReLSNS	Grid	ReLSNS	Grid	ReLSNS	Grid
10	9	2	9	10	9	6	8	0
20	18	20	17	6	16	20	17	20
30	21	6	29	22	25	21	27	28
40	31	30	33	29	32	39	27	6
50	35	2	33	48	39	49	32	28
60	38	49	40	60	42	21	38	22

Table 4: Accuracy rate.

Node required	10	20	30	40	50	60
ReLSNS	0.88	0.85	0.85	0.79	0.71	0.66
Grid Division	0.69	0.83	0.68	0.66	0.65	0.66

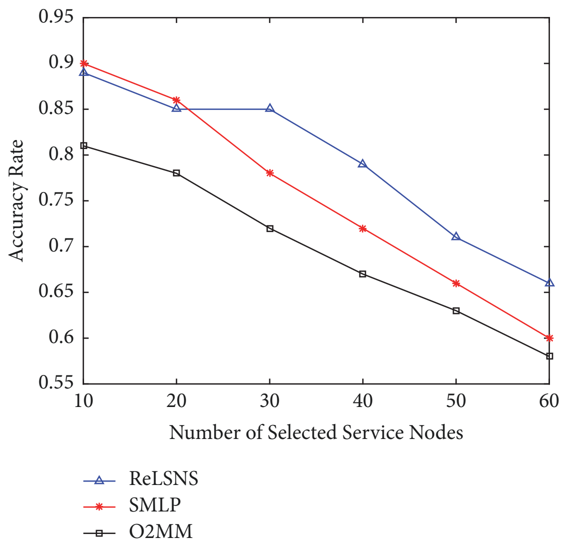


Figure 7: Accuracy of different algorithms.

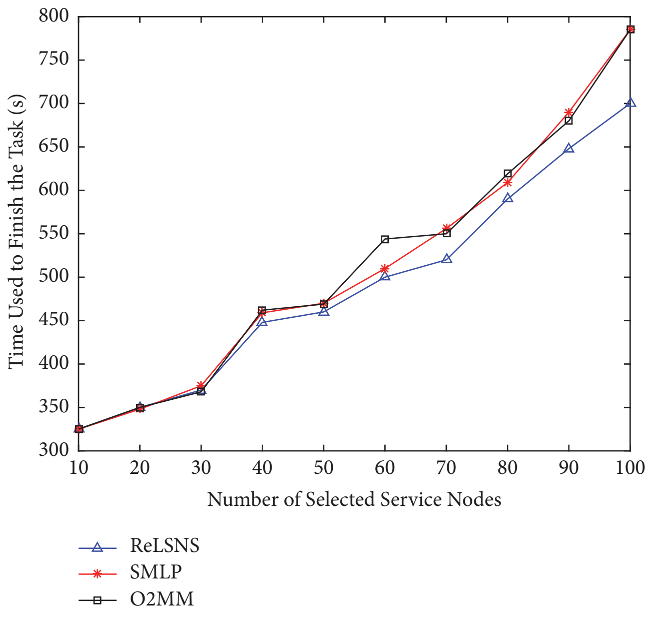


Figure 8: Timeliness of different algorithms.

successfully acquire the sensing task in the simulation and afterward obtain the timeliness of the selected node set. Then, we can obtain the average time taken by the selected nodes to complete the task under different required numbers of nodes; a comparison with other algorithms is shown in Figure 8.

In Figure 8, when the number of selected nodes is small, the time required by each algorithm to complete the task is basically the same. However, when the number of selected nodes increases, the method proposed in this paper uses much less time than the other two algorithms. For example, when we need 100 service nodes, according to Figure 8, ReLSNS requires 9.55% less time compared to the other two methods. ReLSNS can use the locations of selected service nodes as an important reference factor, which can guarantee that the selected nodes are closer to the location so that devices can obtain the sensing data. In this way, it can shorten the time required for nodes to perform the task and ensure that the set of selected service nodes can meet the requirement of high timeliness.

4.4. Generalization of Experimental Results. Based on the simulation tests of selected nodes' accuracy rate and the average time required for task execution, we can arrive at the following conclusions. First, when the number of tasks is small, the performance of the method based on region feature proposed in this paper is basically the same as the social relation models. But with the increasing of the number of selected service nodes, the method in this paper is superior to the other two algorithms regarding accuracy and timeliness.

5. Conclusion

In this paper, a low-cost service node selection method based on region feature vector is proposed to solve the problem that the existing service node selection algorithms do not consider: the association between sensing task and region. The method uses Voronoi polygons as the basis and constructs an eigenvector of each Voronoi polygon region. When the system releases a sensing task, this method can help task requestors to compute matching degrees between the task and region task feature vectors and select regions according to the overall score of each region. And finally, we can get regions with a high matching degree of task characteristics which can then continue the progress of service nodes selection in these regions.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Science and Technology Program of Shenzhen (JCYJ20170816100939373) and partly by Key Projects of Science and Technology Plan in Fujian Province under Grant 2016H0029.

References

[1] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, article 7, 2015.

[2] J. An, X. Gui, Z. Wang, J. Yang, and X. He, "A Crowdsourcing Assignment Model Based on Mobile Crowd Sensing in the Internet of Things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 358–369, 2015.

[3] B. Guo, Z. Yu, X. Zhou, and D. Zhang, "From participatory sensing to Mobile Crowd Sensing," in *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PERCOM WORKSHOPS 2014*, pp. 593–598, March 2014.

[4] Y. Wang, Q. Deng, W. Liu, and B. Song, "A Data-Centric Storage Approach for Monitoring System of Large-Scale Smart Grid," in *Proceedings of the 2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pp. 1–5, Shanghai, China, September 2012.

[5] F. Guidi, A. Guerra, and D. Dardari, "Personal mobile radars with millimeter-wave massive arrays for indoor mapping," *IEEE Transactions on Mobile Computing*, vol. 15, no. 6, pp. 1471–1484, 2016.

[6] K. Xing, Z. Wan, P. Hu et al., "Mutual privacy-preserving regression modeling in participatory sensing," in *Proceedings of the IEEE INFOCOM 2013 - IEEE Conference on Computer Communications*, pp. 3039–3047, Turin, Italy, April 2013.

[7] H. Chen, H. Jin, L. Guo, S. Wu, and T. Gu, "Audio-on-demand over wireless sensor networks," in *Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service (IWQoS)*, pp. 1–9, Coimbra, Portugal, June 2012.

[8] J. Zaman and W. De Meuter, "DisCoPar: Distributed components for participatory campaigning," in *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 160–165, St. Louis, MO, March 2015.

[9] Y. Xu and S. Hu, "QMapper," in *Proceedings of the the 22nd International Conference*, pp. 211-212, Rio de Janeiro, Brazil, May 2013.

[10] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma, "PRISM: platform for remote sensing using smartphones," in *Proceedings of the 8th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '10)*, pp. 63–76, June 2010.

[11] J. Oyekan, . Dongbing Gu, and . Huosheng Hu, "Visual Imaging of Invisible Hazardous Substances Using Bacterial Inspiration," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 5, pp. 1105–1115, 2013.

[12] R. Cheng, X. Xie, M. L. Yiu, J. Chen, and L. Sun, "UV-diagram: A Voronoi diagram for uncertain data," in *Proceedings of the 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pp. 796–807, Long Beach, CA, USA, March 2010.

[13] I. J. Vergara-Laurens, D. Mendez, and M. A. Labrador, "Privacy, quality of information, and energy consumption in Participatory Sensing systems," in *Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 199–207, Budapest, Hungary, March 2014.

[14] J. Jin, L. Rothrock, P. L. McDermott, and M. Barnes, "Using the Analytic Hierarchy Process to Examine Judgment Consistency in a Complex Multiattribute Task," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 5, pp. 1105–1115, 2010.

[15] M. Bernasconi, C. Choirat, and R. Seri, "The analytic hierarchy process and the theory of measurement," *Management Science*, vol. 56, no. 4, pp. 699–711, 2010.

[16] S. Hachem, A. Pathak, and V. Issarny, "Probabilistic registration for large-scale mobile participatory sensing," in *Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications (PerCom 2013)*, pp. 132–140, San Diego, CA, March 2013.

[17] M. Mun, S. Reddy, K. Shilton et al., "PEIR, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 55–68, June 2009.

[18] R. Yu, X. Xia, S. Liao, and X. Wang, "A Location Prediction Algorithm with Daily Routines in Location-Based Participatory Sensing Systems," *International Journal of Distributed Sensor Networks*, vol. 2015, pp. 1–12, 2015.

[19] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes, "ICrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 2010–2022, 2016.

[20] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "ActiveCrowd: A Framework for Optimized Multitask Allocation in Mobile Crowdsensing Systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, 2017.

[21] D. Mendez, M. Labrador, and K. Ramachandran, "Data interpolation for participatory sensing systems," *Pervasive and Mobile Computing*, vol. 9, no. 1, pp. 132–148, 2013.

[22] Y. Tong, L. Wang, Z. Zhou et al., "Flexible online task assignment in real-time spatial data," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1334–1345, 2017.

[23] D. Mendez and M. A. Labrador, "Density Maps: Determining Where to Sample in Participatory Sensing Systems," in *Proceedings of the 2012 Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing (MUSIC)*, pp. 35–40, Vancouver, Canada, June 2012.

[24] X. Zhang, Z. Yang, Y. Liu, and S. Tang, "On Reliable Task Assignment for Spatial Crowdsourcing," *IEEE Transactions on Emerging Topics in Computing*, pp. 1-1.

[25] X. Zhang, Z. Yang, Y. Gong, Y. Liu, and S. Tang, "Spatial Recruiter: Maximizing sensing coverage in selecting workers for spatial crowdsourcing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, 2017.

[26] J. Howe, "The rise of crowdsourcing," *Wired Magazine*, vol. 14, no. 6, pp. 176–183, 2006.

[27] W. A. Chaovalitwongse, Y.-J. Fan, and R. C. Sachdeo, "On the time series K-nearest neighbor classification of abnormal brain activity," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 37, no. 6, pp. 1005–1016, 2007.

[28] W. Cai, S. Chen, and D. Zhang, "A multiobjective simultaneous learning framework for clustering and classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 21, no. 2, pp. 185–200, 2010.

[29] B. Kao, S. D. Lee, F. K. F. Lee, D. W. Cheung, and W.-S. Ho, "Clustering uncertain data using voronoi diagrams and R-tree index," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 9, pp. 1219–1233, 2010.

[30] C. H. Wu, K. C. Lee, and Y. C. Chung, "A Delaunay Triangulation based method for wireless sensor network deployment," *Computer Communications*, vol. 30, no. 14-15, pp. 2744–2752, 2007.

[31] X. Li and W. Zhu, "Capacity Regions for General Ad Hoc Networks," in *Proceedings of the 2010 6th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pp. 1–4, Chengdu City, China, September 2010.