Evaluation of Range Queries with Predicates on Moving Objects

Mitzi McCarthy, Zhen He and X. Sean Wang

Abstract—A well-studied query type on moving objects is the continuous range query. An interesting and practical situation is that instead of being continuously evaluated, the query may be evaluated at different degrees of continuity, e.g. every 2 seconds (close to continuous), every 10 minutes or at irregular time intervals (close to snapshot). Furthermore, the range query may be stacked under predicates applied to the returned objects. An example is the count predicate that requires the number of objects in the range to be at least γ. The conjecture is that these two practical considerations can help reduce communication costs. We propose a safe region-based solution that exploits these two practical considerations. An extensive experimental study shows that our solution can reduce communication costs by a factor of 9.5 compared to an existing state-of-the-art system.

Index Terms—Spatial databases, temporal databases, query processing

1 Introduction

Recently, there has been a rapid increase in the number of geolocation-aware devices. These devices give rise to a new class of applications called location-based services (LBS). An interesting problem arising in LBS is the continuous evaluation of spatial queries.

There has been extensive research into developing techniques for answering both continuous and snapshot range queries on moving objects. These studies can be categorized as those focused mainly on minimizing computation time [1], [4], [7], [9], [13], [15] or minimizing communication costs [6], [8], [12], [16] incurred by mobile devices reporting their location to a central server. Our work falls in the latter category. Like other work in this category, we also use the concept of safe regions to reduce communication costs. Data objects (the mobile devices being queried) only report their locations when they move out of their rectangular safe region or are probed by the server. This prevents the need for objects to constantly report their location.

We extend the traditional continuous range query in the following three ways:

- aggregation predicates
- answer queries which have different degrees of continuity, covering the entire spectrum from continuous to snapshot queries
- moving range queries

- *M. McCarthy and Z. He are with the Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, VIC, Australia.*
  E-mail: {m.mccarthy, z.he}@latrobe.edu.au
- *X. S. Wang is with the School of Computer Science, Fudan University, Shanghai, China.*
  E-mail: xywangCS@fudan.edu.cn

We describe and motivate each of these three extensions to the continuous range query.

1.1 Range with aggregation predicates

In practice, more operations may be applied to the set of objects returned by the range query before the results are presented to the user. As an example, a count predicate may be applied to the range query result. More specifically, given a range query q and a set of data objects S, we may issue the query: if $COUNT(P) \geq \gamma$ then return true else false, where $P = \{s|$ location of s is within the range of q, $s \in S\}$ and γ is a user defined threshold. We can replace $COUNT(P) \geq \gamma$ with a predicate on one of the aggregation functions MIN, MAX, SUM or AVERAGE. We call queries of this type *range queries with an aggregation predicate* (RAP). Note RAP can be used to answer the traditional range query by setting the predicate $COUNT(P) \geq 0$ and returning all objects within the range if the predicate evaluates to true.

We use two examples to illustrate the use of RAP queries.

Example 1: An advertising company operating a dynamic electronic billboard. At any given time, we need to select a small car or family sedan to advertise. Suppose age was the most important factor for choosing which car type to display. We can then specify that if the average age of people within 100 meters of the billboard is at least 30, then advertise the family sedan else the small car. This RAP query (q_1) can be written as follows: return true if $AVERAGE(P.age) \geq 30$ else return false, where $P = \{s|$ location of s is within 100 meters of $q_1\}$. If the query returns true, then advertise the family sedan else the small car.

Example 2: A query q_2 alerts a taxi driver if there are likely more than 5 people wishing to use the service within a specified range of q_2. This can be written as follows: return true if $COUNT(P) > 5$ else return false, where $P = \{s|$ location of s is within range of $q_2\}$.

Motivation for a new approach. RAP queries cannot be handled efficiently by existing safe region-based work [6], [8], [12], since the existing work would answer RAP queries by first executing a range query and then applying the aggregation predicate on all returned objects. This approach would result in more communication than necessary since RAP queries are concerned with *group* properties rather than *individual* properties. Suppose there are 10,000 objects and our RAP query wants to know if there are at least two objects within the query range. The existing work would precisely determine whether each of the 10,000 objects is within range and then check if the number of objects within range is at least 2. This is clearly acquiring more information than necessary. In contrast, our approach only tries to find two objects (any two) within query range.

1.2 Queries with different degrees of continuity

Queries can be evaluated at different frequencies. There is a trade-off between frequency of query evaluations and communication costs. More frequent evaluations means there is less chance of missing a monitored event, but also means higher communication costs. In most existing work on traditional range queries, queries are either continuous [6], [8], [12] or snapshot [1], [4], [14], [16]. We note these are actually the two extremes among a spectrum of possibilities. For example, queries can be answered every 2 seconds (close to continuous), or every 10 minutes or at irregular time intervals (close to snapshot). In this paper, we support the entire spectrum of possibilities. Our system seamlessly supports a mixture of queries with different degrees of continuity. Our solution reactively adapts to these different degrees of continuity.

Motivation for a new approach. Most systems designed to answer snapshot range queries either use the constant update model [16], the linear update model [1], [4] or the recursive motion function model [14]. The linear update and recursive motion function models require fewer updates than the constant model. However, all three models result in a lot of communication since they effectively track the movement of the objects at all times. These models are not adaptive to when and where queries occur, and therefore result in higher communication costs than necessary. For example, objects moving in a region which is never queried will still need to have their movements tracked by the server.

Most systems designed to answer continuous range queries use safe regions [6], [8], [12]. However, they allocate unnecessarily small safe regions for queries that require infrequent answers because their safe regions ensure query answers are known at all times. If a query only needs an answer every 10 minutes, then it is better to assign nearby objects larger safe regions and deal with the consequences (more object location reporting) when it comes time to answer the query. We develop a reactive safe region assignment algorithm which considers the frequency at which query answers are required.

1.3 Moving queries

In many situations, the queries will be moving rather than stationary. This could be the case for both the examples above. In Example 1, the electronic billboard can be mounted on a bus or tram. In Example 2, the queried region can be around the taxi itself.

Motivation for a new approach. Existing work [6], [8], [12] for handling continuous traditional range queries using safe regions assumes queries are stationary. They would handle moving queries by continual query insertion and deletion which requires the moving query object to continuously report its location. In contrast, we assign safe regions to query objects, thereby reducing the frequency that moving query objects report their locations.

1.4 Our overall approach

Our approach consists of two main parts. First, we assign safe regions to objects and queries and second, we answer RAP queries given the current set of object/query safe regions. We call the first part safe region assignment and the second query evaluation.

There is a trade-off between larger safe regions which give objects more freedom to move versus smaller safe regions which benefit query evaluation because more precise location information is known. In order to address this trade-off, we take a *reactive approach to assigning safe regions*. Objects which frequently move out of their safe regions will have their safe regions enlarged. In contrast, objects which are frequently asked to report their location during query evaluation will have them shrunk. The shape of the safe regions is also adaptive to the query and object movements. This safe region assignment approach is very general in that it can be used for any spatial query.

Evaluating a set of RAP queries is an interesting problem since, given a set of object/query safe regions, we often have to choose which objects/queries will report their precise location to the server (called exposure). In an example RAP query which requires at least two objects within the query region, we are happy to find *any* two objects. Hence, we need to determine which object to target first. We may believe the query is false, in which case we need to efficiently determine that at most one object is within the query range. Making the right choices can result in lower communication costs. In this paper, we develop an information theoretic approach and a rank-based heuristic approach for choosing which objects to expose during query evaluation. These two approaches provide almost identical amounts of location exposures. However, the rank-based approach incurs lower execution time whilst the information theoretic approach is more generic in that the same formulation can be used with any predicate.

We have conducted experiments to determine the merit of our proposed algorithms against the state-of-the-art safe region-based query monitoring algorithm (SRB) [6] designed for continuous range queries. The results show that for moving RAP queries with different degrees of continuity, we significantly outperform SRB for all settings tested by up to a factor of 9.53 in terms of communication costs whilst incurring similar computational costs.

We quantify the benefits of using our approach to handle each of the three extensions to the traditional continuous range query in isolation as follows: 1) our approach incurs up to 26% less communication costs by exploiting the aggregation predicate of RAP queries as opposed to range queries without the aggregation predicate; 2) exploiting varying degrees of continuity allows us to reduce communication cost by up to a factor of 5.23 compared to SRB; and 3) using safe regions around *moving* queries results in answering up to 63% of moving queries without any communication costs. Overall, our experiments show our approach is effective in saving communication costs for a wide variety of usage scenarios.

In summary, we make the following three main contributions. 1) We propose an approach that can take advantage of any combination of the following query characteristics to minimize communication costs: RAP queries; queries of varying degrees of continuity; and moving query objects. 2) We develop an information theoretic and a rank-based query evaluation algorithm and a reactive safe region assignment algorithm. 3) We conduct an extensive performance assessment of our algorithm against the existing state-of-the-art SRB algorithm.

Following the introduction, we formally define the RAP query with different degrees of continuity answering problem in Section 2. In Section 3, we provide an overview of our approach. In Section 4, we describe the query evaluation algorithms for RAP queries. We describe the algorithm used to assign safe regions for objects in Section 5. Section 6 details our experimental analysis. We discuss the related work in Section 7. Finally, in Section 8, we conclude the paper and describe directions for future work.

2 Problem Definition

In this section, we give a formal definition of range queries with an aggregation predicate (RAP).

We assume a collection O of *moving objects* is registered with a central server. These objects are modeled as moving points in the plane. An object may "appear" and "disappear" at different times, i.e., an object may come in and out of O.

We assume there is a set of data objects $S \subseteq O$ and a set of query objects $Q \subseteq O$. The two sets can overlap, that is, an object can act as a query as well as a data object for another query. (We use "query" and "query object" interchangeably when no confusion arises.) We assume a query object is never used as a data object for its own answer. A data object has an associated attribute a used by the aggregation predicates (other than COUNT). A query object q is associated with a rectangular region centered on its position (c_x, c_y) with user selected radii r_x and r_y. This defines the query range $\boldsymbol{q.R}$.

We formally define the range query with aggregation predicate (RAP).

Definition 1: Given a threshold γ, a RAP query for a query object q is denoted $RAP(q)$. **RAP(q) returns true if** $AGGREGATE(\{s.a|s \in P\})\ \theta\ \gamma$ **else returns false**, where AGGREGATE is one of COUNT, SUM, AVERAGE, MIN or MAX, θ is a comparison operator $(<, \leq, >, \geq, =)$, $s.a$ is an attribute of object s (not required for COUNT) and $P = \{s \in S|$location of s is within the range of q and $s \neq q\}$.

A simple extension to the RAP query returns the set P if $AGGREGATE(P)\ \theta\ \gamma$ is true. This is useful when the users want to know which particular objects are within range once the predicate is true.

Based on the definitions above, we define our problem as follows.

Definition 2: We assume a set of moving objects O, query objects $Q \subseteq O$ and data objects $S \subseteq O$. Each query object q is associated with a RAP. The **RAP query answering with different degrees of continuity problem** is that, at the user-required times, the server needs to precisely answer the RAP query associated with each q in Q.

The user-required times for query answering can either follow a user defined schedule, or at user's request (ad-hoc times). Note that the latter option will incur an additional communication cost because the user will need to alert the server each time an answer is needed.

The goal of this paper is to find algorithms that precisely answer the given set of RAP queries Q at user required times with minimal object location communication. Similar to [5], we only count the communication cost from the objects (and queries) to the server because our aim is to prolong the battery lifespan of the mobile devices. There are no restrictions on the frequency or regularity/irregularity of time intervals between queries, therefore supporting the entire spectrum from continuous to snapshot queries.

3 Overview of our approach

As mentioned in the introduction, this paper takes the approach of assigning safe regions around objects to minimize communication costs. A safe region $s.SR$ is defined as a rectangular area around an object s.

While an object stays within its safe region, it does not need to tell the server its current location unless the server explicitly requests a location update (called object exposure). This explicit request may be needed in order to answer queries whose answer cannot be determined precisely otherwise.

Moving queries. In our work, queries can move. To reduce the rate of location reporting, we assign safe regions to queries. This means queries only have to report their locations when they move outside of their safe region.

Queries with different degrees of continuity. We handle varying degrees of continuity of queries by using the concept of *liberal safe regions* as opposed to the traditional approach of *conservative safe regions*. Conservative safe regions used by existing algorithms such as SRB [6] build safe regions small enough to ensure that the query answer is known at all times. In contrast, liberal safe region assignment allows safe regions to grow freely and addresses the consequences during query evaluation time. During query evaluation, some exposures may be needed in order to answer the queries. This approach allows queries of varying degrees of continuity to be supported efficiently since it means the frequency of exposures during query evaluation will depend on how often the queries need to be answered.

A consequence of using liberal safe regions is that the safe region assignment algorithm needs to balance between smaller safe regions (desirable for query evaluation) and larger safe regions (harder for objects to move out). We use a reactive safe region assignment algorithm that dynamically adjusts the size and shape of the safe regions to minimize exposures during query evaluation while keeping the safe regions as large as possible. By assigning safe regions in this way, our system adjusts itself to take maximum advantage of time gaps between query answers.

Our approach for evaluating RAP queries is to take the current set of safe regions and queries which need to be answered and determine the object exposure order such that the total number of exposures is minimized and all queries are answered. This is preferable to using the existing traditional range query solutions which would first answer the range query and then apply the predicate. The often complex decision of which object to expose first is discussed in Section 4.4.

The next two sections provide more details of our approach. Section 4 explains how we evaluate queries given a set of object safe regions. Section 5 details our reactive safe region assignment algorithm.

4 Query Evaluation Approach

In this section, we describe our algorithm for evaluating RAP queries. The evaluation algorithm solves the following problem: given a set of objects inside their safe regions, and a set of RAP queries Q_t to be answered at time t, answer all queries in Q_t with minimum communication costs.

4.1 Evaluation algorithm at a high level

At a high level, the evaluation algorithm first tries to evaluate queries without exposing the current locations of any objects. If some query answers are unknown without exposures, we then ask certain objects to expose (report) their exact locations. This second step (described in Section 4.4) requires careful planning in order to minimize communication costs.

Algorithm 1 High-level pseudo code for the evaluation algorithm.

```
Input: $Q_t$: set of queries that need to be answered at the current time $t$ and $S$: all data objects
1: Let $UQ$ store the set of queries for which the result is unknown which is initialized to $Q_t$
2: $UQ$ = answer_queries($UQ$, $S$) {answers all queries that do not need exposure using Algorithm 2}
3: for each query $u$ in $UQ$ do
4:     if query $u$ is still unanswerable then
5:         Request query $u$ to expose its current location
6:         Set $u.SR$ to a point which is represented by the reported location of $u$
7:         Mark that $u.SR$ "should shrink" from its previous size
8:     end if
9: end for
10: $UQ$ = answer_queries($UQ$, $S$) {Algorithm 2}
11: while $UQ$ is not empty do
12:     Select a data object $s$ to expose {See Section 4.4}
13:     Expose $s$
14:     Set $s.SR$ to a point which is represented by the reported location of $s$
15:     Mark that $s.SR$ "should shrink" from its previous size
16:     $UQ$ = answer_queries($UQ$, $S$){Algorithm 2}
17: end while
18: Report to the user the query results for those queries that have changed.
```

Algorithm 1 shows the straightforward algorithm for query evaluation at a high level. It works by first trying to answer as many queries as possible without exposing anything (line 2). We then expose the unanswered queries in UQ incrementally (lines 3-9), taking care to ensure we do not expose queries that are now answerable due to the exposure of prior queries in UQ. Next, we again try to answer the queries in UQ to remove any that are now answerable (line 10).

We found exposing the queries before exposing their surrounding objects almost always results in fewer total exposures. This is because exposing a query first means we reduce more uncertainty (whether each surrounding object is within query range) with a single exposure, whereas exposing an object first only reduces uncertainty regarding that single object. The exposure order of queries in UQ has little effect on the overall number of exposures because we can only avoid exposing a query if it can be answered solely by exposing other queries. These exposed queries must also be unknown objects for the query in question. We conducted an experiment using our default experimental settings to test the effect of the order of exposing queries. We found for 1000 random query exposure orders, the standard deviation of total exposures was between 0.04% and 0.30% for the different algorithms. This result shows that the query exposure order makes very little difference.

If there are still queries left unanswered after exposing queries (lines 3-10), then we answer the remaining queries by selecting one object at a time to expose (line 12) and exposing it (line 13). We aim to select the object to expose which leads to the least number of other exposures in order to answer the queries (Section 4.4). We then set the safe region to be just a point at the reported location of the exposed object (line 14). Given that the previous safe region (before it was a point) was exposed, it was probably too large. Hence, we mark s.SR as "should shrink" as an indication for the next safe region assignment task (described in Section 5). We then try to answer the queries in UQ again (line 16) and repeat this process until all the queries have been answered.

4.2 Interesting regions

In order to determine which queries are answerable without exposure (lines 2, 10 and 16 of Algorithm 1), we need to first study the various important regions around a query object. Figure 1 shows a safe region

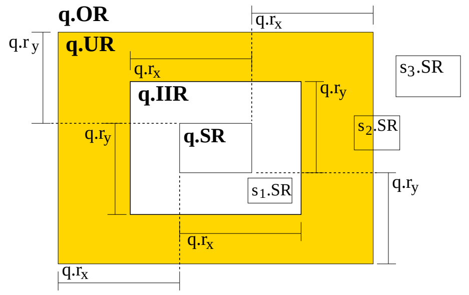


Fig. 1. A query with various interesting regions.

for a query q labeled as $q.SR$ and safe regions for three data objects labeled as $s_1.SR$, $s_2.SR$ and $s_3.SR$. Using $q.SR$ and the radii of the query range ($q.r_x$ and $q.r_y$), we can define three interesting regions that aid in answering query q. The regions are the inner influence region ($q.IIR$), the unknown region ($q.UR$) and the outer influence region ($q.OR$). We first describe the important properties of these regions and then in Theorems 1 - 3, we prove the properties.

$\boldsymbol{q.IIR}$ is the smallest rectangular region enclosing $q.SR$ in Figure 1. It tells us which objects are definitely inside $q.R$. For example, in Figure 1, we know s_1 is definitely within $q.R$ since $s_1.SR$ is fully inside of $q.IIR$. We give the name $\boldsymbol{q.OFI}$ to the set of objects whose safe regions are fully inside $q.IIR$.

$\boldsymbol{q.UR}$ is the shaded region in Figure 1. It tells us which objects have an unknown status in terms of being inside or outside $q.R$. For example, in Figure 1, we do not know if s_2 is inside or outside q.R since $s_2.SR$ intersects $q.UR$. We give the name $\boldsymbol{q.OU}$ to the set of objects whose safe regions intersect $q.UR$. We call the objects inside $q.OU$ the **unknown objects.**

$\boldsymbol{q.OR}$ is the region outside $q.UR$ in Figure 1. It tells us which objects are definitely outside $q.R$. For example, in Figure 1, we know s_3 is definitely outside $q.R$ since it is fully in $q.OR$.

We now formally define these regions and explain how we arrive at the above properties.

Definition 3: Given query q with safe region $q.SR$, let $q.IIR = \bigcap_{u \in q.SR} q.R[u]$ and $q.OR = \bigcap_{u \in q.SR} \overline{q.R[u]} = \overline{\bigcup_{u \in q.SR} q.R[u]}$ where $\overline{R}$ is the complement of a region R, and each rectangular region $R[u] = (u_x, u_y, r_x, r_y)$ is the same region $R = (c_x, c_y, r_x, r_y)$ but with center shifted to $u = (u_x, u_y)$. Furthermore, $q.UR = \overline{q.IIR \cup q.OR} = \overline{q.OR} - q.IIR$.
Note that $q.IIR$, $q.OR$ and $q.UR$ do not depend on (c_x, c_y) but only on $q.SR$, i.e., these regions are agnostic with respect to the exact location of q within its safe region. A practical method of obtaining the corners of these regions when $q.IIR \neq \emptyset$ is depicted in Figure 1.

Theorem 1: If $s.SR$ is entirely inside $q.IIR$, then s is within q.R.
Proof. From Definition 3, we have that $q.IIR$ is the intersection of all the possible positions of $q.R$. If we know that $s.SR$ lies entirely within this intersection, then it must be within $q.R$.

Theorem 2: If $s.SR$ is entirely inside $q.OR$, then s is not within $q.R$.
Proof. If $s.SR$ is entirely inside $q.OR$ then, by Definition 3, $s.SR$ is in the complement of the union of all possible positions for $q.R$. That is, $s.SR$ is not in any possible $q.R$. Therefore $s.SR$ is not in $q.R$.

Theorem 3: If $s.SR$ intersects $q.UR$, then it is unknown if s is inside $q.R$.
Proof. If $s.SR$ intersects $q.UR$, then there is a point $p \in s.SR$ such that $p \in q.UR$. From Definition 3, we have p is in the complement of the union of $q.IIR$ and $q.OR$. That is, p is neither in $q.IIR$ nor $q.OR$. Therefore, neither Theorem 1 nor Theorem 2 applies and it is unknown whether s is inside $q.R$.

Table 1 shows the symbols used to refer to the various regions and sets of objects related to a query.

4.3 Evaluation before any exposure

Having defined the various important regions around a query, we can now define the algorithm

Symbol	Meaning
SR	Safe region of a query
R	Range of a query
IIR	Inner influence region of a query
UR	Unknown region of a query
OR	Outer influence region of a query
OFI	Set of all objects with safe regions fully inside $q.IIR$
OU	Set of all objects with safe regions overlapping $q.UR$

TABLE 1
Table of symbols associated with a query.

Algorithm 2 Algorithm to answer queries before exposing any objects

Input: $UQ \subseteq Q$: queries and S: all data objects
Output: X: the set of queries unanswerable without exposing any object
Method:

```
1: Let $X$ store the queries in $UQ$ that cannot be answered without exposures
2: Initialize $X$ to $\emptyset$
3: for each query $q$ in $U$ do
4:     tentative answer = AGGREGATE($q.OFI$) $\theta$ $\gamma$
5:     if using $q.OFI \cup$ any subset of $q.OU$ instead of just $q.OFI$ will change the tentative answer in line 4 then
6:         Place $q$ inside $X$
7:     else
8:         The tentative answer is the answer to $q$
9:     end if
10: end for
11: return $X$
```

used to answer a query *without any object location exposures* (used in Lines 2, 10 and 16 of Algorithm 1). Algorithm 2 shows the pseudo code. The algorithm tries to answer the query without exposing any objects; if unsuccessful, it puts the query into the set X which is returned to Algorithm 1. The idea is to first give a tentative answer to query q using the objects in $q.OFI$ (line 4). We then need to consider if any subset of $q.OU$, when combined with the $q.OFI$ objects, can invalidate the tentative answer. If this occurs, we cannot accurately answer the query using the current information (line 5). We need to expose at least one unknown object or the query itself. Therefore, we place the query in the set of unanswerable queries (line 6). Otherwise, the query can be accurately answered using the tentative answer (line 8).

Evaluating line 5 for AVERAGE merits some discussion. The other aggregation functions are straightforward. The condition can be evaluated since the server already has stored the attribute of interest for $q.OFI$ and $q.OU$. Knowing this, we can determine the tentative answer of line 4 by considering the AVERAGE of some subsets of $q.OU \cup q.OFI$. For example, assume θ is $\geq$ and the tentative answer is true. We only need to consider at most $|q.OU|$ subsets by incrementally growing the subset one object at a time, where the object inserted is always the object with the smallest $s.a$ among objects remaining in $q.OU$.

For example, let's use the query q in Figure 1 as the input for Algorithm 2 (i.e. $UQ = \{q\}$). We assume q has a predicate of $COUNT(P) \geq 1$. In this case, $q.OFI$ only contains s_1 since it is the only object in $q.IIR$. The tentative answer for q is therefore true (line 4). We can also see that, irrespective of the location of the unknown objects, this tentative answer cannot change (line 5). Therefore, the query answer is true.

Algorithm 2 can be used for all the aggregation functions and comparison operators. For example, assume the query predicate is $SUM(P) < 100$ and there is one object s_1 in $q.OFI$ where $s_1.a = 150$. Therefore, our tentative answer is false. Now assume we have five unknown objects. One of these objects s_2 has $s_2.a = -100$. If s_2 is found to be inside $q.R$ then $SUM(\{s_1\} \cup \{s_2\}) = 50 < 100$ (line 5) which changes the tentative answer to true. Therefore, the answer to the query is unknown without further exposure.

4.4 Evaluation with exposures

In our settings, some queries (the ones returned from Algorithm 2) cannot be answered without requiring some objects to report their locations. In this situation, as described in line 12 of Algorithm 1, we need to pick an object to expose. The number of objects that need to be exposed depends on the order of exposure. This is shown by the example in Figure 2.

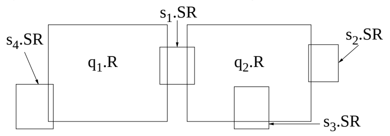


Fig. 2. Example of the importance of exposure order

In the example, there are two RAP query objects q_1 ($COUNT(P) \geq 2$) and q_2 ($COUNT(P) \geq 3$), and four data objects s_1, s_2, s_3 and s_4. We assume the exact locations of q_1 and q_2 are known, hence $q_1.R$ and $q_2.R$ are also known. All the data objects are unknown objects. Both q_1 and q_2 have a high probability of containing less than the γ required objects. The question now is which unknown object should be exposed first in order to make the query results unambiguous. The answer is not trivial since s_1, s_2 and s_4 are all good candidates. Exposing s_1 first would be good if s_1 turns out to be outside both $q_1.R$ and $q_2.R$, since in this case we know both queries must return false without needing further exposure. However, if s_1 is inside either $q_1.R$ or $q_2.R$, then we need to expose more unknown objects. On the other hand, exposing both s_2 and s_4 would be sufficient to answer false for both queries if both objects are outside the range of the queries. Exposing s_3 is not good since it has a low probability of being outside $q_2.R$. As this example shows, the decision of which unknown object to expose first is non-trivial.

We now define the evaluation with exposures problem.

Definition 4: Given a set of RAP queries, we define the **evaluation with exposures problem** as one of picking the next **unknown object** to expose which results in the minimum number of expected exposures.

Instead of considering all objects, we only need to consider the unknown objects. We do not need to consider objects which are fully inside $q.OR$ or in $q.OFI$ since these objects are definitely outside and inside the query range, respectively.

We propose an elegant information theoretic approach and a rank-based heuristic approach to solve the problem in Definition 4. The information theoretic approach provides a single generic formulation for use with any predicate. The rank-based heuristic requires less computation while exposing a similar number of objects when compared to the information theoretic approach.

4.5 Information theoretic approach

In this section, we outline our information theoretic algorithm. We propose a greedy algorithm that is based on an information gain calculation. Intuitively, information in our situation refers to how close we are to deterministically knowing the query result. Therefore, if exposing an object does not result in any information gain, then the exposure does not bring us closer to knowing if the query returns true or false.

4.5.1 Generic aggregation predicate

We present the generic information gain formulations which can be applied to queries with any aggregation predicate including: COUNT, SUM, AVERAGE, MIN and MAX. Consider a query q with a generic aggregation predicate. We denote the probability that a query returns true as $P(q)$ and false as $P(\bar{q}) = 1 - P(q)$. The probability that an object s is within range of q is denoted $P(s \in q.R)$. In order to compute the information gain, we need to first define the entropy for query q under the current safe regions. This is defined as usual:

$$H(q) = -P(q)log\, P(q) - P(\bar{q})log\, P(\bar{q})$$

We then consider the conditional entropy under the condition that only object s is exposed:

$$H(q|s) = P(s \in q.R)H(q|s \in q.R) + P(s \notin q.R)H(q|s \notin q.R)$$

where $P(s \in q.R)$ and $P(s \notin q.R)$ are the probabilities that s falls inside $q.R$ and outside $q.R$, respectively. $H(q|s \in q.R)$ and $H(q|s \notin q.R)$ are the entropies of q, assuming s falls inside $q.R$ and outside $q.R$, respectively. The $H(q|s \in q.R)$ entropy is given below. The equation for $H(q|s \notin q.R)$ is similar.

$$\begin{aligned} H(q|s \in q.R) &= -P(q|s \in q.R)log\, P(q|s \in q.R) \\ &- P(\bar{q}|s \in q.R)log\, P(\bar{q}|s \in q.R) \end{aligned}$$

The *information gain* of exposing s is then

$$G(q, s) = H(q) - H(q|s).$$

We then approximate the effect of exposing an object s on all unanswered queries in $U \subseteq Q$ to be $G(s) = \Sigma_{q \in U} G(q, s)$ and pick the object s in line 12 of Algorithm 1 which gives the greatest $G(s)$ value. (This is an approximation since we ignore the dependency among the queries with respect to their probabilities to return true.)

The critical part of the above calculation is computing the various probability values, namely $P(s \in q.R)$, $P(s \notin q.R)$, $P(q)$, $P(q|s \in q.R)$ and $P(q|s \notin q.R)$. Assuming uniform probability distribution for the location of s within $s.SR$, we can compute $P(s \in q.R)$ and $P(s \notin q.R)$ as the fraction of $s.SR$ which overlaps $q.R$ and outside of $q.R$ respectively.

Calculating $P(q)$: Once we know $P(s \in q.R)$ for each s, we can now compute $P(q)$ as follows:

$$\begin{aligned} P(q) = &\Sigma_{\{U \subseteq q.OU | AGGREGATE(U \cup q.OFI)\ \theta\ q.\gamma\}} (\Pi_{s \in U}(\\ &P(s \in q.R)) \times \Pi_{ns \in q.OU - U}(P(ns \notin q.R))) \quad (1) \end{aligned}$$

Equation 1 looks at all possible subsets U of $q.OU$ such that the aggregation predicate when applied to the objects $U \cup q.OFI$ compared to $q.\gamma$ returns true (when using the comparison operator θ).

Note that we can compute which subsets return true for any aggregation function since the server stores $s.a$ for all data objects s. For each of these subsets, we compute the probability that exactly this subset is in range. For example, if the aggregate is COUNT and θ is $\geq$ we would look at all subsets of unknown objects which contain $q.\gamma - |q.OFI|$ or more objects.

The remaining two probabilities, $P(q|s \in q.R)$ and $P(q|s \notin q.R)$, can be calculated similarly to $P(q)$. This is done by replacing $q.OU$ with $q.OU - \{s\}$, and in the case of $P(q|s \in q.R)$, we also replace $AGGREGATE(U \cup q.OFI)$ with $AGGREGATE(U \cup q.OFI \cup s)$.

Complexity analysis. The worst case run-time complexity of the information theoretic approach is $O(2n2^{n-1}(n-1)\mu) = O(\mu n(n-1)2^n)$ where $n = |OU|$ and μ is the number of unanswered queries. While the complexity is high, we propose strategies which are very effective in reducing the actual computational costs (as shown in results of Section 6.1).

4.5.2 Strategies for reducing computation

In this section, we present two strategies for speeding up the information theoretic algorithm without compromising result quality (results in same exposure order). The two strategies can be used together.

Both strategies focus on reducing the cost of computing the generic $P(q)$ formulation shown in Equation 1. The other computationally dominant terms $P(q|s \in q.R)$ and $P(q|s \notin q.R)$ can be improved in the same way. The first strategy caches previously computed results and the second exploits particular characteristics of the individual predicates.

Strategy 1: caching previous results. We reduce the number of times $P(q)$ is calculated by caching previous calculation results, since the calculations are often repeated. When the predicate is COUNT, the cache lookup key consists of the adjusted threshold $q.\gamma$ ($q.\gamma - |q.OFI|$) and the probability of each unknown object being in range. This lookup key results in high cache hits since typically there is only a small number of objects in $q.OU$.

We can also cache the previous calculations of $P(q)$ for SUM and AVERAGE. This is done by extending the COUNT cache by including $s.a$ for each object $s \in q.OU$. We also replace $q.\gamma - |OFI|$ with $q.\gamma - \Sigma_{s \in q.OU} s.a$ for SUM, and $q.\gamma$, $\Sigma_{s \in q.OFI} s.a$ and $|q.OFI|$ for AVERAGE. For the MIN predicate where θ is $<$ or $\geq$ ($\leq$ or $>$) the lookup key is $q.\gamma$ with $P(s \in q.R) \forall s \in q.OU$ where $s.a < (\leq) q.\gamma$. If θ is $=$ the key is $q.\gamma$ with $P(s \in q.R) \forall s \in q.OU$, where $s.a < q.\gamma$ and $\forall s \in q.OFI \cup q.OU$ where $s.a = q.\gamma$. MAX is similar.

Strategy 2: exploiting predicate-specific characteristics. For the COUNT predicate, we have two sub-strategies to reduce the computational cost of the generic $P(q)$ (Equation 1). The first sub-strategy reduces the number of subsets considered when computing $P(q)$ by only considering subsets of $q.OU$ with restricted cardinality. The second is to consider $P(\bar{q})$ instead of $P(q)$ if this results in even less subsets. The first method involves rewriting the generic $P(q)$ equation as follows.

$$P(q) = \Sigma_{\{\{s_1,\ldots,s_m\} \subseteq q.OU | m \; \theta \; q.\gamma - |q.OFI|\}} P(s_1 \in q.R) \times \cdots \times P(s_m \in q.R) \times P(s_{m+1} \notin q.R) \times \cdots \times P(s_n \notin q.R) \quad (2)$$

where $n = |q.OU|$. The second method is possible when calculating $P(\bar{q})$ results in a smaller number of subsets of $q.OU$ than computing $P(q)$. For example, if the comparison operator θ is $\geq$ and $q.\gamma - |q.OFI| - 1 < |q.OU| - (q.\gamma - |q.OFI|)$ then it is better to calculate $P(\bar{q})$ rather than $P(q)$, since the number of subsets which satisfy $m \geq q.\gamma - |q.OFI|$ (for $P(q)$ calculation) is larger than those which satisfy $m < q.\gamma - |q.OFI|$ (for $P(\bar{q})$ calculation). For future work, we will detail this strategy for the other predicates.

4.6 Rank heuristics

The computation cost of the information theoretic approach can be high when the number of objects is large. We therefore propose a faster heuristic algorithm for each of the standard aggregation predicates: COUNT; SUM; AVERAGE; MIN; MAX. These heuristics result in very similar exposures compared to the information theoretic approach. Conceptually, the algorithms can be thought of as follows. We consider two sets of objects. The first is the set of objects we would expose if we believe the query will return true. We call this the **true set**. The second is the objects to be exposed if we believe the query will return false. We call this the **false set**. We expose the single object that lies in the intersection of these two sets. Intuitively, exposing this object is the best choice because we would expose this object whether we believed the query will return true or false. We extend the above approach to handle multiple queries by each query calculating and nominating its preferred object for exposure. The object with the most nominations is then exposed. We call the heuristic algorithms created using this approach *rank* because each of the heuristics involve ranking the objects. In the subsequent subsections, we assume that the comparison operator θ is $\geq$. The other comparison operators can be done similarly.

Complexity analysis. The worst case run-time complexity of rank is far lower than the information theoretic approach. The complexity of rank for the COUNT, SUM and AVERAGE predicates is dominated by the complexity of ranking the objects. The complexity is therefore $O(\mu n \log n)$ where $n = |OU|$ and μ is the number of unanswered queries. For the MIN and MAX predicates, we only need to look for the object with the highest probability of being in range. The complexity is therefore $O(\mu n)$ for MIN and MAX.

4.6.1 COUNT predicate

We now describe the two overlapping sets (true and false set) for the COUNT predicate and the resulting heuristic. If we believe the query will return true, we need to show that $q.\gamma - |q.OFI|$ unknown objects are within $q.R$. We would select the unknown objects with the highest probability of being in $q.R$. Therefore, the true set is the $q.\gamma - |q.OFI|$ unknown objects whose safe regions overlap most with $q.R$. If however we guess that the query will return false, we should expose a set of unknown objects which leaves less remaining unknown objects than the threshold. To get closer to this situation, we need to expose unknown objects which are most likely to be outside $q.R$. The minimum number of unknown objects we can expose to show the query answer is false is $|q.OU| - (q.\gamma - |q.OFI|) + 1$. Therefore, the false set is the set of $|q.OU| - (q.\gamma - |q.OFI|) + 1$ objects with the lowest probability of being in $q.R$. There is one object in the intersection of the true set and the false set. This object is the one with the $(q.\gamma - |q.OFI|)^{th}$ highest overlap with $q.R$. This is the object we nominate for exposure. Each time an object is exposed to be within range, $|q.OFI|$ is increased by 1. We continue nominating the $(q.\gamma - |q.OFI|)^{th}$ highest overlap unknown object until the query is deterministically answerable.

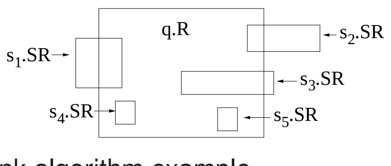


Fig. 3. Rank algorithm example

Figure 3 shows an example illustrating the rank algorithm for the COUNT predicate. In the example, the query returns true if there are at least 4 objects inside $q.R$ ($COUNT(P) \geq 4$). There are two objects completely inside $q.R$ and three objects intersecting it. Therefore $|q.OFI|$ and $|q.OU|$ are 2 and 3 respectively. Our true set is $\{s_3, s_1\}$, the $q.\gamma - |OFI| = 2$ unknown objects with the highest probability of being in $q.R$. On the other hand, our false set is $\{s_2, s_1\}$, the $|OU| - (q.\gamma - |OFI|) + 1 = 2$ unknown objects with the lowest probability of being in $q.R$. The intersection of the true set and the false set is s_1. Therefore, s_1 is the object nominated for exposure.

4.6.2 SUM and AVERAGE predicates

A variant of the rank algorithm for COUNT can also be designed for SUM and AVERAGE.

We describe the SUM algorithm as follows. Let $ec(s)$, defined formally below, be the expected contribution of object s to the sum of the attribute of interest (sum of $s.a$) for q. We define $ec(s)$ as: $ec(s) = P(s \in q.R) \times s.a$. Once we have calculated this value for all the objects which overlap the $q.UR$ region, we rank the objects in descending order on $ec(s)$. We then nominate for exposure the i^{th} object s_i such that $\Sigma_{j=1}^{i}(ec(s_j)) + \Sigma_{s \in q.OFI} s.a \geq q.\gamma$ and $\Sigma_{j=1}^{i-1}(ec(s_j)) + \Sigma_{s \in q.OFI} s.a < q.\gamma$. We continue this until the query can be deterministically answered.

We similarly define a rank algorithm for AVERAGE. We rank the objects which overlap $q.UR$ on $ec(s)$ which is an approximation of the expected contribution to the aggregation. Assume the ranked list is $s_1, s_2, ..., s_n$. We then nominate for exposure the object with the smallest i such that the expected average of the first i objects is above the threshold i.e. $\frac{\Sigma_{j=1}^{i} ec(s_j) + \Sigma_{s \in q.OFI} s.a}{\Sigma_{j=1}^{i} P(s_j \in q.R) + |q.OFI|} \geq q.\gamma$. We continue to nominate objects for exposure in this way until the query can be deterministically answered.

4.6.3 MIN and MAX predicates

For the MIN predicate, the only objects which can alter the query answer are those objects s, where $s.a < q.\gamma$. We therefore only consider these objects for exposure. We call the set of these objects UB. We again relate to the idea of selecting the object at the intersection of the true set and the false set. To show the query is true, we need to expose all objects $s \in UB$ and discover all exposed objects are outside $q.R$. Therefore, our true set is UB. To show the query answer is false, we only need to expose one object $s \in UB$ and discover that it is inside $q.R$. We select the object s in UB which is most likely to be inside $q.R$ ($s.SR$ with the highest percent overlap with $q.R$). This object is also in the query true set. Our heuristic therefore nominates this object for exposure. This is repeated until we find an object which is inside $q.R$ (query returns false), or UB is empty (query returns true).

The rank heuristic for the MAX predicate is similar.

5 Assign safe regions

Objects are assigned new safe regions whenever they move out of their safe region or are exposed during query evaluation (line 13 of Algorithm 1). As explained in the introduction, there is a fundamental tradeoff between larger and smaller safe regions. Larger safe regions will make it harder for the object to move out but will mean we know less about its exact location. The consequence is more exposures during query evaluation to find the deterministic query answer. Smaller safe regions are better for query evaluation but are easier for objects to move out of.

The optimal tradeoff will depend on a range of factors including: object movement speed; object movement pattern; query answering frequency; the density of queries around the object; the thresholds for the RAP queries; and so on. Most of these factors can potentially change with time. It would be impractical to formulate a mathematical model that incorporates all these factors and use the model to find a global optimal safe region assignment for all the objects. We therefore propose a practical and simple reactive mechanism to dynamically balance the competing concerns with minimal overhead. Our algorithm has the added benefit of being general enough to be used across a broad range of spatial queries.

In Section 5.1, we present our reactive strategy to adjust safe regions. Next, in Section 5.2, we discuss assigning the initial safe regions when the system starts with no safe regions for any moving object.

5.1 Reactive safe region assignment

The algorithm we propose simply reactively adjusts the size and shape of the safe region, based on the cause for an object to lose its safe region and its movement patterns. We use the perimeter length (rather than area) as the measure of size for a safe region. This was proven to be the correct metric to optimize by Hu et. al. [6] when assigning safe regions for moving objects. For each object o, we use a *size* value $o.s$ to keep track of the perimeter length of the safe region. That is, the safe region has a perimeter of length equal to $4\alpha o.s$, or $o.s =$ (perimeter of safe region)$/4\alpha$, where α is a system-level parameter called the unit length (or "quantum"). Intuitively, α is the "unit" by which the user wants the safe regions to grow or shrink in the x and y axes each time a safe region is re-assigned.

The $o.s$ value for object o is reactively adjusted as follows. When object o moves out of its safe region, it is marked as needing expansion. This is because o moving out of its safe region indicates the safe region is too small for o. In this case, we increase the $o.s$ value by 1. On the other hand, when the evaluation algorithm decides that o should be exposed in order to answer one or more queries, then o is marked for shrinking. In this case, we should decrease the $o.s$ value by 1. Once the $o.s$ value is determined for object o, what remains to finalize its safe region is the shape of the safe region. Shape for us is determined by the ratio between the rectangular safe region's x radius and y radius. Here, we advocate the use of

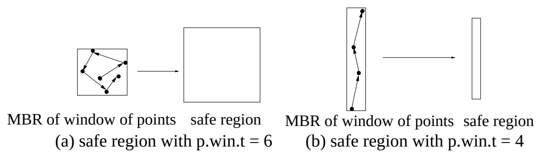


Fig. 4. Example safe regions derived from windows of $p.win.t = 6$ and $p.win.t = 4$ respectively.

the historical movement pattern of the object as a prediction of likely future movement patterns. For example, a person walking in a northernly direction should have a narrow vertical safe region rather than a square one. We assume that the object o (at the client side) constructs the minimum bounding region (MBR) enclosing object o's locations in the past window of time. Figure 4 shows two example safe regions derived from the MBRs defined by the past 6 δ time-unit intervals and 4 δ time-unit intervals, respectively, where δ is a system-level time unit parameter. To assign the new safe region, we use the ratio $\rho =$ MBR length along the x-axis / MBR length along the y-axis.

With $o.s$ and ρ values for o given, we can determine the height and width for the safe region ($o.SR.l_y$ and $o.SR.l_x$ respectively) of o as follows:

$$2o.SR.l_x+2o.SR.l_y = 4\alpha o.s \quad \text{and} \quad o.SR.l_x/o.SR.l_y = \rho.$$

Hence, we have

$$o.SR.l_x = 2\alpha o.s/(1+1/\rho) \text{ and } o.SR.l_y = 2\alpha o.s/(1+\rho) \quad (3)$$

We assign a rectangle with the above calculated dimensions centered on the location of o, as o's safe region. In Algorithm 3, we only consider objects

Algorithm 3 The re-assignment algorithm

Input: O_e: the set of all objects that need safe regions assigned
Output: safe regions for all objects in O_e
Method:

```
1: for every $o \in O_e$ do
2:    if $o$ was marked to expand then
3:       $o.s = o.s + 1$
4:    else if $o$ was marked to shrink then
5:       $o.s = max(o.s - 1, 0)$
6:    end if
7:    let $o.SR$ be the safe region having the center at
      $o$'s exact location, and width and height given
      in Equation 3.
8: end for
9: return the assigned safe regions for $O_e$.
```

which do not currently have a safe region due to either query evaluation exposure or moving out of their safe region. Each of these objects is either marked to expand or to shrink. Objects are marked to expand whenever they move out of their assigned safe region. We check if each object has been marked to expand on line 2. We expand (shrink) by incrementing (decrementing) $o.s$ by one (lines 2 - 6). In the case of shrink, we do not allow $o.s$ to reach below 0. Then Equation 3 is used to determine the x and y lengths of the rectangle, respectively. Finally, safe regions are assigned to the objects in O_e.

5.2 Initialize all the safe regions

The initialization step assumes that all objects do not have a safe region, and the system knows the exact locations of all of the objects. For all $o \in O$, we initialize $o.s = 1$ and $o.SR$ to be a square (i.e. $\rho = 1$). This method takes no consideration of whether a data object is close to a query or not. Intuitively, if a data object is far away from any query, we can start with a large safe region. However, we contend this may not be necessary for two reasons. First, our system will later reactively adjust the shape of the safe regions according to movement patterns. Second, we deal with a dynamic situation in which objects (including query objects) can appear anywhere. Hence, starting with a large safe region may not be a good strategy, especially if the object does not move much.

6 EXPERIMENTAL EVALUATION

Our experiments were conducted on a simulation of the RAP query processing environment. The experiments were run on a single core of an Intel i7 2.80GHz machine with 8GB RAM running Ubuntu 11.10. We used a variety of data sets, including the uniform, Gaussian and road network data sets with the default parameters of Chen et. al [3] and also the MilanoByNight simulation [10] data set. We only report the results for the MilanoByNight data set due to space constraints. The results for the other data sets were similar.

The MilanoByNight data set was designed to simulate people using a friend finder service. It modeled the movements of people on a typical Friday or Saturday night (between 7pm and 8:15pm) in a big city. The total area covered by user movement is 324 km^2. The data consists of 150 snapshots of object locations with a 30 second interval between snapshots. The data set was discretized to 1024 by 1024 cells.

We modeled all the users as moving data objects with a randomly-picked subset as moving query objects. The following parameters were generated following a Gaussian distribution: query range; the query threshold $q.\gamma$; the object attribute $s.a$. For the SUM predicate, we restricted $s.a$ to be greater than 0. The parameter values used are shown in Table 2, where μ and σ are used to represent the mean and standard deviation of Gaussian distributed data, respectively. The default values are in bold.

We simulated queries with different degrees of continuity by assigning queries with different probabilities (using random uniform distribution) of needing an answer at each query evaluation instance. Evaluation instances occur at a fixed period of 60 seconds by default.

parameter	values
no. of objects	1000, ..., **5000**, ..., 10000
no. of queries	**500**
$q.R$	$(\mu, \sigma^2) = (\mathbf{100}, \mathbf{20^2})$
aggregation predicate	**COUNT**; SUM; MIN
θ	$\geq$
COUNT: $q.\gamma$	$(\mu, \sigma^2) = (\mathbf{2}, \mathbf{1^2})$
SUM: $q.\gamma$	$(\mu, \sigma^2) = (\mathbf{80}, \mathbf{40^2})$
MIN: $q.\gamma$	$(\mu, \sigma^2) = (\mathbf{60}, \mathbf{30^2})$
SUM and MIN: $s.a$	$(\mu, \sigma^2) = (\mathbf{40}, \mathbf{20^2})$
Safe region assignment	
$p.win.t$	**4**
δ	**30 seconds**
α	**4**

TABLE 2
Experiment parameters (default values in bold)

The algorithms used in the experiments are as follows (where RSR refers to the use of our reactive safe region assignment algorithm):

RSR-InfoTh This is our system using the information theoretic evaluation algorithm with the two speedup strategies of Section 4.5.2. The cache of Strategy 1 of Section 4.5.2 was pre-filled.

RSR-Rank This is our system using the rank heuristics described in Section 4.6.

RSR-Naive This is our system where line 12 of Algorithm 1 selects a random unknown object to expose.

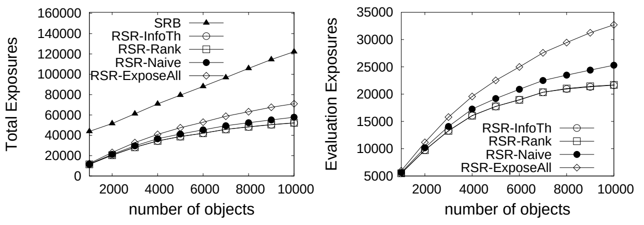
RSR-ExposeAll This is our system where lines 11 to 17 of Algorithm 1 are replaced with expose all unknown objects in one step. This effectively answers a traditional range query.

SRB This is the state-of-the-art safe region-based algorithm designed for the continuous stationary range and kNN queries proposed by Hu et. al. [6]. We adapted this work to answer RAP queries by first answering the range query and then applying the predicate. SRB cannot be trivially modified to exploit the characteristics of RAP queries when answering queries since it assigns conservative safe regions. This does not allow unknown objects to exist because it needs to ensure that the answer to the traditional range query is always known. Queries with different degrees of continuity were handled by simply reporting query answers less often. Moving queries were handled by deletion followed by insertion in the new position. The grid was set to the experimentally determined optimal of 256 $\times$ 256 divisions.

Evaluation metrics. For the majority of our experiments, we report the *total exposures* and *evaluation exposures*. The total exposures include object location exposures during query evaluation and objects moving out of their safe regions. Evaluation exposures is the number of times the objects reported their location during *query evaluation*. This metric reflects the quality of the evaluation algorithms used in the different variants of our algorithm. We do not report the results for SRB for this metric since SRB does not have a separate query evaluation step. Instead, it continuously adjusts safe regions to ensure the query results are always known.

6.1 Results with all three query characteristics present

In this section, we report the experimental results when all three extensions to the traditional range query are present, namely, RAP queries, queries of varying degrees of continuity and moving queries.



(a) Total number of exposures (b) Number of evaluation exposures

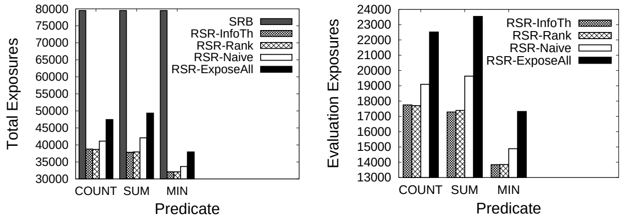
Fig. 5. Results of varying number of objects

Varying number of objects. In this experiment, we vary the number of objects. The results for the total number of exposures (Figure 5(a)) show our algorithms consistently outperform SRB by up to a factor of 3.84 (this increases up to 9.53 in later experiments) because our algorithms exploit the three query characteristics. We can see that exploiting RAP queries can save 26% of the total number of exposures by comparing RSR-InfoTh or RSR-Rank (both exploit RAP queries) against RSR-ExposeAll (does not exploit RAP queries).

For evaluation exposures (Figure 5(b)), RSR-ExposeAll is the worst performing RSR algorithm since it exposes all the unknown objects instead of incrementally exposing objects and testing query answerability.

The results show that RSR-InfoTh and RSR-Rank consistently outperform RSR-Naive, because RSR-InfoTh and RSR-Rank expose objects in a more optimal order. The RSR-InfoTh and RSR-Rank results are very similar because they follow a similar principle. RSR-Rank exposes the object with the $(q.\gamma - |OFI|)^{th}$ highest probability of being in $q.R$. This is the object we are most uncertain about for knowing whether the query is true or false. This is similar to RSR-InfoTh which exposes the object that is expected to reduce the overall entropy the most.

Varying aggregation predicate. In this section, we show the results for the experiments using the MIN, SUM and COUNT aggregation predicates. The results in Figure 6(a) show that all versions of our algorithm significantly outperform SRB. The difference between our algorithms and SRB is larger for the MIN predicate compared to COUNT and SUM. This is because the MIN query is much easier to answer than COUNT



(a) Total number of exposures (b) Number of evaluation exposures

Fig. 6. Results of varying the aggregation predicate

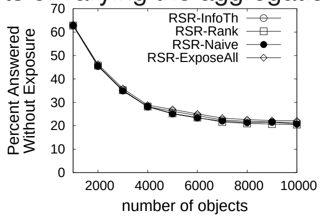


Fig. 7. Percentage of queries answerable without exposure

and SUM since we only need to find one object below the threshold which is in range.

Query safe region effectiveness. In this experiment, we explore the effectiveness of assigning safe regions to moving queries. The results in Figure 7 show that by using safe regions around moving queries, we are able to answer up to 63% of the queries without exposing the query or any data objects. As the number of objects increases, the query becomes harder to answer (smaller percentage of queries answered without exposure) since it becomes more likely that there are γ objects within the query's range. We expect that if the number of objects was to increase far enough, the query would become easier to answer again.

Computational costs. In these experiments, we compare our algorithms against SRB in terms of computational costs. Figure 8(a) shows that the average computation time per time stamp for all versions of our algorithm is less than 0.6 seconds for all number of objects tested. This is much lower than the 30 second interval between timestamps. We see that our approach requires similar computation time compared with SRB. As expected, as the number of objects increases, the RSR-Rank heuristic outperforms RSR-InfoTh by a larger margin.

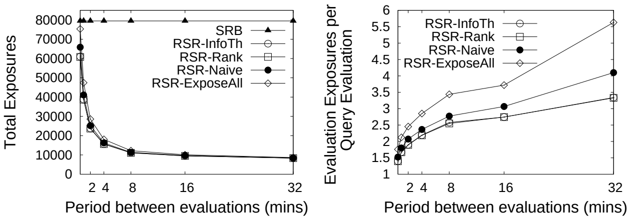
Figure 8(b) shows the safe region assignment algorithm consumes a significant part of the computational cost.

Figure 8(c) shows the breakdown of evaluation time between picking the objects to be exposed and the rest of evaluation, such as checking query answers and finding unknown objects. The results show the high computational costs incurred by RSR-InfoTh for picking which objects to expose compared to negligible picking costs of RSR-Rank.

6.2 Results of varying query characteristics

In this section, we compare our algorithm against SRB as each of the three query characteristics are gradually taken away. Due to space constraints, we only report the results for the COUNT predicate in this section.

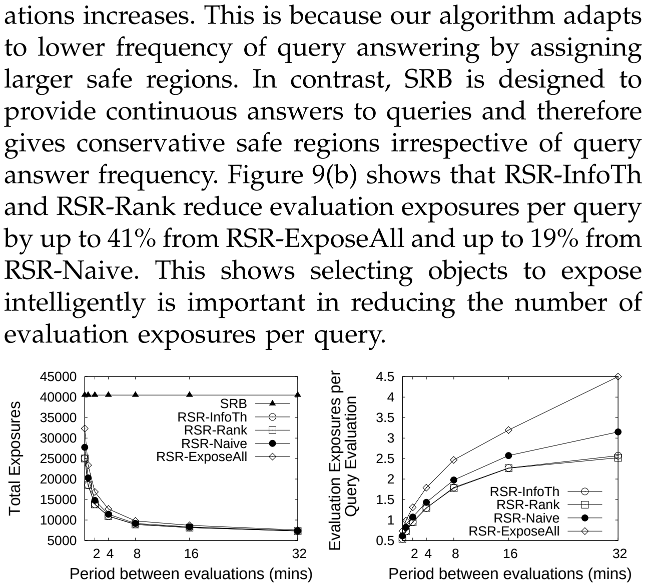
In these experiments, we vary the degree of continuity by changing the period between query evaluations (the time interval between when queries can be evaluated), thereby effectively moving along the spectrum from continuous queries (small period) to snapshot queries (large period). Consequently, the number of query evaluations changes throughout the graphs. Therefore, we report the number evaluation exposures per query evaluation instead of the total number of evaluation exposures.



(a) Total number of exposures (b) Evaluation exposures per query evaluation

Fig. 9. Results of varying degrees of continuity for moving RAP queries

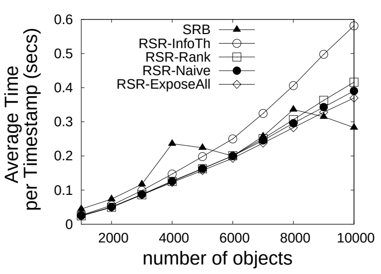
Moving RAP queries. In this experiment, we vary the degree of continuity of the queries. We keep both the other advantages of our algorithm, namely moving queries and RAP queries. The results in Figure 9(a) show all three of our algorithms consistently outperform SRB across the range of evaluation periods (different degrees of continuity) by up to a factor of 9.53. The margin by which our algorithm outperforms SRB increases as the time period between query evaluations increases. This is because our algorithm adapts to lower frequency of query answering by assigning larger safe regions. In contrast, SRB is designed to provide continuous answers to queries and therefore gives conservative safe regions irrespective of query answer frequency. Figure 9(b) shows that RSR-InfoTh and RSR-Rank reduce evaluation exposures per query by up to 41% from RSR-ExposeAll and up to 19% from RSR-Naive. This shows selecting objects to expose intelligently is important in reducing the number of evaluation exposures per query.



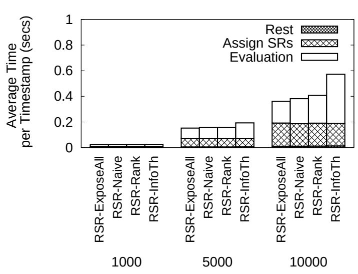
(a) Total number of exposures (b) Evaluation exposures per query evaluation

Fig. 10. Results of varying degrees of continuity for stationary RAP queries

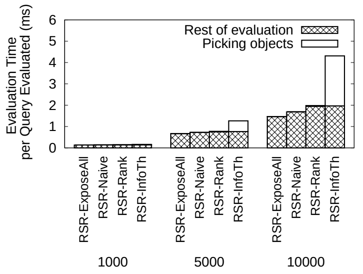
Stationary RAP queries. In this experiment, we remove the advantage of moving queries to be closer



(a) Average time per timestamp



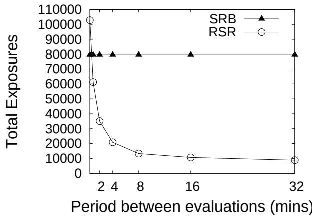
(b) Average time per timestamp breakdown for 1000, 5000 and 10000 objects



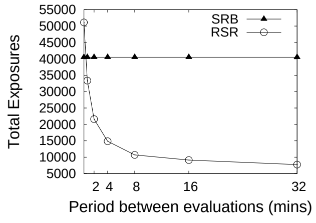
(c) Evaluation time breakdown for 1000, 5000 and 10000 objects

Fig. 8. Computational time results

to the setting that SRB is designed for (stationary queries). Therefore, the results in Figure 10 show that our algorithms outperform SRB by less in this setting than the previous experiment (Figure 9). For example, in Figure 9, our algorithms outperformed SRB by a factor of 8.36 when the period between evaluations was 16 minutes. However, with the same period between evaluations, that margin reduced to a factor of 4.97. The SRB safe regions are now more stable since queries do not move, which in turn significantly reduces the total number of exposures. In contrast, our algorithms no longer benefit from building safe regions around queries. However, the results show that our algorithms still outperform SRB due to the use of RAP queries.



(a) Moving queries



(b) Stationary queries

Fig. 11. Results of varying degrees of continuity for traditional range queries

Moving traditional range queries. In this experiment, we remove the advantage of RAP queries but retain the benefit of moving queries. We report the results for only one of our RSR variants (RSR-ExposeAll labeled as RSR). All our algorithm variants give the same results when answering a traditional range query since all unknown objects need to be exposed. The results in Figure 11(a) show that when the period between evaluation is higher than 30 seconds, our algorithm starts to outperform SRB. This is because the benefit of answering the queries less frequently outweighs losing the advantage of RAP queries.

Stationary traditional range queries. In this experiment, we remove both the RAP query and moving query advantages of our algorithm. These are the most favorable settings for SRB. We only show one variant of RSR for the same reason as the previous experiment. The results in Figure 11(b) show the benefit of our approach for queries with different degrees of continuity in isolation from the other two advantages. The results show RSR outperforms SRB by up to a factor of 5.23. The benefit is again due to our algorithms taking advantage of different degrees of continuity by relaxing constraints on safe regions. It is encouraging to note that our algorithm can still outperform SRB even with two of our advantages removed, albeit at a higher period between evaluations.

7 RELATED WORK

In this section, we review the work on handling continuous and snapshot spatial queries that has not been discussed in the introduction. We begin with work that does not use safe regions and then review the more related safe region-based work.

Mokbel et al. [11] proposed the Scalable INcremental hash-based Algorithm (SINA) for processing continuous range and k nearest neighbor (kNN) queries. Wu et al. [17] proposed an algorithm for processing continuous range queries with count predicates to aid in the efficient processing of reverse kNN queries. In contrast to our work, neither of these studies focus on reducing communication, instead they focus on reducing computational time.

In the work by Cai et al. [2] and Zheng et al. [19] objects use the resident domain and the roaming region respectively to directly monitor their nearby queries. Objects then report their location to the server whenever their movement effects a query answer. This model would not be efficient for RAP queries since the group property of RAP queries cannot be monitored by individual objects without incurring excess communication costs.

Prabhakar et al. [12] proposed a safe region-based solution for answering continuous range queries by swapping query and object roles and by assigning safe regions based on an object's distance to its nearest queries. Kalashnikov et al. [8] performed a comprehensive evaluation of different data structures for indexing safe regions in main-memory. They found the grid data structure gave the best results. SRB [6] is a safe region-based solution used to reduce communication when answering continuous range and kNN queries. Tzoumas et al. [16] propose a workload adaptive index system which self adjusts based on whether the workload is query intensive or location update intensive. Their index is designed for the snapshot range and kNN queries. Yin et al. [18] handle the proximity detection problem which detects if pairs of friends are within a given distance of each other. They use circular safe regions to reduce communication. Chen et al. [5] use the current trajectory of moving data objects to predict safe regions for the objects. These safe regions are used to answer range and kNN queries. Their work focuses on developing an efficient update protocol for continuous traditional range and kNN queries. All of these studies differ from our work because they are designed to handle continuous range queries that return all objects in the range, without a clear way of taking advantage of the three characteristics in our setting.

8 Conclusion

We have studied the problem of communication cost minimization for moving range queries with predicates and different degrees of continuity. The study reveals that communication costs can be greatly reduced by exploiting RAP query characteristics and adapting to the degree of continuity of queries using a reactive safe region assignment algorithm.

We propose a rank-based heuristic and an information theoretic approach for determining the order of exposing objects during query evaluation. The rank-based heuristic is computationally more efficient than the information theoretic approach but not as generic. The information theoretic approach allows the same formulation to be used with any predicate. Both approaches result in a similar number of exposures. Experiments show our system imposes substantially lower communication costs when compared to the existing state-of-the-art SRB algorithm for a wide variety of settings. The execution time results of our system show it is practical for use in real-time applications.

For future work, we will explore k nearest neighbor (kNN) queries with different degrees of continuity using a framework similar to ours and extend the framework for probabilistic range and kNN queries.

References

[1] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. *Journal of Computing Systems Sciences*, 66(1):207–243, 2003.

[2] Y. Cai, K. A. Hua, G. Cao, and T. Xu. Real-time processing of range-monitoring queries in heterogeneous mobile databases. *IEEE Trans. Mob. Comput.*, 5(7):931–942, 2006.

[3] S. Chen, C. Jensen, and D. Lin. A benchmark for evaluating moving object indexes. *Proceedings of the VLDB Endowment*, 1(2):1574–1585, 2008.

[4] S. Chen, B. C. Ooi, K.-L. Tan, and M. A. Nascimento. st^2b-tree: a self-tunable spatio-temporal b+-tree index for moving objects. In *SIGMOD Conference*, pages 29–42, 2008.

[5] S. Chen, B. C. Ooi, and Z. Zhang. An adaptive updating protocol for reducing moving object database workload. *Proceedings of the VLDB Endowment*, 3:735–746, September 2010.

[6] H. Hu, J. Xu, and D. L. Lee. A generic framework for monitoring continuous spatial queries over moving objects. In *Proceedings of SIGMOD*, pages 479–490, 2005.

[7] C. S. Jensen, D. Lin, and B. C. Ooi. Query and update efficient b+-tree based indexing of moving objects. In *VLDB conference*, pages 768–779, 2004.

[8] D. V. Kalashnikov, S. Prabhakar, and S. E. Hambrusch. Main memory evaluation of monitoring queries over moving objects. *Distributed and Parallel Databases*, 15(2):117–135, 2004.

[9] G. Kollios, D. Papadopoulos, D. Gunopulos, and V. J. Tsotras. Indexing mobile objects using dual transformations. *The VLDB Journal*, 14(2):238–256, 2005.

[10] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia. On the impact of user movement simulations in the evaluation of lbs privacy-preserving techniques. In *International Workshop on Privacy in Location-Based Applications*, 2008.

[11] M. F. Mokbel, X. Xiong, and W. G. Aref. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In *SIGMOD Conference*, pages 623–634, 2004.

[12] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch. Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects. *IEEE Transactions on Computers*, 51(10):1124–1140, 2002.

[13] S. Saltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD Conference*, pages 331–342, 2000.

[14] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD Conference*, pages 611–622, 2004.

[15] Y. Tao, D. Papadias, and J. Sun. The TPR*-Tree: An optimized spatio-temporal access method for predictive queries. In *VLDB Conference*, pages 790–801, 2003.

[16] K. Tzoumas, M. L. Yiu, and C. S. Jensen. Workload-aware indexing of continuously moving objects. In *VLDB Conference*, pages 1186–1197, 2009.

[17] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan. Continuous reverse k-nearest-neighbor monitoring. In *International Conference on Mobile Data Management*, pages 132–139, 2008.

[18] M. L. Yiu, L. H. U, S. Šaltenis, and K. Tzoumas. Efficient proximity detection among mobile users via self-tuning policies. In *VLDB Conference*, pages 985–996, 2010.

[19] B. Zheng, W.-C. Lee, K. C. K. Lee, J. Winter, and M.-C. Chen. Disqo: A distributed framework for spatial queries over moving objects. In *ICPP*, pages 414–423, 2010.

Mitzi McCarthy is a PhD candidate in the Department of Computer Science at La Trobe University Australia where she completed her undergraduate degrees. Her research interests include mobile data management and flash memory databases.

Zhen He is a senior lecturer in the Department of Computer Science at La Trobe University Australia. He obtained his undergraduate and PhD degrees in computer science from the Australian National University. His research interests include moving object databases, flash memory databases, relational database optimization and parallel databases.

X. Sean Wang is a Professor at the School of Computer Science, Fudan University, Shanghai, China. Before that, he was the Dorothean Chair Professor in Computer Science at the University of Vermont and a Program Director at the US National Science Foundation. He received his PhD degree in Computer Science from the University of Southern California. He has published widely in the general area of databases and information security, and was a recipient of the US NSF Research Initiation and CAREER awards. His research interests include database systems, information security, parallel large-scale data analytics.