Range Query Techniques in Jail Management

Yuan-Ko Huang and Lien-Fa Lin
Department of Information Communication, University of KaoYuan, Taiwan, R.O.C.
Email: {huangyk, lienfa}@cc.kyu.edu.tw

***Abstract*—In recent years, developing processing techniques for spatio-temporal databases is a very hot topic. There are many real-life applications related to spatio-temporal queries, such as geographic information, traffic monitoring system, and mobile information system. In this paper, we focus on how to process the range query in jail management. Different from the previous research, the moving objects are the convicts and the jail guards in jail management. Moreover, the positioning technique used in jail management is not the GPS technique, but is the sensor network technique. This results in a greater positioning error. The jail management system developed in this paper tries to manage the convicts and the guards with uncertain location information. With the location information, we will design efficient algorithm to process the range query on the convicts and the jail guards. Benefiting from the jail management system, we can efficiently manage the convicts and the quards so as to provide useful information to the jail manager.**

***Index Terms*—spatio-temporal databases, range query, jail management, uncertain location information**

I. Introduction

Range query is one of the fundamental queries in the field of spatio-temporal databases. A range query can be used to find the objects that are inside a spatial range R. An example of a range query is "monitoring the surrounding enemy warships of a warship so as to avoid the possible attack". Previous studies [1]-[6] focus on processing the range queries on *outdoor* moving objects whose locations are obtained through the GPS technique. In this paper, we take the *indoor* moving objects into consideration in processing the range queries. We apply the query processing technique of range query to jail management, in which the indoor moving objects are classified into the convicts and the jail guards.

For each indoor moving object, its location cannot be obtained through the GPS technique. Instead, the indoor moving object is monitored by taking advantage of some indoor positioning technique, such as the sensor-based tracking system. As the location of each object is determined by the sensor network, the uncertainty of object's location is much higher than that of object monitored by the GPS technique. This would incur a higher cost of processing the range query in jail management. In the following, we will discuss the difficulties that need to be addressed in processing the range query in jail management.

Manuscript received June 20, 2013; revised August 29, 2013.

The first difficulty is how to quantify the uncertainty of moving objects (including the convicts and the jail guards). As the indoor moving objects are monitored in the sensor network, the uncertainty of objects is dominated by the number of sensors. Due to the limited number of sensors, different deployment strategies for the sensor network will result in different uncertainty of object locations. As such, developing a good sensor deployment is important to effectively reduce the uncertainty of object locations.

The second difficulty is how to develop an efficient index for managing the indoor moving objects. The existing index structures are well-designed for the outdoor moving objects that have precise location information and for one type of objects only. In this paper, we consider two types of indoor moving objects with uncertainty when the range query is processed. Therefore, a new index structure is needed to manage such indoor moving objects.

The third difficulty is how to design an efficient range query processing algorithm to find the convicts or the jail guards that are inside the query range. Due to the movement of indoor objects, the query result would change with time (that is, the query result is dependent on the current location of the query object). This makes the execution of the range query more complicated. We will analyze the characteristic of the range query so as to develop the query processing algorithm with better performance.

The last difficulty is how to reasonably give each indoor moving object a probability of being inside the query range (i.e., the probability of object being the query result). As the indoor objects move with uncertain location, there exist some objects that are possible to be inside the query range (that is, these objects are possible answers for the range query). When the number of possible answers increases, the situation becomes very hard to choose the most appropriate objects. Therefore, a probability model must be built to reasonably provide probabilistic answers to the range query.

To sum up, the major contributions of this paper are as follows.

- We design an effective mechanism to quantify the uncertainty of the convicts and the jail guards in jail management.
- We develop an efficient index structure to manage the convicts and the jail guards with uncertainty.
- We propose a processing algorithm to efficiently answer the range query.

doi: 10.12720/lnit.1.4.165-169

- We develop a reasonable probability model to quantify the possibility of each indoor object being the query result.

The remainder of this paper is organized as follows. In Section II, the uncertain model is introduced. Section III describes the grid index used in query processing. In Section IV, we present the proposed algorithm for answering the range query. Section V presents the designed probability model. Finally, Section VI concludes this paper.

II. Uncertain Model

To monitor the convicts and the jail guards in the jail, we need to design a sensor deployment to construct the sensor network. In the constructed sensor network, each indoor object moves with uncertain location as time progresses. In the following, we introduce the sensor deployment strategy, and then present the uncertain model for the indoor moving objects.

A. Sensor Deployment

Different sensor deployment strategy would result in different covering region of sensor network, which incurs a significant difference between the uncertainties of objects' locations. In order to effectively reduce the cost of constructing the sensor network, we apply a grid deployment strategy to build the sensor network covering the jail. Let us use Fig. 1 to illustrate the grid deployment. In this example, the sensor network is constructed by the 9 sensors. As each sensor has the same sensing radius r, the distance between each sensor is equal to r. It is easy to implement a grid-based sensor network in the jail, and thus the constructing cost can be effectively reduced.

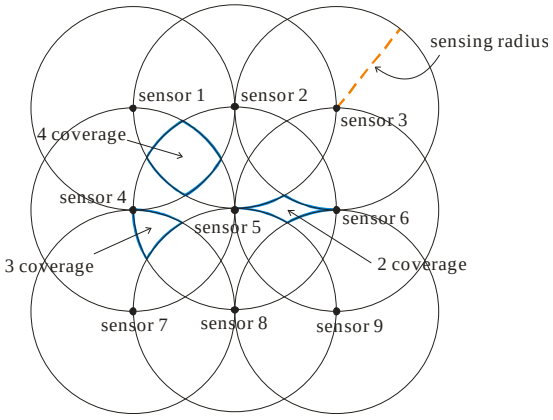


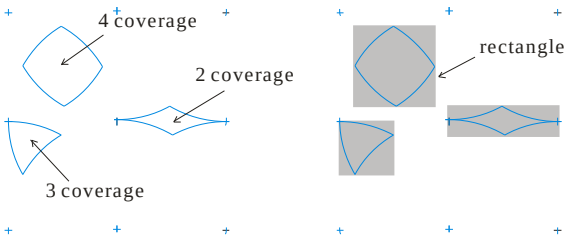
Figure 1. Grid deployment.

As the sensing technique of the sensor network cannot precisely monitor the indoor moving objects, the locations of the convicts and the jail guards in the jail are uncertain. As for the uncertain extent of each object, it is dependent on the covering region of sensor network. Using the above grid deployment, each object is monitored by at least 2 sensors (i.e., the 2 coverage), and is monitored by at most 4 sensors (i.e., the 4 coverage). That is, each object in the jail belongs to one of the 2 coverage, the 3 coverage, and the 4 coverage (depicted as the blue regions shown in Fig. 1).

B. Object Uncertainty

For each convict and jail guard, he/she is monitored by at least 2 sensors. However, due to the inherent uncertain property of sensors, the convict and the jail guard are imprecisely tracked. That is, we can know the possible locations of each object, instead of its precise location. Let us consider again the example in Fig. 1. For the 2 coverage region monitored by the sensor 5 and the sensor 6, all the points inside it could be the actual location of object. Similarly, the 3 and 4 coverage regions represent the possible locations of object. We term the coverage regions *the uncertainty regions*.

When the range query is processed, different locations of object would result in different query results. As such, all the points inside the uncertainty region need to be considered in processing the range query. However, as we can see from Fig. 2(a), the shape of the uncertainty regions (including the 2, 3, and 4 coverage regions) is irregular, which makes it hard to quantify the uncertainty of object. As a result, we adopt a minimum rectangle to enclose the uncertainty region as shown in Fig. 2(b), and utilize it to approximate the uncertainty region in query processing. We term the minimum bounding rectangle *the approximate region*.



(a) uncertainty region (b) approximate region

Figure 2. Uncertainty model.

III. Grid Index Structure

To efficiently manage the convicts and the jail guards in the jail, a well-designed index structure is required. However, the uncertainty on object locations makes the design and the maintenance of index structure much harder. In this section, we design a grid index to store the location information of the convicts and the jail guards. In the following, we first describe the designed grid index. Then, we illustrate how to manage the objects using the grid index. Finally, we discuss the update operation of the grid index.

A. Grid Index Structure

In our model, the uncertainty of objects' locations is represented as an approximate region with rectangular shape. We use a grid index to efficiently manage the approximate regions. The data space covering the entire jail is first divided into $N*N$ grid cells, and then each grid cell stores information of the convicts, the jail guards, and the sensor inside it. Consider the example in Fig. 3. There are 3 convicts c1 to c3 and 2 jail guards p1 to p2 moving in the jail. The entire space of jail is divided into 3*3 grid cells and monitored by 9 sensors s1 to s9. That is, each

sensor is located at the center of a grid cell and responsible for monitoring this cell. In the sequel, we describe the data structures used for storing information of the sensors, the convicts, and the jail guards.

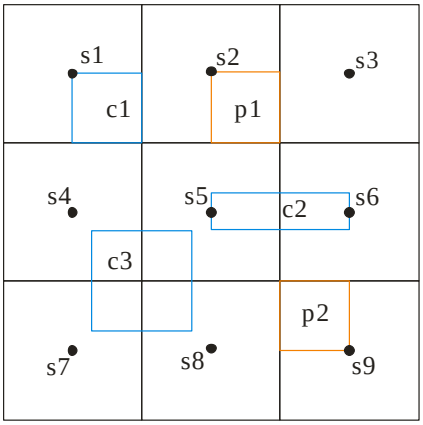


Figure 3. Grid index structure.

B. Data Structures

Three tables, T_c, T_p, and T_s, are used to keep the information of grid index. The first table T_c stores the information of the convicts in the jail. The information contains (1) c-id: the convict id, (2) s-list: the sensors that monitor this convict currently, (3) t_c: the time point at which this convict begins to be monitored, and (4) c-data: the basic data of this convict (such as age, height, and weight). The second table T_p maintains the information of the jail guards. The information includes (1) p-id: the guard id, (2) s-list: the sensors currently monitoring the jail guard, (3) t_p: the time point when the guard is monitored, and (4) p-data: the basic data of this guard. The last table T_s keeps the information of the sensors. The information is (1) s-id: the sensor id, (2) s-radius: the sensing radius of this sensor, (3) c-list: the convicts monitored by this sensor, and (4) p-list: the jail guard monitored by this sensor.

C. Index Update

As the convicts and jail guards in the jail move with time, the grid index needs to be updated so as to precisely store the information of objects. In this section, we design an efficient mechanism to rapidly update the location information of objects. Let us use the example in Fig. 4 to illustrate how to efficiently process the updates of object locations. In this example, at time 3 the convict c1 moves from the region monitored by the sensor s1 to the region monitored by the sensor s4. Also, the jail guard p1 moves from the covering region of the sensor s2 to the covering region of the sensor s3. Due to the movement of the convict c1 and the jail guard p1 at time 3, the following 5 tables of the grid index need to be updated.

- Updating the table T_p of the jail guard p1: when the jail guard p1 moves, the sensor s5 cannot monitor p1 while the sensor s6 begins to monitor p1. Therefore, its table T_p is changed from (p1, {s2, s3, s5}, 2, p-data) to (p1, {s2, s3, s6}, 3, p-data).
- Updating the table T_c of the convict c1: once the convict c1 moves, the sensor s2 cannot monitor c1 but the sensor s5 can monitor c1. As such, the table T_c is updated from (c1, {s1, s2, s4}, 1, c-data) to (c1, {s1, s4, s5}, 3, c-data).
- Updating the table T_s of the sensor s2: due to the movement of the convict c1, the table T_s of the sensor s2 needs to be updated. It is changed from (s2, 60, {c1}, {p1}) to (s2, 60, {null}, {p1}).
- Updating the table T_s of the sensor s5: because the locations of the convict c1 and the jail guard p1 are changed, the table T_s of the sensor s5 is updated from (s5, 60, {c2, c3}, {p1}) to (s5, 60, {c1, c2, c3}, {p1}).
- Updating the table T_s of the sensor s6: due to the movement of the jail guard p1, the table T_s of the sensor s6 is updated. It is changed from (s2, 60, {c1}, {p1}) to (s2, 60, {null}, {p1}).

Benefiting from the above update mechanism, the table inform that has been affected by the object movement can be quickly determined and updated. As such, the grid index can be used to efficiently manage moving objects so as to facilitate the range query processing.

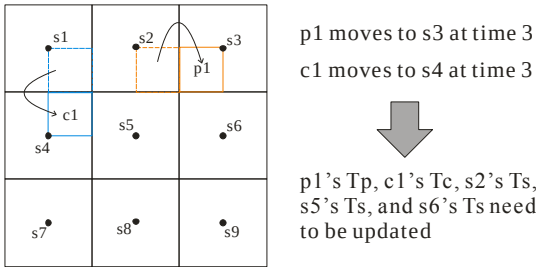


Figure 4. Update mechanism.

IV. Query Processing Algorithm

In this paper, we focus on processing the range query on indoor moving objects in the jail. Given a spatial region R in the jail, the range query retrieves the convicts and the jail guards that are inside R currently. We develop an efficient algorithm combined with the grid index to answer the range query. Our algorithm consists of six steps, which are discussed as follows.

- Step 1: determining the cells of the grid index that intersect with the spatial region R. This is because only the convicts and the jail guards enclosed by these grid cells are possible to be inside R (i.e., the query result). In other words, the convicts and the jail guards enclosed by the cells not intersecting R can be safely pruned. After this step, a set G of grid cells is obtained.
- Step 2: for each grid cell in G, accessing the table T_s of its corresponding sensor. By looking up c-list of T_s, we can know the monitored convicts and they could be inside R. After this step, a set C of the convicts is obtained.
- Step 3: for each grid cell in G, accessing the table T_s of its corresponding sensor. By looking up p-list of T_s, we determine the monitored jail guards that are possible to be inside R. After this step, a set P of the jail guards is obtained.

- Step 4: for each convict in C, accessing its table T_c to obtain the sensors currently monitoring this convict. If there exists at least one sensor whose corresponding cell does not intersect R, then this convict can be pruned (i.e., this convict is removed from C). After this step, the set C is modified.
- Step 5: for each jail guard in P, accessing its table T_p to know the sensors monitoring this jail guard. If there exists one sensor whose corresponding cell does not intersect R, this jail guard is removed from the set P. After this step, the set P is updated.
- Step 6: returning the convicts in C and the jail guards in P as they are possible to be the query results. Then, each convict and jail guard is given a probability of being inside the region R (which will be discussed later).

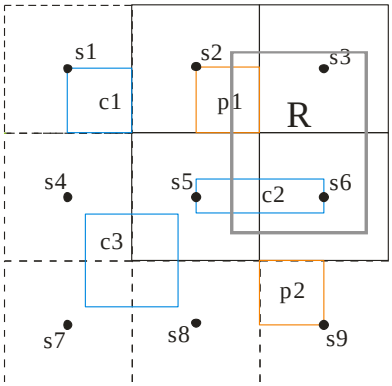


Figure 5. Range query.

Fig. 5 gives an example of how to answer the range query using the proposed algorithm. In this example, the query range R is depicted as a gray rectangle. There are 3 convicts c1 to c3 and 2 jail guards p1 to p2 in the jail monitored by the 9 sensors s1 to s9. The process of the algorithm is described as follows.

- Step 1: in the grid index, only the grid cells for the sensors s2, s3, s5, and s6 intersect with the region R. As such, the set G = {s2, s3, s5, s6} after this step.
- Step 2: accessing the table T_s of the sensors s2, s3, s5, and s6 to check its c-list. For the sensor s2, its c-list = {c1}, and thus c1 is added into the set C. Similarly, for the sensor s5 (s6), its c-list = {c2, c3} ({c1, c2, c3}). After this step, the set C = {c1, c2, c3}.
- Step 3: looking up the table T_s of the sensors s2, s3, s5, and s6 to obtain its p-list. For the sensor s2, its p-list = {p1}, and then p1 is added into the set P. As for the sensor s6, its p-list = {p2}. After this step, the set P = {p1, p2}.
- Step 4: accessing the table T_c of the convicts c1, c2, and c3 to check its s-list. For the convict c1, its s-list = {s1, s2, s4}. As the grid cells for the sensors s1 and s4 do not intersect the region R, the convict c1 is removed from C (i.e., C is updated to {c2, c3}). Similarly, the convict c3 is removed from C because the cells for the sensors s4, s7, and s8 do not intersect R. Finally, the set C = {c2}.
- Step 5: accessing the table T_p of the jail guards p1 and p2 to obtain its s-list. For the jail guard p2, only the corresponding cell for the sensor s6 intersects with R, and thus the jail guard p2 is removed from the set P (i.e., P = {p1}).
- Step 6: giving the convict c2 in C and the jail guard p1 in P a probability to quantify the possibility of being inside the region R.

V. Probability Model

The probability model is designed to reasonably give each convict and jail guard in the jail a probability which quantifies the possibility of being the query result. Based on the probability, the server can provide more useful information for the user to decide the most appropriate result. The main idea of this probability model is to quantify the possibility of each object based on the proportion of the overlapping area between the region R and its approximate region to the area of its approximate region. The following two parameters, P(c) and P(p), are designed to compute the probabilities of the convict c and the jail guard p, respectively.

- The probability P(c) of the convict c: $P(c) = A(R \cap c) / A(c)$, where A(c) refers to the area of the approximate region of the convict c and $A(R \cap c)$ is c's overlapping area between the approximate region and R.
- The probability P(p) of the jail guard p: $P(p) = A(R \cap p) / A(p)$, where A(p) refers to the area of the approximate region of the jail guard p and $A(R \cap p)$ is the overlapping area between p's approximate region and R.

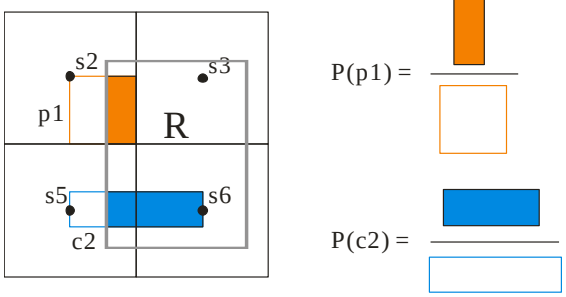


Figure 6. Probability calculation.

Let us use Fig. 6, which continues the example in Fig. 5, to illustrate how the probabilities of the convict and the jail guard are computed. In this example, the set C = {c2} and the set P = {p1}. To calculate the probability of the convict c2 being inside the range R, the area A(c2) of the approximate region of c2 needs to be determined first, which is shown as the blue rectangle in Fig. 6(a). Then, the area $A(R \cap c2)$ of the overlapping region between c2's approximate region and the range R is calculated as the blue solid rectangle. With the two areas A(c2) and $A(R \cap c2)$, the probability P(c2) of the convict c2 is computed as $A(R \cap c2) / A(c2)$. Similarly, the area A(p1) of the approximate region of the jail guard p1 is represented as the orange rectangle and the area $A(R \cap p1)$ of the overlapping region is shown as the orange solid

rectangle. Finally, the probability P(p1) of the jail guard p1 is computed as $A(R \cap p1) / A(p1)$.

VI. Conclusions

In recent years, many studies focused on processing techniques in spatio-temporal databases. In this paper, we tried to apply the query processing techniques to the jail management, in which objects move indoor with uncertainty. We first designed an effective mechanism to quantify the uncertainty of the convicts and the jail guards moving in the jail. Then, we developed a grid index to efficiently manage the convicts and the jail guards with uncertainty. We also proposed an efficient algorithm to answer the range query. Finally, a reasonable probability model was designed to quantify the possibility of each indoor object being the query result.

Acknowledgment

This work was supported by National Science Council of Taiwan (R.O.C.) under Grants NSC101-2119-M-244-001 and NSC102-2119-M-244-001.

References

[1] J. Zhang, M. L. Zhu, D. Papadias, Y. F. Tao, and D. L. Lee, "Location-based spatial queries," in *Proc. ACM SIGMOD International Conference on Management of Data*, June 9-12, 2003, pp. 443-454.

[2] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2000, pp. 331-342.

[3] D. V. Kalashnikov, S. Prabhakar, S. Hambrusch, and W. Aref, "Efficient evaluation of continuous range queries on moving objects," in *Proc. International Conference on Database and Expert Systems Applications*, September 2-6, 2002, pp. 731-740.

[4] Y. Tao and D. Papadias, "Time parameterized queries in spatio-temporal databases," in *Proc. ACM SIGMOD*, Madison, Wisconsin, 2002, pp. 322-333.

[5] M. F. Mokbel, X. Xiong, and W. G. Aref, "Sina: Scalable incremental processing of continuous queries in spatio-temporal databases," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2004, pp. 623-634.

[6] K. Mouratidis, M. Hadjieleftheriou, and D. Papadias, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2005, pp. 634-645.

Yuan-Ko Huang received the ME and PhD degrees in computer science from the University of Cheng-Kung, Taiwan, R.O.C. in 2004 and 2009, respectively. His research interests include databases, mobile computing, and sensor networks

Lien-Fa Lin is a PhD student in the Department of Information Management, NCTU, Taiwan, R.O.C. His research interests include databases, social networks, and sensor networks.

