Approximate Evaluation of Queries for Spatial, Uncertain and Probabilistic Databases

by

Haiquan Chen

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 6, 2011

Keywords: Data Cleansing, Spatial databases, RFID, Probabilistic Databases, Query Processing

Approved by

Wei-Shinn Ku, Chair, Assistant Professor of Computer Science and Software Engineering
Xiao Qin, Associate Professor of Computer Science and Software Engineering
Weikuan Yu, Assistant Professor of Computer Science and Software Engineering

Abstract

In modern geographic information systems, route search represents an important class of queries. In route search related applications, users may want to define a number of traveling rules (traveling preferences) when they plan their trips. However, these traveling rules are not considered in most existing techniques. In this dissertation we propose a novel spatial query type, the multi-rule partial sequenced route (MRPSR) query, which enables efficient trip planning with user defined traveling rules. The MRPSR query provides a unified framework that subsumes the well-known trip planning query (TPQ) and the optimal sequenced route (OSR) query. The difficulty in answering MRPSR queries lies in how to integrate multiple choices of points-of-interest (POI) with traveling rules when searching for satisfying routes. We prove that MRPSR query is NP-hard and then provide three algorithms by mapping traveling rules to an activity on vertex network. Afterwards, we extend all the proposed algorithms to road networks. By utilizing both real and synthetic POI datasets, we investigate the performance of our algorithms. The results of extensive simulations show that our algorithms are able to answer MRPSR queries effectively and efficiently with underlying road networks. Compared to the Light Optimal Route Discoverer (LORD) based brute-force solution, the response time of our algorithms is significantly reduced while the distances of the computed routes are only slightly longer than the shortest route.

The past few years have witnessed the emergence of an increasing number of applications for tracking and tracing based on Radio Frequency Identification (RFID) technologies. However, raw RFID readings are usually of low quality and may contain numerous anomalies. An ideal solution for RFID data cleansing should address the following issues. First, in many applications, duplicate readings of the same object are very common. The solution should take advantage of the resulting data redundancy for data cleaning. Second, prior knowledge

about the environment (e.g., prior object distribution, false negative rates of readers) may help improve data quality, and a desired solution must be able to take into account such knowledge. Third, the solution should take advantage of physical constraints in target applications (e.g., the number of objects in a same location cannot exceed a given value) to elevate the accuracy of data cleansing. There are several existing RFID data cleansing techniques. However, none of them support all the aforementioned features. In this dissertation we propose a Bayesian inference based framework for cleaning RFID raw data. We first design an n-state detection model and formally prove that the 3-state model can maximize the system performance. In addition, we devise a Metropolis-Hastings sampler with Constraints (MH-C), which incorporates constraint management to clean RFID data with high efficiency and accuracy. Moreover, to support real-time object monitoring, we present the Streaming Bayesian Inference (SBI) method to cope with real-time RFID data streams. Finally, we evaluate the performance of our solutions through extensive experiments.

Acknowledgments

I am sincerely grateful to my supervisor, Dr. Wei-Shinn Ku, for his academic guidance, valuable comments, continuous support, and encouragement during my doctoral studies. I am very appreciative of his personableness and endless help in the achievement of this work. Without his support, this dissertation would not have been possible. I owe my deepest gratitude to him.

I would like to thank my committee members: Dr. Xiao Qin and Dr. Weikuan Yu for their time, patience and advices that led to me improving this work.

Special thanks to Dr. Stanley Reeves for his time, kindness, and giving me the courage to realize my goal. I do believe it is God who leads me towards achieving this.

Finally, I am deeply grateful to my parents. It is their love and support that makes this dissertation possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Spatial Databases

In Geographic Information Systems (GIS) related research [8, 22, 49, 52, 55], significant efforts have been spent on nearest neighbor (NN) queries, range queries as well as their variants [38, 40, 57, 68]. While these query types are building blocks for many existing applications, more advanced spatial query types must be studied for future GIS systems. Route queries [40, 52] are an important class of spatial queries for users to request a efficient path by specifying a source and a destination. As an essential component, route queries are widely supported by many of today's popular online map service providers (e.g., Google Maps[1], MapQuest [2], Yahoo! Maps[3], Bing Maps[4]). By issuing a route query to a map service provider, users will obtain a recommended route on the map with an estimated mileage and turn-by-turn driving instructions. Li et al. [40] proposed solutions for Trip Planning Queries (TPQ). With TPQ, the user specifies a set of Point of Interest (POI) types and asks for the optimal route (with minimum distance) from her starting location to a specified destination which passes through exactly one POI in each POI type. On the other hand, Sharifzadeh et al. [52] presented OSR queries where the user asks for an optimal route from her starting location and passing through a number of POIs (each with a different type) in a particular order (sequence) imposed on all the types of POIs to be visited. However, both TPQ and OSR queries fails to consider the sub-sequences of POI types which occur naturally in many GIS applications. To remedy this, in this chapter, we propose a novel route query type,

[1] http://maps.google.com/
[2] http://www.mapquest.com/
[3] http://maps.yahoo.com/
[4] http://maps.bing.com/

Multi-Rule Partial Sequenced Route (MRPSR) query [10]. Our objectives are to assist users to plan trips that involve multiple POIs which belong to different POI categories (types) and satisfy a number of user defined traveling rules in road networks with a short response time. Our MRPSR query aims at unifying the well-known TPQ and OSR queries.

1.1.1 Motivation

As a motivating application, consider the scenario as shown in Figure 1.1. Alice is planning a trip that starts from her home and involves visiting the following POI categories: a bank, a restaurant, a gas station, and a movie theater. In addition, Alice also makes the following traveling rules on her trip:

1. Visit a bank to withdraw money before having lunch at a restaurant.

2. Fill up gas before going to watch a movie.

In order to fulfill these two traveling rules, the returned trip must follow two sub-sequences simultaneously: (a) traveling to a bank before going to a restaurant and (b) visiting a movie theater after filling up the gas tank in a gas station. Aside from these two sequences, Alice is free to visit any of the other POI categories in any order she pleases and furthermore, they can be interleaved in any order with the two rule-based sequences. Figure 1.1 illustrates two possible satisfying routes in a road network with different travel distances.

User defined traveling rules can be formulated as sub-sequences of POI categories in MRPSR queries. Such sub-sequences (or partial sequence) exists inherently in many GIS applications or can be specified by users as external constraints. Therefore, MRPSR queries are useful in numerous fields such as automotive navigation systems, transportation planning, supply chain management, online Web mapping services, etc.

Note that the MRPSR query differs from the *Traveling Salesman Problem* (TSP). In both cases a least-cost route is sought. However, with TSP a set of POIs (e.g., cities) is given

and each element must be visited exactly once. On the other hand, with MRPSR each POI is associated with a category and one may select any element of that category. For example, if the route should include a gas station visit, then one may choose any one of the available gas stations.

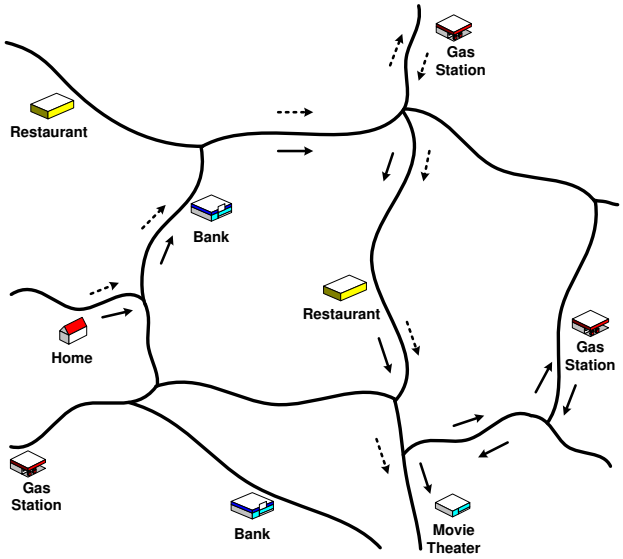


Figure 1.1: Two possible routes (solid and dashed arrows) of a MRPSR query.

1.1.2 Contribution

In this dissertation we present the MRPSR query and provide three fast approximation algorithms which are designed to efficiently compute satisfying routes with the near-optimal travel distance in road networks. The contributions of our work are as follows:

- We formally define the Multi-rule Partial Sequenced Route (MRPSR) query and prove the MRPSR problem to be a member of the *NP*-complete class.
- By casting traveling rules into an activity-on-vertex network, we utilize topological sorting [33] to integrate traveling rules with multiple choices of POIs and study the solvability of MRPSR queries.

- We propose the Nearest Neighbor-based Partial Sequence Route query (NNPSR) algorithm. The NNPSR algorithm uses activity-on-vertex networks to guide the search to retrieve a near-optimal route satisfying all the traveling rules in road networks.

- We integrate NNPSR with the Light Optimal Route Discoverer (LORD) algorithm [52] to create NNPSR-LORD that further reduces the trip distance based on the NNPSR algorithm.

- We also design an Advanced A* Search-based Partial Sequence Route query (AASPSR(k)) algorithm. AASPSR(k) takes advantage of the location of the destination as well as traveling rules to generate an efficient trip plan in road networks.

- We compare the performance of NNPSR, AASPSR(k) and NNPSR-LORD analytically.

- By using real and synthetic POI datasets, we compare experimentally the performance of NNPSR, AASPSR(k), NNPSR-LORD and the LORD-based brute-force solution in the road network of California.

1.2 RFID Databases

Radio Frequency Identification (RFID) is a very popular electronic tagging technology that allows objects to be automatically identified using an electromagnetic challenge/response exchange [66]. An RFID driven system consists of a large number of low-cost and identifiable tags that are attached to objects, and readers which can identify tags at a distance through RF communications. An increasing number of major retailers such as Wal-Mart, The Home Depot, Kroger, and Costco have installed RFID based inventory management systems in their warehouses and distribution centers. RFID technologies enable unprecedented visibility to support numerous track and trace applications. However, RFID researchers and practitioners are facing a challenging problem: the raw data collected by RFID readers are inherently unreliable [31, 56]. Therefore, middleware systems [30] are required to correct

readings and provide cleansed data. Most previous solutions [16, 21, 29, 31, 44] for cleansing RFID raw data focused on smoothing the readings generated by a group of readers. However, these existing solutions suffer from three major limitations:

- Data redundancy introduced by overlapping detection regions of multiple stationary readers (spatial redundancy) or continuous readings over time of a single mobile reader (temporal redundancy) is not utilized to improve reading accuracy.
- Prior knowledge about tagged objects and RFID readers is not effectively utilized to improve reading accuracy.
- Constraints in target applications (e.g., the maximal capacity of a room or a shelf) are not effectively utilized to cleanse the data.

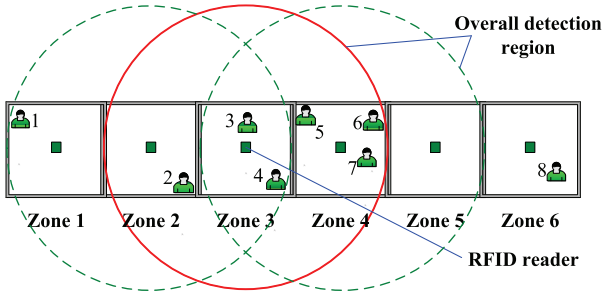


Figure 1.2: Spatial overlapping of detection regions.

	Zone 1 Reader	Zone 2 Reader	Zone 3 Reader	Zone 4 Reader	Zone 5 Reader	Zone 6 Reader
Obj 1	1	0	0	0	0	0
Obj 2	0	1	1	0	0	0
Obj 3	0	0	0	1	0	0
Obj 4	0	0	1	1	0	0
...	...	...	...	...	...	...

Table 1.1: RFID readings.

For addressing these limitations, in this dissertation we propose a Bayesian inference based framework to elevate the accuracy and efficiency of data cleansing by taking full advantage of data redundancy, prior knowledge, and application constraints [12].

1.2.1 Data Redundancy

Two types of redundancy may arise in RFID related applications: spatial redundancy, where an object is detected by multiple readers in its neighborhood, and temporal redundancy, where an object is detected multiple times by a single reader over time.

Spatial Redundancy: In order to reduce the complexity of data analysis, previous works [16, 21, 29, 31, 44, 67] assume that each object is read once, and read by one reader only. Clearly, this assumption is difficult to enforce, and more importantly, it oversimplifies the reality. Because RFID readings are of low quality, many applications have to employ redundant readers to cover the target area completely to improve localization accuracy, which means objects are read by multiple readers simultaneously.

Indeed, in RFID systems, spatial redundancy is very common. Figure 1.2 shows an example of spatial redundancy where the target area is divided into six zones (using one dimensional model) and an RFID reader is located in the center of each zone. Spatial overlap of readers' detection regions leads to duplicate readings, i.e., an object is in the detection regions of multiple readers. A possible set of readings is shown in Table 1.1 wherein 1's denote successful detections and 0's otherwise. The table shows two effects of redundancy:

- Object 2 is detected by the reader in Zone 2 and also the reader in Zone 3, which makes it difficult to tell the exact location of Object 2. However, since an object cannot appear in more than one zone at the same time, at least one of the readings belongs to spatial redundancy.

- Object 3 is detected in Zone 4 only. However, it does not necessarily mean that Object 3 is in Zone 4 for sure. It is possible that the reader in the zone where Object 3 is located simply failed to detect it.

On the first look, spatial redundancy causes confusion as it introduces inconsistent information (e.g., about the location of Object 2). However, redundant readings may supply the necessary information for the system to derive the location of an object when its intended

reader fails to detect it (e.g., Object 3). Thus, the challenge is how to take advantage of redundancy while avoiding its undesirable effect in data cleansing.

Temporal Redundancy: Besides employing multiple stationary readers, many applications monitor the target area using a mobile reader (e.g., a handheld or a robot-mounted reader [61]) to take continuous readings on its route. Because the exact location of the mobile reader is always changing when the reader reports raw data, the detection regions at different time points may overlap, introducing temporal data redundancy of readings. However, if we treat the same reader at different time points as different readers, e.g., as shown in Table 1.1, when a mobile reader traverses from zone 1 to zone 6, the reader can be considered as the zone 1 reader while moving in zone 1 and can be treated as the zone 2 reader while moving in zone 2, the temporal redundancy problem can be reduced to the spatial redundancy problem. Therefore, we will mainly focus on spatial redundancy in this dissertation.

1.2.2 Prior Knowledge

As false negatives and false positives abound in raw RFID readings [31, 56], in order to recover the true information, the data cleansing system should take prior knowledge into account. Prior knowledge may include information such as, for example, the detection areas of readers in Zone 2 and Zone 3 have significant overlapping, the position of the reader in Zone 4 makes it more likely to detect objects in Zone 3 than objects in Zone 5, or the reader in Zone 3 has high false negative rate, etc. Such information, when properly integrated with the readings, is extremely valuable for data cleansing.

1.2.3 Constraints

Environmental constraints can be utilized to improve data cleansing. For example, the maximal capacity of each zone (the number of objects that can reside in the same zone) is a constraint. If each zone represents a rack or shelf in a warehouse, one possible constraint is

the total size or weight of the objects which the rack can hold. In addition to these physical constraints, information obtained from other channels can be translated into constraints. For instance, if an extra source indicates that two certain objects are in the same zone, it may help cleanse the data of these two as well as other objects when the information is integrated with readings and other constraints.

1.2.4 Overview of Our Approach

In this chapter, we propose an innovative approach of cleansing RFID raw data which is able to take full advantage of duplicate readings and integrate prior knowledge as well as environmental constraints. Our approach is based on Bayesian inference. We introduce an n-state detection model and prove by entropy analysis that the 3-state detection model can maximize the system performance. Furthermore, we devise a Metropolis-Hastings sampler with constraints to approximate the posterior efficiently (Metropolis-Hastings sampling is a Markov Chain Monte Carlo method [4, 46]). Besides, we present the Streaming Bayesian Inference (SBI) method to deal with RFID data streams in support of real-time object monitoring. Consequently, our framework is able to answer the location query for all tagged objects in a target area in a real-time fashion based on the cleansed RFID raw data.

The contributions of our work are as follows:

- By using Bayesian inference, we derive a universal framework for computing the posterior probabilities (of the location of each object).
- Based on the physical characteristics of RFID readers, we propose an n-state detection model to capture likelihoods, which enables us to take full advantage of duplicate readings.
- By investigating the impact of the number of states in a detection model on the system entropy, we formally prove that the system entropy can be minimized if the 3-state

model is adopted compared with other state models. In other words, having even more states (greater than 3) can in fact deteriorate the overall system performance.

- We devise MH-C, an improved Metropolis-Hastings sampler, to sample from the posterior while taking the environmental constraints into consideration.
- In order to enable real-time object monitoring, we present the Streaming Bayesian Inference method to cope with the real-time RFID data stream.
- We demonstrate the efficiency and effectiveness of our approach through extensive simulations.

Chapter 2

Approximate Query Evaluation on Spatial Databases

2.1 The Multi-Rule Partial Sequenced Route Query

In this section, we formulate the proposed multi-rule partial sequenced route query and then discuss the properties of the proposed query type.

2.1.1 Problem Formulation

Definition Given n disjoint sets of POI category $\{C_1, C_2, \ldots, C_n\}$, each containing a number of POIs in R^2, the MRPSR query is to search for a route that satisfies the following three requirements:

1. The route will traverse through exactly one POI in each category;
2. The total traveling distance is minimized;
3. The route conforms with the given constraints (i.e., traveling rules).

While the first two requirements are commonly seen in the other types of route queries [40, 52], the third requirement is unique. Here, the issue is how we should properly define a constraint. Without loss of generality, we assume that each constraint can be mapped into a partial sequence rule, defined as follows.

Definition A *partial sequence rule* is defined as an ordered subset of categories $C_{k_1} \rightarrow C_{k_2} \rightarrow \cdots \rightarrow C_{k_m}$, which specifies the order of visits between $< C_{k_i} >$ in the subset.

For instance, a user may issue a MRPSR query with a constraint that he would like to withdraw money at a bank before going for grocery shopping and dinner. This constraint can be converted to the following two partial sequence rules:

1. $C_{Bank} \rightarrow C_{Supermarket}$

2. $C_{Bank} \rightarrow C_{Restaurant}$.

These two rules enforce that a Bank should be visited before a supermarket and a restaurant on the trip, but do not put a restriction on the order between the supermarket and the restaurant.

Notice that if no restriction is placed on the format of the user's constraints, the translation itself is a challenging artificial intelligence research problem [45]. The human natural language can be ambiguous and non-grammatical. Automatic translation requires algorithms that can deal with not only the ambiguity but also with parsing and interpretation of a large dynamic vocabulary, which is not likely to be accomplished in real time. With the help of input forms, the types of the user's constraints can be limited so that the translation from the constraints to the partial sequence rules can be handled with ease. With the notion of the partial sequence rules, the compatibility of a set of partial sequence rules can be defined as follows.

Definition A set of the partial sequence rules is defined to be *compatible* if and only if there is a total order of $< C_i >$ that satisfies the order specified in each of the rules in the set.

For instance, the set of rules $\{C_1 \rightarrow C_2, C_2 \rightarrow C_3, C_3 \rightarrow C_1\}$ is not compatible since it will be impossible to satisfy all these three rules at the same time. When all the travel constraints are represented as a set of partial sequence rules, the original definition of the MRPSR query can be formulated as follows.

Definition Given a set of POI categories and a set of partial sequence rules, a MRPSR query is defined to return the route with the minimal total traveling distance that satisfies the order specified in each of the partial sequence rules.

2.1.2 Properties of the MRPSR Query

The following theorem shows that MRPSR query provides a unified framework that subsumes the well-known trip planning techniques, including the trip planning queries (TPQ) [40] and the optimal sequenced route (OSR) queries [52].

Theorem 2.1. *The problems of the trip planning query and the optimal sequenced route query are special cases of the problem of the multi-rule partial sequenced route query.*

Proof. According to [40], the problem of the trip planning query is identical to the problem of the multi-rule partial sequenced route query when the set of partial sequence rules is empty. In addition, according to [52], the problem of the optimal sequenced route for a given sequence of categories of POIs is the same as the problem of the multi-rule partial sequenced route query when the set of partial sequence rules contains one partial sequence rule specifying the same order. □

From Theorem 2.1, we obtain the following important property for the MRPSR query.

Corollary 2.2. *The problem of the multi-rule partial sequence route query is NP-hard.*

Proof. According to [40], the problem of the trip planning query is NP-hard. Since the problem of the trip planning query is a special case of the problem of the multi-rule partial sequenced route query (Theorem 2.1), it follows immediately that the problem of the multi-rule partial sequenced route query is NP-hard. □

Corollary 2.2 implies that when the search space is large, it is advisable to quickly find a *suboptimal* route that satisfies the given partial sequence rules instead of the route with the minimal total distance.

The set of the partial sequence rules plays an important role in the MRPSR query. As indicated in Theorem 2.1 and Corollary 2.2, if the set is empty, the search space will be large and the MRPSR query is NP-hard. However, if the rule specifies the total order of the categories, the MRPSR problem can be solved in polynomial time [52]. Intuitively, the

tighter the set of rules is, the smaller the search space will be and the easier the MRPSR query can be answered. While it is difficult to quantify the level of tightness for a set of partial sequence rules, we provide Theorem 2.3 to see if a given set of rules will possibly lead to a solution. Theorem 2.3 shows the relationship between the solvability of a MRPSR query and the compatibility of a given set of rules.

Theorem 2.3. *If a multi-rule partial sequenced route query is solvable, then the corresponding set of the partial sequence rules must be compatible.*

Proof. The proof is done by contradiction. Assume that the set of rules is not compatible, then according to Definition 3 there is no ordered sequence of categories that satisfies all the rules. In other words, no matter how POIs are selected, it will be impossible to order them so that the ordered sequence meets all of the constraints. □

Notice that Theorem 2.3 does not guarantee that a compatible set of partial sequence rules can always lead to a solution for a corresponding MRPSR query because some categories may contain no POI. If each category contains at least one POI, the inverse of Theorem 2.3 (i.e., the compatible set of rules implies the solvability of the corresponding MRPSR query) will also be true. According to Definition 2.1.1, if the partial sequence rules are compatible, then there must exist at least one total order of categories $< C_i >$ that satisfies the order specified in each of the rules. Let one of such orders be $\{C_1, C_2, \ldots, C_n\}$. Now since each category is not empty, we can arbitrarily pick one POI p_x from each category C_i to compose a route $\{p_1, p_2, \ldots, p_n\}$ which traverses through exactly one POI in each category and conforms with the given traveling rules. According to Definition 2.1.1, if there is only one such route, we have our answer. If not, the one with the minimal traveling distance will be what we want to retrieve. In Section 2.2, we will elaborate how to verify if a set of partial sequence rules is compatible.

2.1.3 Percentage of the Constrained Categories

Definition Given a MRPSR query, the *Percentage of the Constrained Categories (PCC)* is defined as the percentage of the number of categories included in the set of traveling rules over the total number of categories to be visited in the query.

PCC is used to measure the extent that a MRPSR query is constrained by traveling rules. According to the definition of PCC, the trip planning query (TPQ) [40] can be considered as a MRPSR query with a PCC of 0% while the optimal sequenced route (OSR) query [52] can be treated as a MRPSR query with a PCC of 100%.

2.2 Activity-on-Vertex Networks

In order to plan a route which can fulfill all the user defined partial sequence rules, we need a solution to combine all the provided traveling rules and verify if they are compatible. The relationship between all the given traveling rules can be represented as a directed graph in which the vertices represent POI categories and the directed edges represent prerequisites. This graph has an edge $<i, j>$ if and only if category i is an immediate prerequisite for category j in one of the rules. The complete graph is named Activity-On-Vertex (AOV) network [27]. The following theorem provides the relationship of an AOV network and the compatibility of the traveling rules.

Theorem 2.4. *The partial sequence rules are compatible if and only if the corresponding AOV network is a directed acyclic graph.*

Proof. Definition 3 indicates that the rules are compatible if and only if there is a category sequence that satisfies the order specified in each of the traveling rules. Let that category sequence be the feasible sequence of tasks that satisfies all of the orders. According to [27], an AOV has a feasible sequence of tasks if and only if the precedence relations in the AOV network are both transitive and irreflexive. In other words, the corresponding AOV network must be directed and acyclic. □

Table 2.1 lists the POI categories and partial sequence rules specified by an example MRPSR query Q. The corresponding AOV network for Q is shown in Figure 2.1.

Data Type	**Name**	**Prerequisites**
C1	Bank	None
C2	Bookstore	None
C3	Restaurant	C1, C2
C4	Gas Station	None
C5	Hospital	C4
C6	Shopping Center	C5
C7	Church	C3, C6
C8	Coffee Shop	C3
C9	Gift Shop	C7, C8
C10	Park	C7

Table 2.1: POI categories and partial sequence rules in an example MRPSR query Q.

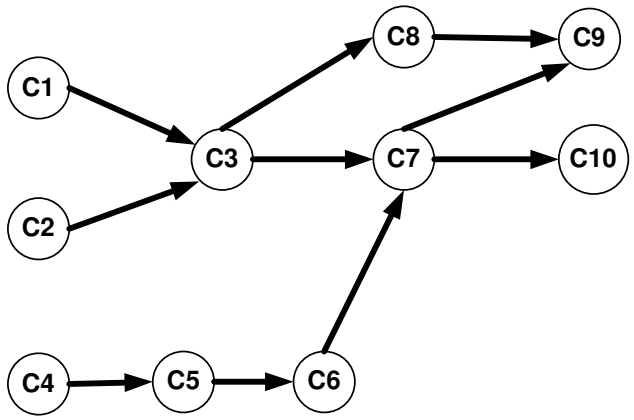


Figure 2.1: The AOV network of Q represents POI categories as vertices and prerequisites as edges.

After we represent all the partial sequence rules in a MRPSR query as an AOV network, providing that the AOV network is directed and acyclic, *Topological Order* (or *Topological Sorting*) can be used to generate a feasible complete ordering of POI categories which is compatible with every partial sequence rule in the MRPSQ query. In graph theory, a *topological order* of a directed acyclic graph (DAG) is a linear ordering of its vertices in which each vertex comes before all vertices to which it has outbound edges. Each DAG has at least

one *topological order*. The algorithm to find a *topological order* is as follows. The first step is to list out a vertex in the network that has no predecessor. Then the second step is to delete this vertex and all edges leading out from it from the AOV. By repeating these two steps until either all the vertices have been listed or all remaining vertices have predecessors and hence none of them can be removed. In the latter case, the AOV has a cycle and the trip is infeasible, i.e., the partial sequence rules are not compatible. If a topological order has the property that all pairs of consecutive vertices in it are connected by AOV edges, then these edges form a directed Hamiltonian path in the AOV [33]. If a Hamilton path exists, the topological sort order is unique and no other order respects the edges of the path. On the contrary, if a topological order does not form a Hamiltonian path, the AOV will have two or more valid topological orderings, for in this case it is always possible to form a second valid ordering by swapping two consecutive vertices that are not connected by an AOV edge to each other. For supporting both cases, we keep a counter of the number of immediate predecessors for each vertex and represent the network by its adjacency lists. Then we can carry out the deletion of all incident edges of a vertex v by decreasing the predecessor count of all vertices on its adjacency list. Whenever the count of a vertex drops to zero (in-degree = 0), we place the vertex on a list (L_{zero}) of vertices with a zero count. As mentioned in Section 1.1, the traveling rules (the AOV network) may not cover all the user selected POI categories. With the goal of creating a complete trip plan (i.e., the plan covers all requested categories), we add all the requested POI types which are not included in the AOV into the list L_{zero}. The complexity of topological sort is $O(e + n)$, where n is the number of vertices and e is the total edge number. The sort can be finished in linear time.

2.3 Algorithm Design

After having the AOV networks in hand, we can start to compute a trip plan satisfying all the traveling rules. In this section, we propose three approximate algorithms to answer a MRPSR query: the Nearest Neighbor-based Partial Sequenced Route (NNPSR) algorithm,

the Nearest Neighbor-based Partial Sequenced Route with Light Optimal Route Discoverer [52] (NNPSR-LORD) algorithm, and the Advanced A* Search-based Partial Sequenced Route (AASPSR(k)) algorithm. NNPSR applies AOV networks to capture traveling rules and launches successive nearest neighbor queries to answer a given MRPSR query. NNPSR-LORD utilizes the Light Optimal Route Discoverer [52] to optimize the route obtained by NNPSR. In AASPSR(k), as a hybrid scheme of NNPSR and ASPSR, distance heuristic functions are integrated with NNPSR to answer a MRPSR query. All of the proposed algorithms aim to find the near-optimal route which follows all of the traveling rules. Table 2.2 summarizes our set of notations.

Symbol	Meaning
$\mathbb{A}$	The adjacency list representation of an AOV
$\mathbb{C}$	The set of all the user selected categories
$\mathbb{R}$	The set of all the traveling rules
$\mathbb{P}$	A set of POIs
Q	The priority queue
S	The starting point of a MRPSR query
D	The destination of a MRPSR query
q	The query point of a nearest neighbor query
C_i	A POI category
$C_i.\mathbb{P}$	All the POIs of a category
L_{zero}	A list of AOV vertices with a zero count
L_{route}	A list of the POI sequence of a trip plan
P_{NN}	The query result of a nearest neighbor query
$Dist_E(x, y)$	The Euclidean distance between points x and y
$Dist_N(x, y)$	The network distance between points x and y

Table 2.2: Symbolic notations.

2.3.1 Nearest Neighbor Search in Road Networks

In practice, users usually move only in the underlying road networks rather than traveling freely through obstacles (e.g., buildings, rivers, etc.). Network distance computations and nearest neighbor queries in road networks have been well studied [32, 43]. As the basic building block of our proposed algorithms, in this subsection, we briefly review how to answer a nearest neighbor query in road networks by the incremental network expansion approach [43].

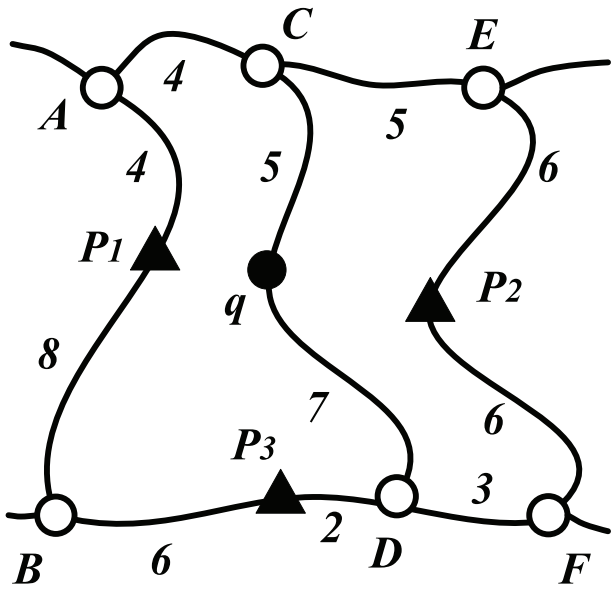


Figure 2.2: NN search by the incremental network expansion [43].

Figure 2.2 demonstrates the nearest neighbor search by the incremental network expansion [43]. In Figure 2.2, the blank point,q, means the query point, white points, A,B,C,D,E,F, denote road network conjunctions, and triangles, P_1, P_2, P_3, represent POIs. (P_1, P_2, P_3) are in ascending order of their Euclidean distance to q. Incremental network expansion performs network expansion from q and examines POIs in the order they are encountered. To be specific, first, the road segment CD that covers q is found and all POIs on CD are retrieved. Then a priority queue, $Q =< (C, 5), (D, 7) >$, is initiated, since no POI is covered by CD, the node C closest to q is de-queued and its adjacent nodes, A and E, are inserted into Q

with their accumulated distance from q, i.e., $Q =< (A, 9), (E, 10), (D, 7) >$. No POI is found on CA and CE. Then D, whose distance is closest to q in current Q, is expanded and its adjacent nodes, B and F, are en-queued, after which $Q =< (A, 9), (E, 10), (B, 15), (F, 10) >$. Afterwards, P_3 can be discovered on DB with an accumulated distance of 9 while no POI is found on DF. This distance offers a upper bound to restrict the search space. Because the next node to expand is A and the distance from q to A is already 9, which is no less than the upper bound, the algorithm terminates and returns P_3 as the nearest neighbor to q with a network distance of 9.

2.3.2 Nearest Neighbor-based Partial Sequenced Route Algorithm

Here we devise a Nearest Neighbor-based Partial Sequence Route (NNPSR) query algorithm by utilizing both the L_{zero} list and the nearest neighbor query (i.e., the incremental network expansion [43] based implementation) to generate a trip satisfying all the traveling rules. With NNPSR, we first search for the nearest POI to the query point q (as the starting point) whose category is included in L_{zero}. The retrieved nearest POI P_{NN} will be stored in a route list L_{route} and the category of P_{NN} (i.e., $P_{NN}.C$) will be removed from L_{zero}. Next, we update the adjacency list and new zero count vertices may be added to L_{zero}. In addition, the query point q is also updated to the location of P_{NN}. The process will repeat until all the selected categories are contained in the route. The complete algorithm of NNPSR is formalized in Algorithm 1.

2.3.3 NNPSR with Light Optimal Route Discoverer Algorithm

Suppose a complete POI sequence to be visited is given, the Light Optimal Route Discoverer (LORD) algorithm [52] can guarantee finding a route of minimum distance. Since we can obtain a complete POI sequence after each execution of the NNPSR algorithm, we can further optimize the trip by applying LORD on the POI sequence found by NNPSR. LORD is a threshold-based algorithm and requires less memory space compared with Dijkstra's

shortest path solution. The first step in LORD is to issue consecutive nearest neighbor queries to find the greedy route that follows the given POI category sequence from the starting point. Then, the length of the greedy route becomes a constant threshold value T_c. In addition, LORD also keeps a variable threshold value T_v whose value reduces after each iteration, and LORD discards all the POIs whose distances to the starting point are more than T_v. Afterward, LORD iteratively builds and maintains a set of partial sequenced routes in the reverse sequence (i.e., from the end points toward the starting point). During each iteration of LORD, POIs from the following category are added to the head of each of these partial sequence routes to make them closer to the starting point. The two thresholds are utilized to prune non-promising routes for reducing the search space.

After executing the NNPSR algorithm, we can acquire a sequence of POIs. Since each POI belongs to an individual POI category, we can also obtain a POI category sequence as the input of LORD. For most cases, the NNPSR-LORD solution outperforms the original NNPSR algorithm in terms of route distance. More detailed performance evaluations are contained in Section 2.4.

2.3.4 Advanced A* Search-based Partial Sequenced Route Algorithm

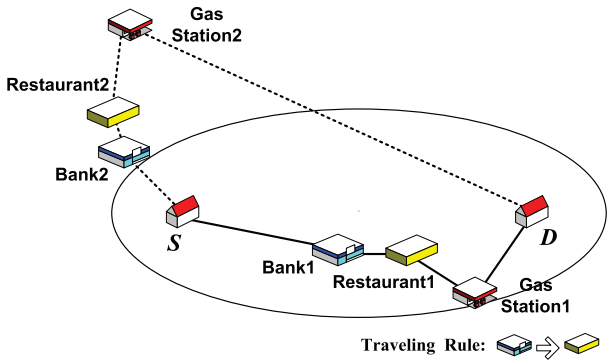


Figure 2.3: Two trips generated by NNPSR (the dashed route) and ASPSR (the solid route) with the traveling rule **Bank** $\rightarrow$ **Restaurant**.

Although the NNPSR and NNPSR-LORD algorithms can fulfill the traveling rules and reduce the travel distance of a trip, they do not consider the location of the destination

when greedily generating the route sequence. Consider the example shown in Figure 2.3. In Figure 2.3, S and D denote the start point and destination of the trip. Suppose we have a traveling rule which can be denoted as **Bank** $\rightarrow$ **Restaurant**. We will find that the dashed route returned by NNPSR is much longer than another feasible trip (the solid route) which considers the location of the destination. Therefore, another approach is to limit the trip planning within a range defined by S and D (e.g., an ellipse whose two focal points are S and D).

The A* search based Partial Sequenced Route (ASPSR) algorithm considers the location of the destination in its heuristic function. Similar to the *admissible heuristic* of the A* algorithm [48], in ASPSR, we retrieve the POI p with the minimum cost of $Dist_E(S, p) + Dist_E(p, D)$ in each category included in L_{zero}. Afterward the POI p with the lowest cost will be added into the route list L_{route} and the category of p will be withdrawn from L_{zero}. Then both $\mathbb{A}$ and L_{zero} will be updated and the location of p is set as the new query point. The process will reiterate until all the user selected categories are covered.

However, there could be roundabout ways when we plan a trip by ASPSR. Consider the example as shown in Figure 2.4. Because the POIs which are closer to the major axis of the ellipse (S and D are the two focal points) have a lower distance cost, $Bank1$, $Restaurant1$ and $gasstation1$ will be picked sequentially in ascending order of their costs. Consequently, a detour will occur where the user has to travel far away from from D to visit $Restaurant1$ and $gasstation1$ before reaching D at last. Therefore, we need to improve ASPSR to solve the aforementioned problem.

The improved version of ASPSR is named as the Advanced A* Search-based Partial Sequenced Route query (AASPSR) algorithm. In the following sections of this dissertation, we use AASPSR(k) to denote our AASPSR algorithm with parameter k to specify the number of POI's we retain for each category. AASPSR(k) can be considered as a hybrid scheme to combine ASPSR and NNPSR. To be specific, AASPSR(k) first computes $C_i.\mathbb{P}^*$ for each category C_i in $\mathbb{C}$ such that every POI in $C_i.\mathbb{P}^*$ is a POI with the top-k minimum

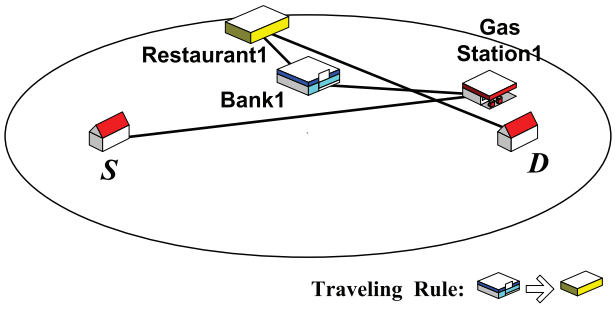


Figure 2.4: A trip search by ASPSR with the traveling rule **Bank** $\rightarrow$ **Restaurant**.

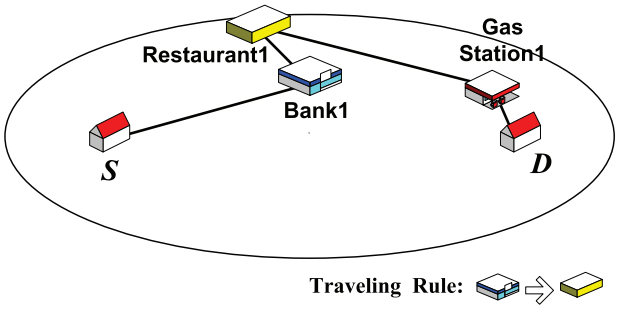


Figure 2.5: A trip search by AASPSR(1) with the traveling rule **Bank** $\rightarrow$ **Restaurant**.

traveling distance sum from S to D in C_i. In particular, if $k = 1$, only one POI with the shortest traveling distance sum from S to D for C_i is added to $C_i.\mathbb{P}^*$. On the contrary, if $k = \infty$, then all the POIs on the underlying road network will be included in $C_i.\mathbb{P}^*$ for each C_i. After $C_i.\mathbb{P}^*$ has been generated for each category, we launch NNPSR to generate a route only on these selected POIs in each $C_i.\mathbb{P}^*$. Starting with S, we search for the nearest POI p in $C_i.\mathbb{P}^*$ ($C_i \in L_{zero}$). Afterward, p is inserted into L_{route} and the location of p is used as the query point of the following NN query. Next we remove the category of p from L_{zero} and recompute the adjacency list. The whole process will repeat until L_{zero} becomes empty. The complete algorithm of AASPSR(k) is illustrated in Algorithm 2. For comparison purpose, the trips generated by AASPSR(1) and by AASPSR(2) are demonstrated in Figure 2.5 and Figure 2.6, respectively.

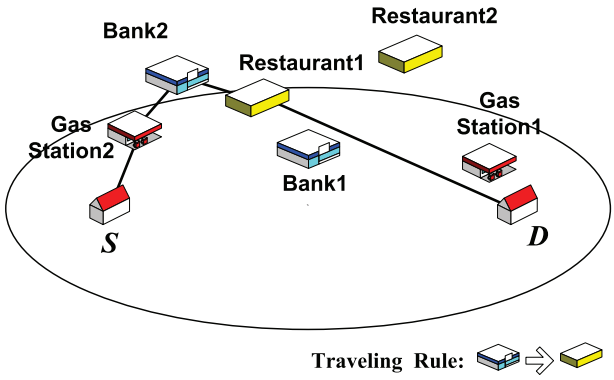


Figure 2.6: A trip search by AASPSR(2) with the traveling rule **Bank** $\rightarrow$ **Restaurant**.

2.3.5 Comparison between NNPSR, NNPSR-LORD and AASPSR

In order to analyze the performance of the three aforementioned algorithms, we have the following two definitions:

Definition A MRPSR query is called to be a *strictly constrained query* if its PCC value is relatively high.

Definition A MRPSR query is called to be a *loosely constrained query* if its PCC value is relatively low.

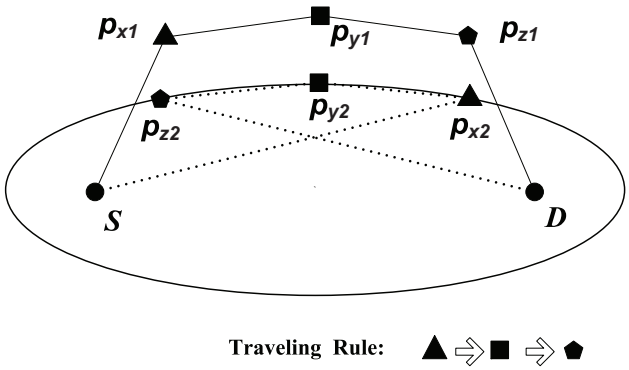


Figure 2.7: An example where AASPSR generates a longer route than NNPSR.

Theorem 2.5. *Given a MRPSR query, AASPSR does not necessarily return a shorter route than NNPSR.*

Proof. We can prove it by a counter-example as shown in Figure 2.7. In Figure 2.7, the triangle, rectangle, and pentagon each represents a different category of POIs, and S and D denote the start point and destination of the trip, respectively. Additionally, we have a traveling rule denoted as **triangle** $\rightarrow$ **rectangle** $\rightarrow$ **pentagon**. Consequently, the NNPSR created trip plan $T_1 = \{S, p_{x1}, p_{y1}, p_{z1}, D\}$ (the solid route) is shorter than the AASPSR created trip plan $T_2 = \{S, p_{x2}, p_{y2}, p_{z2}, D\}$ (the dashed route). The existence of traveling rules leads to a roundabout route in AASPSR, which only chooses the POIs inside the ellipse. Therefore, AASPSR performs worse than NNPSR in terms of route distance in this scenario. □

In each category, AASPSR(k) only chooses the k POI's which have the minimum traveling distance sum from the start point to the destination for the subsequent NN search. Consequently, if there are too many traveling rules imposed, i.e., there are many restrictions on the category order, AASPSR(k) will be very likely to generate a roundabout route. Therefore, for strictly constrained MRPSR queries, which have a higher PCC, NNPSR usually returns a shorter route than AASPSR(k). On the other hand, for loosely constrained MRPSR queries, which hold a lower PCC, such as TPQ (PCC equals zero), AASPSR(k) usually outperforms NNPSR in terms of route distance.

By taking advantage of the LORD algorithm [52], NNPSR-LORD can further shorten the length of the route retrieved by the NNPSR algorithm. Consequently, NNPSR-LORD can consistently outperform NNPSR in terms of route distance. However, as far as the response time is concerned, NNPSR-LORD, compared with NNPSR and AASPSR, needs much more computational time, especially in road networks. This is due to its extensive usage of network distance functions. Therefore, NNPSR-LORD is only applicable to non-time sensitive applications. A complete comparison among proposed algorithms is shown in Table 2.3.

	NNPSR	AASPSR with a small k	AASPSR with a large k	NNPSR-LORD
Strictly constrained MRPSR (e.g. OSR)	✓		✓	✓
Loosely constrained MRPSR (e.g. TPQ)		✓		✓
Further optimized route				✓
Time sensitive application	✓	✓		
Non-time sensitive application			✓	✓

Table 2.3: The feature comparison among proposed algorithms.

2.4 Experimental Validation

2.4.1 Experimental Setup

The experimental results are reported in this section. We implemented the NNPSR, NNPSR-LORD, and AASPSR(k) algorithms in road networks to evaluate their performances with respect to the returned *route distance* and the *response time* to generate the corresponding routes. Besides, to highlight the benefits of our three approximate approaches, we used the LORD-based brute-force solution as the baseline, which applies LORD [52] on each possible permutation of all categories to get the optimal sequenced route for each particular category sequence. For each run of the LORD-based brute-force solution, we compared the distances of all the possible optimal sequenced routes and recorded the minimum route distance and overall response time.

As we discussed before, AASPSR(k) will exhibit more characteristics of NNPSR when k increases (in particular, AASPSR(k) degrades to NNPSR if $k = \infty$) and show more characteristics of AASPSR(1) when k decreases. Therefore, in this section we only focused on the performance of AASPSR(1) (the terms AASPSR and AASPSR(1) are used interchangeably

in this section). We varied the following parameters to obtain their effects on the route distance and response time: the Percentage of the Constrained Categories (PCC), the average category cardinality, and the number of query categories. PCC describes the percentage of the number of categories involved in traveling rules over the total number of categories to be visited in a query. The average category cardinality is the average number of POIs over all categories while the number of query categories is the total number of categories to be visited in the query. For each result of the NNPSR, AASPSR and NNPSR-LORD algorithms, 100 MRPSR queries were launched with a starting point and a destination generated randomly on the road network, and then the results were averaged. All the experiments were conducted on a Linux machine with an Intel Core2 Quad CPU (Q9400 2.66GHz) and 4GB memory.

Fig. 2.8(a) Road network of California.

Fig. 2.8(b) California points of interest distribution.

Figure 2.8: Real Datasets from the state of California.

Road Network Dataset

To investigate the performance of our proposed algorithms for road networks, first we obtained the road network dataset of the state of California from [1]. As shown in Figure 2.8(a), the road network of California contains 21,048 nodes and 22,830 edges. Each node is described with a tuple of $\langle Node\_ID, Longitude, Latitude\rangle$ and each edge is represented by a tuple of $\langle Edge\_ID, Start\_Node\_ID, End\_Node\_ID, L2\_Distance\rangle$.

California Point of Interest Dataset

We collected the points of interest of the state of California from [2] as shown in Figure 2.8(b). This California dataset has 63 different categories, including airports, hospitals, schools, populated places, etc., which correspond to more than 100,000 points of interest. Each category exhibits a distinct density and distribution. Each point of interest is represented as a tuple of $\langle Category\_Name, Longitude, Latitude \rangle$. The cardinalities of all the categories used in our research is shown in Table 2.4.

To merge the points of interest in the real dataset with the road network, we adopted the map format where each point of interest was at first mapped to a point on an edge and then represented as the distance of this point to the start node of that edge.

Synthetic Point of Interest Datasets

To control different cardinalities and distributions of categories, we also applied synthetic datasets in our experiments. We generated different numbers of points of interest for different datasets and uniformly distributed the points of interest on the edges of the California road networks.

Traveling Rules

Without loss of generality, for the real dataset, rules were generated between Building and Populated place, Church and Hospital, and Locale and Park, i.e., rules can be represented as follows: *Building* $\rightarrow$ *Populated place*, *Church* $\rightarrow$ *Hospital* and *Locale* $\rightarrow$ *Park*. For the synthetic datasets, rules were generated between any two arbitrary categories at random.

2.4.2 Effect of the Percentage of the Constrained Categories

In our first experiment, we varied the Percentage of the Constrained Categories (PCC) to investigate the performance of NNPSR, AASPSR, NNPSR-LORD and the LORD-based brute-force solution in terms of the route distance and response time. Since our proposed

Category	Size
Airport	995
Area	287
Bar	278
Building	4110
Church	7680
Hospital	835
Locale	13481
Park	6728
School	11173
Populated place	6900
Summit	5594
Valley	7596

Table 2.4: The category cardinalities used in our California dataset.

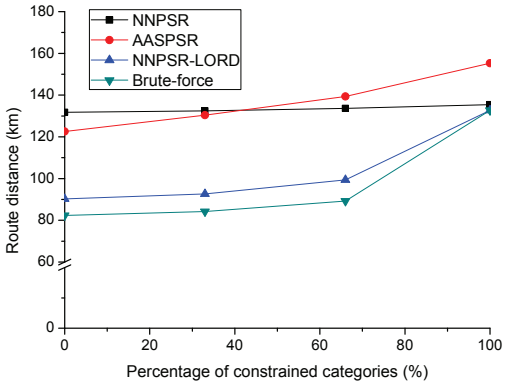


Fig. 2.9(a) California dataset

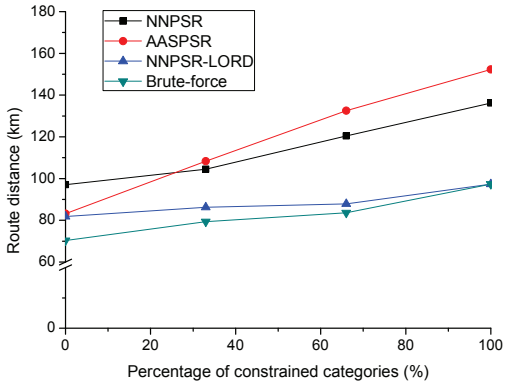


Fig. 2.9(b) Synthetic dataset

Figure 2.9: Route distance of NNPSR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of PCC.

MRPSR query subsumes the TPQ and OSR queries, the MRPSR queries exhibit the characteristics of TPQ queries when PCC decreases and the characteristics of OSR queries when PCC increases. Our results are based on the California POI dataset and the synthetic POI datasets, respectively. In the synthetic POI dataset, the average category cardinality is 6000. Furthermore, we assumed that the number of query categories is 6. Figure 2.9 illustrates the relationship between route distance and PCC for NNPSR, AASPSR, NNPSR-LORD, and LORD-based brute-force algorithms.

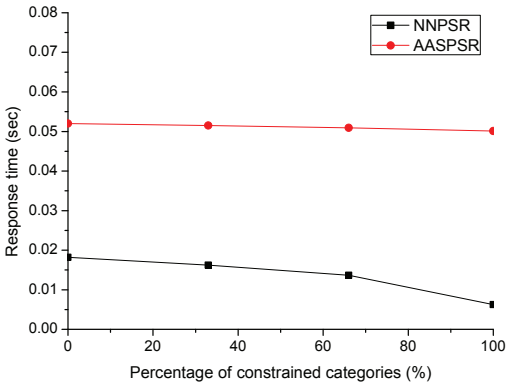


Fig. 2.10(a) NNPSR and AASPSR (California dataset)

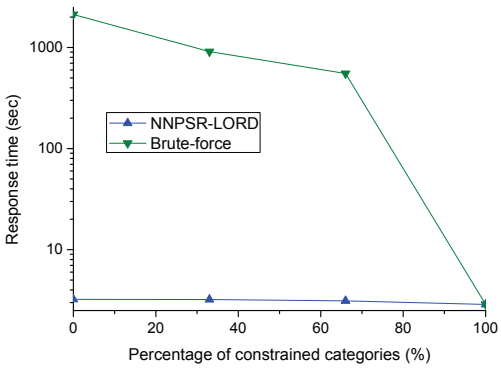


Fig. 2.10(b) NNPSR-LORD and LORD-based brute-force (California dataset)

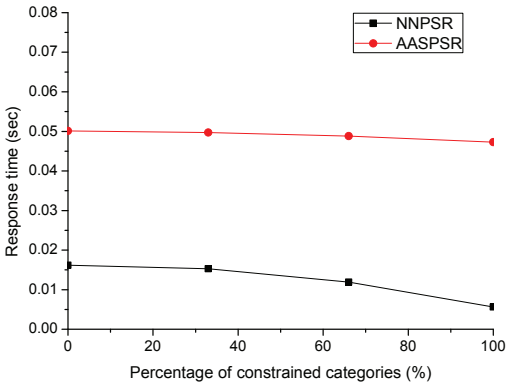


Fig. 2.10(c) NNPSR and AASPSR (synthetic dataset)

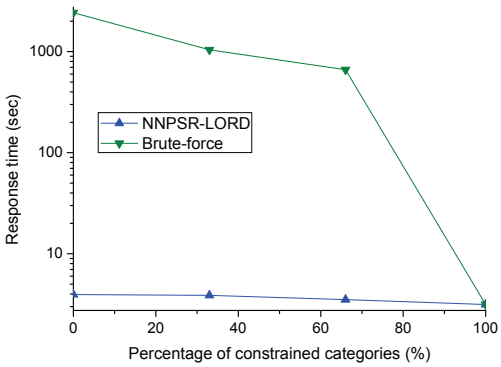


Fig. 2.10(d) NNPSR-LORD and LORD-based brute-force (synthetic dataset)

Figure 2.10: Response time of NNSPR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of PCC

In Figure 2.9 route distance increases with the increase of PCC for all the algorithms. This is because with a higher PCC, there will be more restrictions on the order of the categories, which leads to a longer route. Note that the route distance of AASPSR changes remarkably against PCC in contrast to NNPSR and NNPSR-LORD. The lower PCC is, the better AASPSR works compared with NNPSR. With a higher PCC, the route distance of

AASPSR increases dramatically. In other words, AASPSR is only suitable for planning a trip with a low PCC, such as TPQ (PCC equals zero). This is because AASPSR only picks a single POI in each category for the subsequent NN search. Consequently, given a higher PCC (there are more restrictions on the category order), a longer route may be needed to traverse all the POIs picked up in the first step. In addition, NNPSR-LORD outperforms NNPSR and AASPSR in terms of route distance given any PCC. The reason is that NNPSR-LORD employs LORD to obtain the shortest route under the specific order of categories in the route found by NNPSR.

Figure 2.10 plots the response time against PCC for NNPSR, AASPSR, NNPSR-LORD, and the LORD-based brute-force method. Notice that in Figure 2.10 (b) and (d), we plotted the relationship by using the *log* value of the response time instead of the original value because the response times of NNPSR-LORD and LORD-based brute-force solutions are in different orders of magnitude. First, as shown in Figure 2.10, all our proposed algorithms significantly reduce the response time compared with the LORD-based brute-force solution. To be specific, NNPSR and AASPSR are more than 10000 times faster in some cases than the LORD-based brute-force method while NNPSR-LORD is more than 100 times faster. Second, the response times decrease with the increase of PCC. This is because a higher PCC will decrease the search space of POIs.

2.4.3 Effect of the Average Category Cardinality

Next, we studied the effect of the average category cardinality by varying the cardinality from 2000 to 14000 using synthetic datasets. Here we assumed that the number of query categories is 6. Figure 2.11 shows the route distances of NNPSR, AASPSR, NNPSR-LORD, and the LORD-based brute-force method where PCC equals 33% and 66%, respectively. As Figure 2.11 shows, the route distance decreases for each algorithm with the increase of the average category cardinality. The reason is that a denser distribution of a category will lead to more POI choices, which result in a lower probability of detours. Notice that AASPSR has

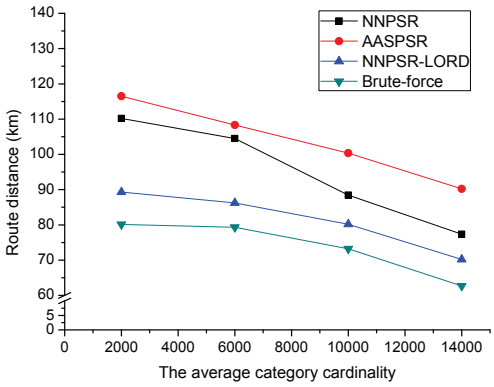


Fig. 2.11(a) PCC = 33%

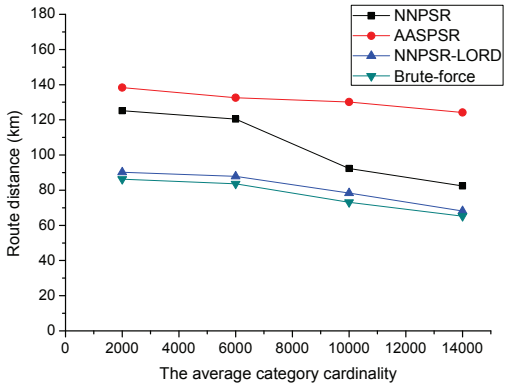


Fig. 2.11(b) PCC = 66%

Figure 2.11: Route distance of NNSPR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of the average category cardinality.

relatively poor performance with either a PCC of 33% or 66% because AASPSR outperforms NNPSR in terms of route distance only if PCC is very low. The PCC of 33% or 66% is large enough to deteriorate the performance of AASPSR. Figure 2.12 shows the response time for the above algorithms. As Figure 2.12 demonstrates, the response time of each algorithm increases with the enlargement of the average category cardinality. This is due to a higher density of each POI category which elongates the computational time.

2.4.4 Effect of the Number of Query Categories

In this subsection, we changed the number of query categories to 3, 6, 9 and 12 to investigate the impact of the number of query categories on the performance of NNPSR, AASPSR, NNPSR-LORD, and the LORD-based brute-force method. Our experiments are based on the California POI and the synthetic POI datasets, respectively. In the synthetic POI dataset, the average category cardinality is assumed to be 6000. In addition, we assume that PCC equals 66%. Figures 2.13 and 2.14 illustrate the experimental results. As shown in Figure 2.13, when the number of query categories increases, the route distance of each algorithm extends dramatically. This is because with an increasing number of categories to be visited, there will be more POIs to be traversed in a trip. Notice that the number

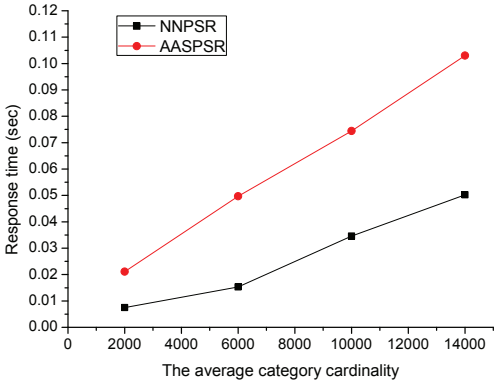


Fig. 2.12(a) NNPSR and AASPSR (PCC = 33%)

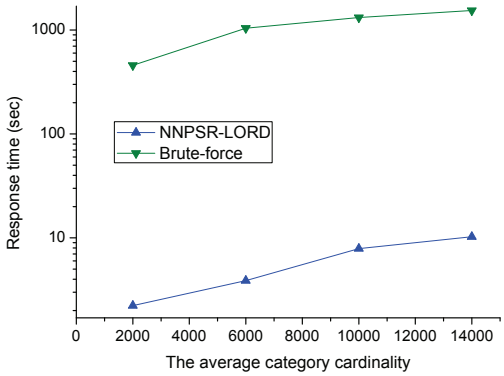


Fig. 2.12(b) NNPSR-LORD and LORD-based brute-force (PCC = 33%)

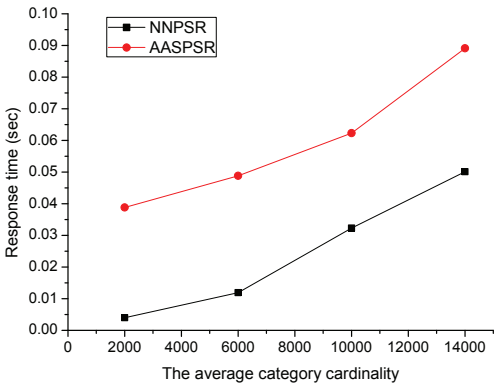


Fig. 2.12(c) NNPSR and AASPSR (PCC = 66%)

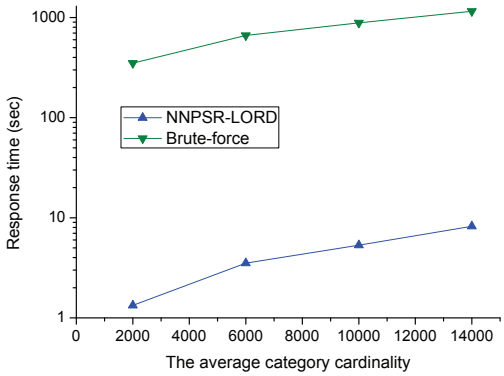


Fig. 2.12(d) NNPSR-LORD and LORD-based brute-force (PCC = 66%)

Figure 2.12: Response time of NNSPR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of the average category cardinality.

of query categories has a significant impact on whether AASPSR outperforms NNPSR in terms of route distance. For three-category cases, AASPSR returns a shorter route distance than NNPSR. On the contrary, for the 6, 9, or 12 category cases, AASPSR reports a longer route than NNPSR. The reason is that fewer categories will lead to a lower probability for

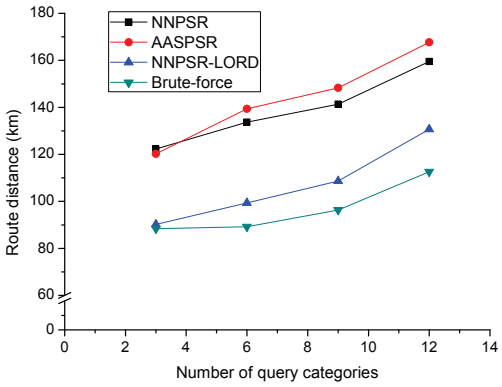


Fig. 2.13(a) California dataset

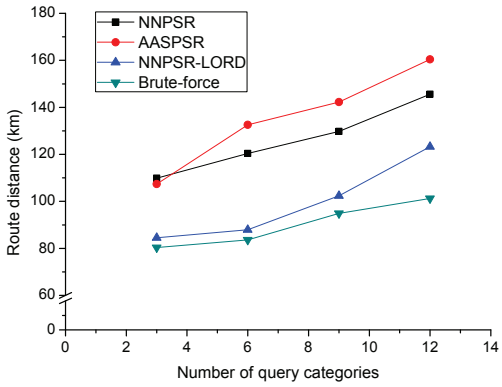


Fig. 2.13(b) Synthetic dataset

Figure 2.13: Route distance of NNPSR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of the number of query categories.

a detour to occur when AASPSR tries to traverse all the POIs selected in its first step. On the other hand, as Figure 2.14 shows, when the number of query categories increases, the response time prolongs accordingly. The reason is that all the algorithms need more time to compute more categories to answer a MRPSR query. In particular, AAPSR consistently needs more response time than NNPSR, irrespective of the number of query categories.

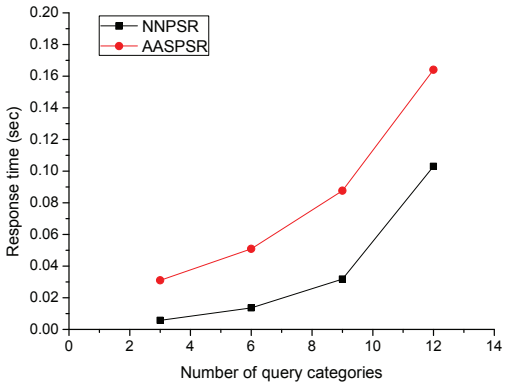


Fig. 2.14(a) NNPSR and AASPSR (California dataset)

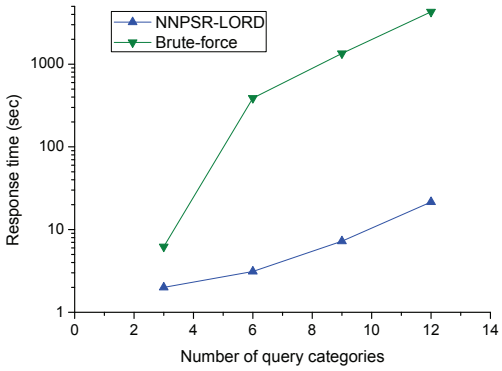


Fig. 2.14(b) NNPSR-LORD and LORD-based brute-force (California dataset)

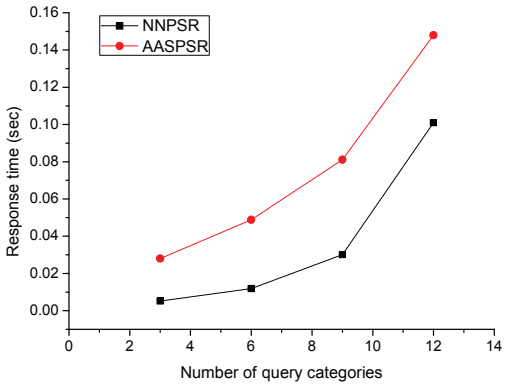


Fig. 2.14(c) NNPSR and AASPSR (synthetic dataset)

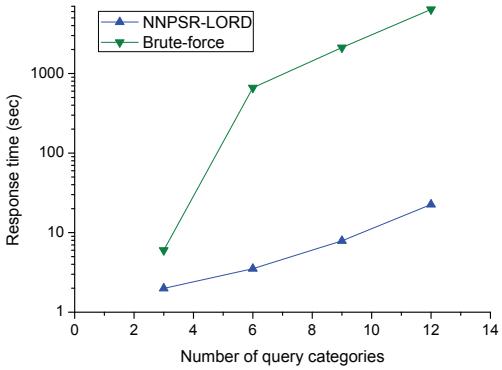


Fig. 2.14(d) NNPSR-LORD and LORD-based brute-force (synthetic dataset)

Figure 2.14: Response time of NNSPR, AASPSR, NNPSR-LORD, and LORD-based brute-force as a function of the number of query categories.

Algorithm 1 Nearest Neighbor-based Partial Sequenced Route query($\mathbb{C}$, $\mathbb{R}$, S, D)

Set $L_{route} = \emptyset$ and $q = S$
Integrate all elements in $\mathbb{R}$ into an AOV adjacency list $\mathbb{A}$ and put all vertices with zero count in L_{zero}
if The AOV network is a DAG **then**
  Add all elements of $\mathbb{C} \setminus \mathbb{A}$ into L_{zero}
  while $L_{zero} \neq \emptyset$ **do**
    $\mathbb{P} = \emptyset$
    for each $C_i \in L_{zero}$ **do**
      $\mathbb{P} = C_i.\mathbb{P} \cup \mathbb{P}$
    end for
    Identify the road segment $n_i n_j$ covering q.
    Find all the POIs in $\mathbb{P}$ on $n_i n_j$.
    if If there exists at least one POI in $\mathbb{P}$ on $n_i n_j$ **then**
      Update p_{NN} with the POI P_k with the smallest $Dist_N(q, P_k)$.
    else
      $Q =< (n_i, Dist_N(q, n_i)), (n_j, Dist_N(q, n_j)) >$
      De-queue the node n in Q with the smallest $Dist_N(q, n)$
      while $Dist_N(q, n) < Threshold$ **do**
        for each non-visited adjacent node n_k of n **do**
          Find all the POIs in $\mathbb{P}$ on the road segment nn_k.
          Update P_{NN} from the POI p' in $\mathbb{P}$ with the smallest network distance found so far
          Update $Threshold$ with $Dist_N(q, p')$
          En-queue $(n_k, Dist_N(q, n_k))$ in Q
        end for
        De-queue the node n in the updated Q with the smallest $Dist_N(q, n)$
      end while
    end if
    $q = P_{NN}$
    $L_{route} = L_{route} \cup P_{NN}$
    Remove $P_{NN}.C$ from L_{zero}
    Update $\mathbb{A}$ and L_{zero}
  end while
  return L_{route}
else
  Report cycles in $\mathbb{R}$
end if

Algorithm 2 Advanced A* Search-based Partial Sequenced Route query($\mathbb{C}$, $\mathbb{R}$, S, D, k)

Set $L_{route} = \emptyset$ and $Q = S$
Integrate all elements in $\mathbb{R}$ into an AOV adjacency list $\mathbb{A}$ and put all vertices with zero count in L_{zero}
if The AOV network is a DAG **then**
  Add all elements of $\mathbb{C} \setminus \mathbb{A}$ into L_{zero}
  for each category $C_i \in \mathbb{C}$ **do**
    for each POI $p_j \in C_i.\mathbb{P}$ **do**
      $Cost_j = Dist_E(S, p_j) + Dist_E(p_j, D)$
    end for
    Sort POI $p_j \in C_i$ in ascending order based on $Cost_j$ and add the top-k $p_j \in C_i$ with the minimum $Cost_j$ into $C_i.\mathbb{P}^*$
  end for
  {ASPSR search done, the subsequent NNPSR search starts}
  while $L_{zero} \neq \emptyset$ **do**
    $\mathbb{P} = \emptyset$
    for each $C_i \in L_{zero}$ **do**
      $\mathbb{P} = \mathbb{P} \cup C_i.\mathbb{P}*$
    end for
    Identify the road segment $n_i n_j$ covering q.
    Find all the POIs in $\mathbb{P}$ on $n_i n_j$.
    if If there exists at least one POI in $\mathbb{P}$ on $n_i n_j$ **then**
      Update P_{NN} with the POI P_k with the smallest $Dist_N(q, P_k)$.
    else
      $Q =< (n_i, Dist_N(q, n_i)), (n_j, Dist_N(q, n_j)) >$
      De-queue the node n in Q with the smallest $Dist_N(q, n)$
      while $Dist_N(q, n) < Threshold$ **do**
        for each non-visited adjacent node n_k of n **do**
          Find all the POIs in $\mathbb{P}$ on the road segment nn_k.
          Update P_{NN} from the POI p' in $\mathbb{P}$ with the smallest network distance found so far
          Update $Threshold$ with $Dist_N(q, p')$
          En-queue $(n_k, Dist_N(q, n_k))$ in Q
        end for
        De-queue the node n in the updated Q with the smallest $Dist_N(q, n)$
      end while
    end if
    $q = P_{NN}$
    $L_{route} = L_{route} \cup P_{NN}$
    Remove $P_{NN}.C$ from L_{zero}
    Update $\mathbb{A}$ and L_{zero}
  end while
  return L_{route}
else
  Report cycles in $\mathbb{R}$
end if

Chapter 3

Approximate Query Evaluation on RFID Databases

3.1 Bayesian Inference

Symbol	Meaning
$\hat{H}$	The random vector representing locations of all objects
h_i	The random variable representing the location of o_i
$\mathbb{Z}$	Raw data reported by RFID readers
z_{ij}	The raw data (0 or 1) reported by the reader in zone j for object o_i
$post(\hat{H}\|\mathbb{Z})$	The posterior probability of the location vector $\hat{H}$ given the raw data $\mathbb{Z}$
$p(z_{ij}\|h_i)$	The likelihood that the zone j reader reports the value of z_{ij} for object o_i given that object o_i is in the zone h_i
$p(h_j)$	The prior probability that object o_j is in the zone h_j

Table 3.1: Symbolic notations of Section 3.1.

$$
\begin{aligned}
post(\hat{H}|\mathbb{Z}) &= post(h_1, h_2, \ldots, h_n| \begin{bmatrix} z_{11} & \cdots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nm} \end{bmatrix}) \\
&\propto p(\begin{bmatrix} z_{11} & \cdots & z_{1m} \\ \vdots & \ddots & \vdots \\ z_{n1} & \cdots & z_{nm} \end{bmatrix} |h_1, h_2, \ldots, h_n) \cdot p(h_1, h_2, \ldots, h_n)
\end{aligned}
\tag{3.1}
$$

$$post(\hat{H}|\mathbb{Z}) \propto \prod_i p(z_{i1}, z_{i2}, \ldots, z_{im}|h_1, h_2, \ldots, h_n) \cdot p(h_1, h_2, \ldots, h_n) \tag{3.2}$$

$$post(\hat{H}|\mathbb{Z}) \propto \prod_i p(z_{i1}|h_i) \cdot p(z_{i2}|h_i) \cdot \ldots \cdot p(z_{im}|h_i) \cdot \prod_j p(h_j) \tag{3.3}$$

In this section, we develop a Bayesian inference-based approach to handle redundant readings and prior knowledge, and we analyze the challenges when applying this approach. Table 3.1 summarizes the notations used in this section.

3.1.1 A Bayesian Inference Based Approach

Bayesian inference is a statistical inference technique that estimates the probability of a hypothesis (x) based on observations (y). Bayesian inference shows that posterior is proportional to the multiplication of likelihood and prior, which can be represented as $p(x|y) \propto p(y|x)p(x)$.

Suppose there are m zones and n objects in our monitoring environment, each zone with a reader mounted in the zone center. Let o_i represent the object with ID i. For each o_i, its location is represented by a random variable h_i. Therefore, a possible distribution of n objects in m zones can be denoted as an instance of the random vector $\hat{H} = (h_1, h_2, \ldots, h_n)$. h_i represents the zone ID where object o_i is in. For example $h_1 = 2$ denotes that object o_1 is in zone 2 in the current instance. For the reader in zone j, the raw data (0 or 1) it receives from the RFID tag of object o_i is denoted as z_{ij}. The raw data matrix for each complete scan from m readers can then be represented as an $n \times m$ matrix $\mathbb{Z} = [z_{ij}]$. Thus the Bayes' theorem can be represented as Equation 3.1, where $post(\hat{H}|\mathbb{Z})$ denotes the posterior probability of location vector $\hat{H}$ given the raw data $\mathbb{Z}$, and a valid hypothesis means the hypothesis satisfies all constraints:

$$post(\hat{H}|\mathbb{Z}) = \alpha \cdot \prod_i p(z_{i1}|h_i) \cdot p(z_{i2}|h_i) \cdot \ldots \cdot p(z_{im}|h_i) \cdot \prod_j p(h_j) \tag{3.4}$$

$$\begin{aligned}
&post(\hat{H}|\mathbb{Z}) = 0 && : \hat{H} \text{ is not valid} \\
&post(\hat{H}|\mathbb{Z}) > 0 && : \hat{H} \text{ is valid} \\
&post(\hat{H}_1|\mathbb{Z}) > post(\hat{H}_2|\mathbb{Z}) && : \hat{H}_1 \text{ is more likely than } \hat{H}_2
\end{aligned}$$

In particular, if $z_{ij} = 1$ in a raw data matrix and the actual location of object o_i is not in zone j, then z_{ij} is a false positive. Take Table 1.1 as an example, at least one of z_{22} and z_{23} is a false positive because object 2 cannot be in zone 2 and zone 3 simultaneously. Similarly, at least one of z_{43} and z_{44} is a false positive.

To compute $post(\hat{H}|\mathbb{Z})$, we make some independence assumptions of random variables. RFID reader transmissions or tag transmissions may lead to collisions because readers and tags communicate over a shared wireless channel. Reader collisions happen when neighboring readers communicate with a tag simultaneously [18] and tag collisions occur when multiple tags transmit to a reader at the same time [20]. However, the two kinds of collisions can be effectively prevented by arbitration protocols (e.g. by scheduling adjacent readers to operate at different times) [26, 42, 64]. Therefore, we assume each reader detects the tags of different objects independently (i.e., whether a reader can successfully detect the tag of a certain object does not interfere with whether the reader can successfully detect that of another object) in this research. Based on the assumption, we can derive Equation 3.2.

Furthermore, because we employ MH-C to take into account constraints (i.e., to ensure that each generated sample satisfies all the constraints), here we can simply assume independence between different h_i (i.e., the locations of objects). In addition, we assume that each reader's detection of the same object is independent. Besides, the prior distribution of each object does not depend on that of other objects. Therefore, we can obtain Equation 3.3. If we rewrite Equation 3.3 using the normalizing constant, denoted as α, we can reach Equation 3.4, which shows how to compute the posterior of each sample. To be specific, $p(z_{ij}|h_i)$

reflects the corresponding *likelihood*, which is the probability that the reader in zone j reports the value of z_{ij} about object o_i given that object o_i is actually in the zone with ID h_i. Furthermore, $p(h_j)$ denotes the *prior probability* that the object o_j is in the zone with the ID of h_j. The prior probability can be interpreted as the assumed distribution before the acquiring of the RFID raw data.

3.1.2 The Goal and the Obstacles

Based on Equation 3.4, given the raw readings $\mathbb{Z}$ and a hypothesis $\hat{H}$ (the location of each object), we can derive the probability of the hypothesis. However, finding just one valid hypothesis will provide nothing more than a biased answer to queries against the uncertain data. To address this issue, we need to query against all valid hypotheses. However, this is unrealistic because there are numerous valid hypotheses in most cases. Thus, our goal is to create a large sample set of valid hypotheses, each associated with a weight computed by Equation 3.4: $(\hat{H}_1, w_1), (\hat{H}_2, w_2), \cdots, (\hat{H}_n, w_n)$. The sample set of valid hypotheses as a whole enables us to answer queries with high credibility. To achieve this goal, we must overcome the following obstacles:

- A prerequisite for effective hypothesis sampling is to be able to compute the posterior probability of each hypothesis precisely. Therefore, we propose the n-state detection model in Section 3.2 to capture likelihoods in an affordable and accurate way.

- The hypothesis space is high dimensional, and the posterior probability is difficult to sample from. Thus, we need a sampling technique that has desirable efficiency. We apply a Markov Chain Monte Carlo method (MCMC) because MCMC can maintain the correlation between samples, resulting in an improved sampling efficiency.

- We need to incorporate constraint management in sampling. We propose a sampler called Metropolis-Hastings sampler with Constraints (MH-C), which improves the

naive Metropolis-Hastings (MH) sampler. Each sample generated by MH-C automatically satisfies all the constraints.

3.2 RFID Reader Detection Models

The major difficulty in computing the posterior of each sample (Equation 3.4) lies in how to accurately estimate the likelihood $p(z_{ij}|h_i)$. To do so, we introduce the n-state detection model to capture likelihood in an affordable and precise way.

3.2.1 Physical Characteristics

An RFID reader sends RF signals to communicate with (passive) tags to retrieve a list of IDs in its detection range. For a typical warehouse application, the frequency channel of 915 MHz is usually employed for the communication because of its long detection range, which is about 10-20 feet [66]. However, RFID data acquisition and transmission are unreliable [20, 29,31,44]. In our experiment, we investigated the change of the read rate over distance using regular RFID readers and tags. The results are illustrated in Figure 3.1. The model of the tags is Gen2 RFID Smart Label and the model of the reader is MPR-6000 (antenna 902-928 MH), provided by WJ Communications Inc. Our test environment is a lab with many metal objects (tables, desks and computer equipment), representing a noisy environment.

As Shown in Figure 3.1, the overall detection range of a reader can be separated into the major detection region and the minor detection region, where in the major detection region from 0 to almost 5 feet, the read rate can keep a level of around 95% and in the minor detection region approximately from 5 to 13 feet, the read rate drops off almost linearly. Furthermore, the read rate deteriorates to zero in the region more than 13 feet away from the reader, which is defined as beyond the overall detection range [31]. The read rate falls with the increase in the distance, which is similar to the scenario where the strength of a wireless signal fades as the signal propagates in an open space.

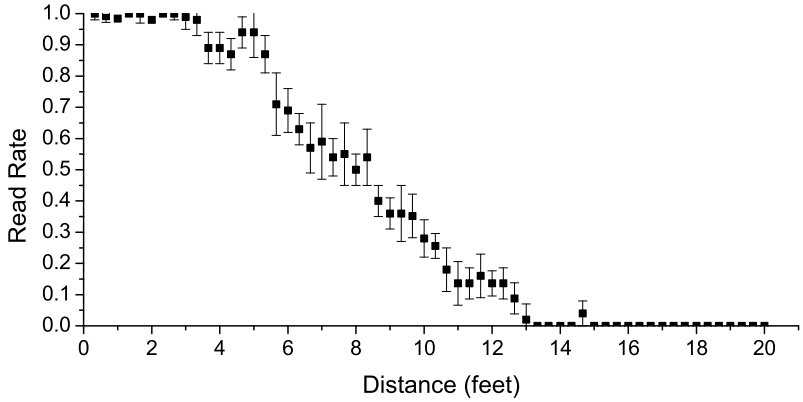


Figure 3.1: An illustration of the relationship between read rate and distance.

3.2.2 Problems of the 2-State Detection Model

Taking the scenario in Figure 1.2 as an example, one way to estimate likelihood is as follows:

$$p(z_{ij} = 1|h_i) = \begin{cases} r & \text{if } h_i \in \{j-1, j, j+1\} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

$$p(z_{ij} = 0|h_i) = \begin{cases} 1-r & \text{if } h_i \in \{j-1, j, j+1\} \\ 1 & \text{otherwise} \end{cases} \tag{3.6}$$

where r is the average read rate. Intuitively, it means that an object o_i holds the same probability to be detected by a reader whether o_i is in the zone (j) the reader is associated to, or in any of neighboring zones ($j-1$ or $j+1$). Apparently, if compared with Figure 3.1, this 2-state detection model is inherently inaccurate as it fails to capture any change of the read rate in the overall detection region. Consequently, when predicting the location of an object, the resulting system is unable to differentiate between its own zone and all its neighboring zones because all the above zones are with an identical read rate.

In order to solve this problem, current works are forced to adopt a simplified 2-state detection model, which is shown in Figure 3.2. This simplified 2-state model assumes readers' detection regions do not overlap, i.e., a reader is only able to detect the objects in its own zone. Then the likelihood can be estimated as follows:

$$p(z_{ij} = 1|h_i) = \begin{cases} r & \text{if } h_i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$p(z_{ij} = 0|h_i) = \begin{cases} 1 - r & \text{if } h_i = j \\ 1 & \text{otherwise} \end{cases} \tag{3.8}$$

This simplified 2-state model, however, has two problems. First, it is unrealistic to assume that we can divide the space into non-overlapping detection regions. Second, the model does not support applications that use redundant information (such as redundant readings) to offset the unreliability of raw RFID readings.

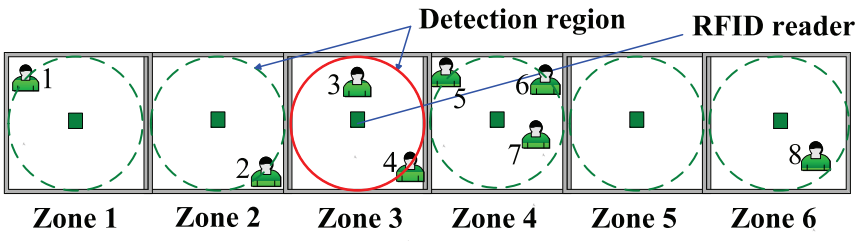


Figure 3.2: An illustration of the simplified 2-state model.

3.2.3 The n-state Detection Model

In order to take full advantage of duplicate readings, we propose an n-state detection model, which is illustrated in Figure 3.3. In Figure 3.3, the overall detection region of an RFID reader is divided into several sub-regions, each of which corresponds to a zone associated with a unique read rate. As far as a specific detection model is concerned, the difference in the read rate over any two adjacent sub-regions is a constant, i.e., the read rates for different states constitute an arithmetic sequence. Take the 4-state model as an

example. Suppose the highest read rate in the model is x. The first state (counted with the increase of the detection distance) holds a read rate of x, the second state keeps a read rate of $\frac{2x}{3}$, the third state maintains a read rate of $\frac{x}{3}$, and the fourth state eventually has a read rate of zero. Thus, as for a specific reader, by employing the n-state detection model, each correlated zone is assigned a distinct read rate according to its distance to this reader. In particular, the simplest model in the family of the n-state detection model is the 2-state model, where an identical read rate is assumed in the overall detection region of each reader.

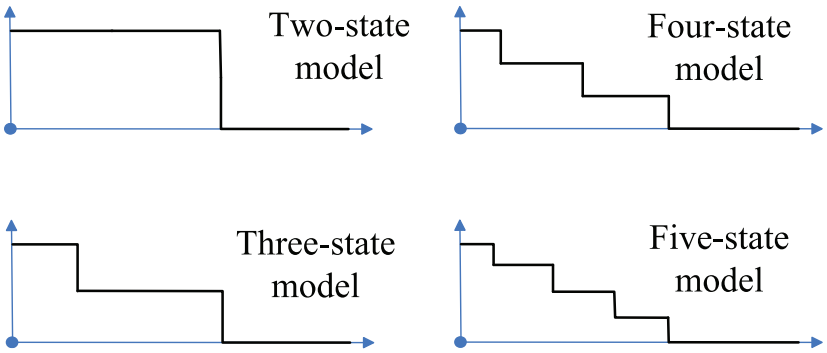


Figure 3.3: The family of the n-state detection model.

Notice that in practice, the n value depends on how zones are divided in overall detection regions of RFID readers. This is because an n-state model in fact implies that every $2(n-2)+1 = 2n-3$ zones correlate with each other (assuming that all the zones are in a 1-dimensional distribution). For example, if it is known as prior knowledge that one object can be read simultaneously by up to five readers, we have to choose $n = 4$ to incorporate the correlation among every 5 successive zones.

3.2.4 A Case Study: The 3-State Model

We elaborate on the 3-state model as a case study. Suppose one reader can only detect its own zone and the two neighboring zones. This assumption implies that, as for a particular reader, there are three distinct location-based states of a object: in the same zone as the reader, in the neighboring zones, and in all the other zones. To capture this correlation,

we have to choose $n = 3$ and the resulting model is the *3-state detection model*, where the overall detection range of a reader is divided into two sub-regions, as shown in Figure 3.4. Specifically, the major detection region, the minor detection region and the zero read rate region in Figure 3.4 correspond to the zone where the reader locates, neighboring zones and all the other zones, respectively. Therefore, the motivating scenario (Figure 1.2), if interpreted by the 3-state model, can be illustrated in Figure 3.5.

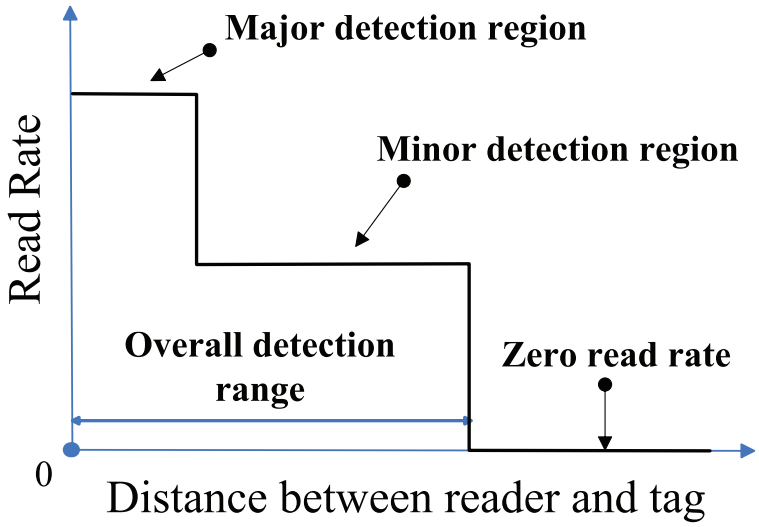


Figure 3.4: The 3-state detection model of RFID readers.

In Figure 3.5, by using the 3-state detection model, not only the duplicate readings can be incorporated, but also a zone and all its neighboring zone can be differentiated because they are with distinct read rates. To be specific, if object o_i is in zone j, not only z_{ij} but also $z_{i(j-1)}$ and $z_{i(j+1)}$ should have a considerable chance to be 1 (false positives). In the meantime, other readers are unable to detect the tag of object o_i. The reason is that object o_i may be in the major detection region of the reader in zone j while it may also be in the minor detection region of both readers in zones $j-1$ and $j+1$. As for the other readers, object o_i is totally beyond their overall detection regions, leading those readers to report 0 for object o_i. If we denote the mean value of the read rate in major detection region as r_{major} and the mean value of the read rate in minor detection region as r_{minor}, the estimate of the likelihood using the 3-state model can be represented in Equation 3.9 and Equation 3.10.

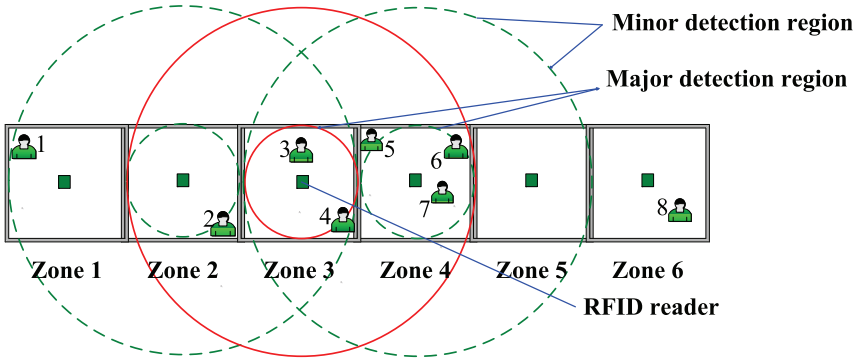


Figure 3.5: The detection region overlap interpreted by the 3-state detection model.

$$p(z_{ij}=1|h_i)=\begin{cases} r_{major} & \text{if } h_i=j \\ r_{minor} & \text{if } h_i\in\{j-1,j+1\} \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

$$p(z_{ij}=0|h_i)=\begin{cases} 1-r_{major} & \text{if } h_i=j \\ 1-r_{minor} & \text{if } h_i\in\{j-1,j+1\} \\ 1 & \text{otherwise} \end{cases} \tag{3.10}$$

The advantage of the 3-state detection model over the 2-state detection model can be measured by the system entropy. We also answer the question whether having even more states (more than 3) can further benefit the system.

3.3 Entropy Analysis

We use entropy to measure the uncertainty in a system after invalid system states have been eliminated by a data cleansing method. Generally, applying an efficient data cleansing method will lead to systems with smaller entropy. In this section, we first show the advantage of the 3-state model over the 2-state model and then prove that the 3-state model can maximize the system performance compared with other detection models with even more than 3 states. A snippet of the RFID raw data is shown in Table 3.2, and the actual location of an object i is denoted as a random variable L.

3.3.1 Entropy versus Read Rate

Entropy of the 2-State Model

Suppose that y denotes the read rate in the 2-state detection model. According to the right side of Equation 3.4, the probability mass function of L in the 2-state model can be represented as:

$$p(L=l) = \begin{cases} \alpha(1-y)y(1-y)\beta & \text{if } l = j \\ \alpha(1-y)(1-y)y\beta & \text{if } l \in \{j-1, j+1\} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

where α is the normalizing constant and β represents the prior probability (we assume the prior distribution as a uniform distribution) in Equation 3.4. Because the entropy H of a discrete random variable X with possible values $\{x_1, x_2, ..., x_n\}$ can be represented as

$$H(x) = -\sum_{i=1}^{n} p(x_i) \cdot \ln p(x_i),$$

where $p(x_i)$ denotes the probability mass function of X, we can calculate the entropy H of L as:

	...	Zone $j-1$ Reader	Zone j Reader	Zone $j+1$ Reader	...
...	...	...	...	...	...
Obj i	...	0	1	0	...
...	...	...	...	...	...

Table 3.2: A snippet of the RFID raw data.

$$
\begin{aligned}
H(L) = \quad & -\alpha(1-y)(1-y)y\beta \cdot \ln(\alpha(1-y)(1-y)y\beta) \\
& -\alpha(1-y)y(1-y)\beta \cdot \ln(\alpha(1-y)y(1-y)\beta) \\
& -\alpha(1-y)(1-y)y\beta \cdot \ln(\alpha(1-y)(1-y)y\beta)
\end{aligned} \tag{3.12}
$$

Because the probabilities on all the locations sum to 1, we can derive Equation 3.13. By applying Equation 3.13 to Equation 3.12 (α and β are canceled out), we can obtain Equation 3.14.

$$
\alpha\beta = \frac{1}{3(1-y)^2 y} \tag{3.13}
$$

$$
H(L) = -3 \cdot \frac{1}{3} \cdot \ln \frac{1}{3} = 1.098 \tag{3.14}
$$

Entropy of the 3-State Model

Figure 3.5 corresponds to the 3-state model scenario. Suppose x is the read rate in the major detection region. Then the read rate in the minor detection region can be denoted as $x/2$. Thus, according to the right side of Equation 3.4, the probability mass function of L can be represented as follows:

$$
p(L = l) = \begin{cases} \alpha(1-\frac{x}{2})x(1-\frac{x}{2})\beta & \text{if } l = j \\ \alpha(1-\frac{x}{2})(1-x)\frac{x}{2}\beta & \text{if } l \in \{j-1, j+1\} \\ 0 & \text{otherwise} \end{cases}
$$

Similarly, α is the normalizing constant and β represents the prior probability in Equation 3.4. Therefore, we can calculate the entropy H of L as:

$$
\begin{aligned}
H(L) = \quad & -\alpha(1-\frac{x}{2})(1-x)\frac{x}{2}\beta \cdot \ln(\alpha(1-\frac{x}{2})(1-x)\frac{x}{2}\beta) \\
& -\alpha x(1-\frac{x}{2})^2\beta \cdot \ln(\alpha x(1-\frac{x}{2})^2\beta) \\
& -\alpha(1-\frac{x}{2})(1-x)\frac{x}{2}\beta \cdot \ln(\alpha(1-\frac{x}{2})(1-x)\frac{x}{2}\beta)
\end{aligned} \tag{3.15}
$$

Since probabilities on all locations sum to 1, we can obtain Equation 3.16.

$$
\alpha\beta = \frac{1}{x(1-\frac{x}{2})(2-\frac{3x}{2})} \tag{3.16}
$$

Combining Equation 3.16 and Equation 3.15, we have:

$$
H(L) = -2 \cdot \frac{1-x}{4-3x} \cdot \ln\frac{1-x}{4-3x} - \frac{2-x}{4-3x} \cdot \ln\frac{2-x}{4-3x}
$$

In Figure 3.6, we plot the relationship between the reconstruction entropy and read rate under the 2-state and 3-state models, respectively. As Figure 3.6 illustrates, the entropy decreases accordingly with the increase of read rate, which indicates that the system will have less uncertainty with more reliable readers. Moreover, Figure 3.6 shows that the entropy in the 3-state model is always smaller than that in the 2-state model. For example, if $x = 0.95$, the entropy in the 3-state model is 0.395 while the entropy in the 2-state model is 1.098. This observation reveals that the 3-state model can be more informative in object localization than the 2-state model.

3.3.2 Entropy versus Number of States

Here we investigate the relationship between system entropy and the number of states in a detection model. Suppose an n-state model is adopted with the highest read rate of x. Thus, the read rate in the i^{th} state (counted with the increase of the detection distance) can be represented as $\frac{(n-i)\cdot x}{n-1}$. Combined with Equation 3.4 and Table 3.2, we can obtain the

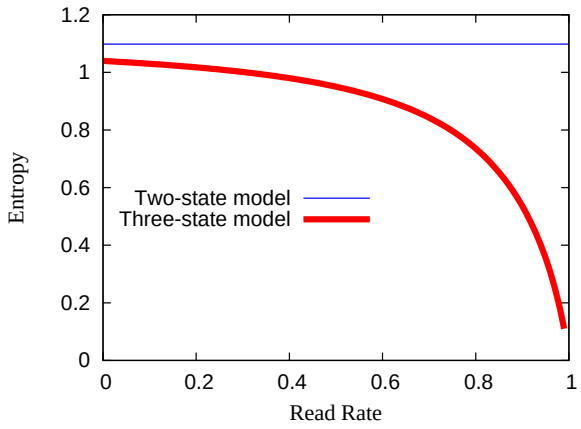


Figure 3.6: Relationship between entropy and read rate in the 2-state and 3-state models.

$$p(L=l)=\begin{cases}\alpha(1-\frac{x}{n-1})(1-\frac{2x}{n-1})...(1-\frac{(n-2)x}{n-1})\cdot x\cdot(1-\frac{(n-2)x}{n-1})...(1-\frac{2x}{n-1})(1-\frac{x}{n-1})\beta & \text{if } l=j\\ \alpha(1-\frac{x}{n-1})(1-\frac{2x}{n-1})...(1-\frac{(n-2)x}{n-1})\cdot\frac{(n-1-k)x}{n-1}\cdot(1-\frac{(n-2)x}{n-1})... & \\ (1-\frac{(n-1-k+1)x}{n-1})\cdot(1-x)\cdot(1-\frac{(n-1-k-1)x}{n-1})...(1-\frac{2x}{n-1})(1-\frac{x}{n-1})\beta & \\ \text{if } l\in\{j-k,j+k\}, k\in\{1,2,...,n-2\} & \\ 0 & \text{otherwise}\end{cases} \tag{3.17}$$

probability mass function of L, represented as Equation 3.17. Equation 3.17 shows that in an n-state model, a successful reading "1" of a certain reader about an object in fact implies that this object may exist in any of the $2(n-2)+1=2n-3$ correlated zones (including the zone which the reader is associated to), each with a non-zero probability.

Based on Equation 3.17, we plotted the relationship between entropy and the number of states in a detection model in Figure 3.7, where we assume x equals to 0.95 (the most common case). According to Figure 3.7, the 3-state detection model can minimize the system entropy and lead to the maximum system performance. In other words, having more states (more than 3) in a detection model can even deteriorate the system performance. Therefore, our experiments are mainly focused on the 3-state model.

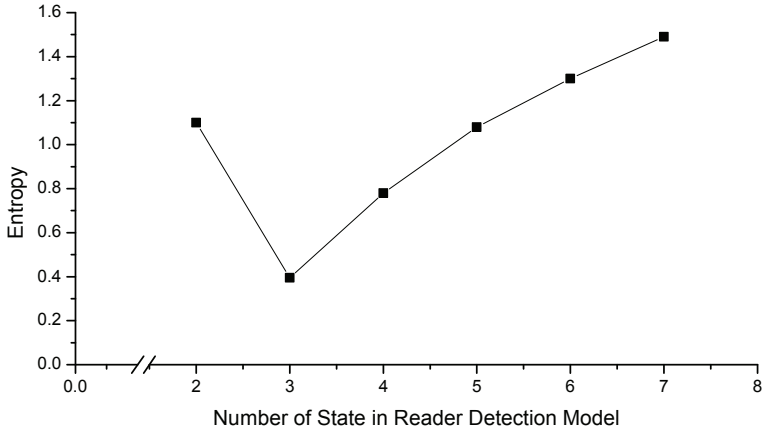


Figure 3.7: Relationship between entropy and the number of states in a detection model.

3.4 Sampling

By using Bayesian inference, we derive the posterior, as shown in Equation 3.4. Since Equation 3.4 is easy to compute but hard to sample from, we need an efficient method to draw samples from the posterior distribution. In this section, we briefly review Markov Chain Monte Carlo (MCMC) and then show why MCMC is chosen in our solution. Next, as the two most commonly used MCMC samplers, the difference between the Metropolis-Hastings sampler and the Gibbs sampler is discussed. Finally, we propose a Metropolis-Hastings sampler with Constraints (MH-C) method.

3.4.1 Preliminary

Monte Carlo Sampling

Monte Carlo Sampling is a technique to evaluate an integration of a random variable by performing statistical sampling experiments, i.e., drawing a set of samples from a target probability density function [4]. Suppose X is a random variable and its density distribution

is defined as $p(x)$. $f(x)$ is an objective function defined on X. Then, we can utilize Equation 3.18 to approximate the integration of $f(x)$, which can be interpreted as the expected value of $f(x)$ on X.

$$\int f(x)p(x)dx \backsimeq \frac{1}{N}\sum_{i} f(x^{(i)}), X \sim p(x) \tag{3.18}$$

In Equation 3.18, $f(x^{(i)})$ is the value of the objective function on $x^{(i)}$, which is the value of the random variable X on the i^{th} sampling. N denotes the total count of the sampling. Equation 3.18 reflects that the expected value can be estimated by the average of the samples using the Monte Carlo method. This evaluation is unbiased and will converge to the exact value of the integration according to the strong law of large numbers [46].

Rejection Sampling

In rejection sampling, instead of sampling directly from the target distribution $p(x)$, we sample indirectly from another proposal density distribution $q(x)$ which is easier to sample. The criterion of choosing a $q(x)$ is that its support[1] should be larger than that of $p(x)$ and Equation 3.19 can be satisfied.

$$p(x) \leq C \cdot q(x) \tag{3.19}$$

where C is an arbitrary bounding constant to guarantee that Equation 3.19 holds for any x within the support of $p(x)$. The exact sampling procedure is as follows. Suppose in the i^{th} cycle of sampling, we sample x' according to $q(x)$. Then we accept x' as $x^{(i)}$ with the probability of the acceptance rate α. If x' is accepted, we move to the next cycle. Otherwise, we obtain another sample from $q(x)$ and follow the above steps repetitively. The acceptance rate α can be computed by Equation 3.20.

[1]The support of a distribution $f(x)$ is the closure of a set X such that $f(x) \neq 0, \forall x \in X$.

$$\alpha = \frac{p(x')}{C \cdot q(x')} \tag{3.20}$$

Notice that each sample generated by rejection sampling is independent.

Importance Sampling

Instead of sampling directly from the target distribution $p(x)$, we sample indirectly from another proposal density distribution $q(x)$. Here, we only require that the support of $q(x)$ subsumes that of $p(x)$. Also, Equation 3.18 can be rewritten as Equation 3.21.

$$\int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \tag{3.21}$$

In Equation 3.21, $p(x)/q(x)$ is defined as the importance weight of the sample x, which can be denoted as $w(x)$. Consequently, the Monte Carlo estimate of an integration based on the importance sampling can be represented as Equation 3.22.

$$\int f(x)p(x)dx \backsimeq \frac{1}{N}\sum_i f(x^{(i)})w(x^{(i)}) = \sum_i f(x^{(i)})w'(x^{(i)}) \tag{3.22}$$

In Equation 3.22, $w'(x^{(i)})$ is the normalized weight of the sample $x^{(i)}$, which can be computed by Equation 3.23 [17].

$$w'(x^{(i)}) = \frac{w(x^{(i)})}{\sum_j w(x^{(j)})} \tag{3.23}$$

Like rejection sampling, all the samples generated according to importance sampling are also independent. This inhibits the possibility of using the correlation between samples to improve the sampling efficiency.

Markov Chain

A Markov chain is a stochastic process which consists of possible states of random variables [46]. It can be denoted as a sequence states of X_1, X_2, X_3, ..., X_n, which satisfy

$$\begin{aligned} &p(X_{n+1} = x | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_1 = x_1) = \\ &p(X_{n+1} = x | X_n = x_n) \end{aligned} \tag{3.24}$$

where $p(x|y)$ is the transition probability from state y to state x.

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is a technique to generate samples from the state space by simulating a Markov chain. The formed Markov chain is constructed in such a way that the chain spends more time in the regions with higher importance (i.e., the Markov chain converges to the posterior as its stationary distribution). In other words, the equilibrium distribution of the Markov chain converges to the target distribution. As the number of samples is sufficiently large, all the samples become the fair samples from the posterior. Consequently, we are able to approximate a sophisticated posterior based on deliberately constructing such a Markov Chain of samples.

3.4.2 Sample Correlation

In our design, we choose MCMC instead of other sampling techniques because MCMC maintains the correlation among samples. In MCMC, the next sample depends on the current sample. Before we elaborate on how we can take advantage of sample correlation to improve the efficiency of sampling in our scenario, we define two terms as follows.

Definition We call any sample generated by the sampler a *candidate sample*. A *qualified sample* is a candidate sample that satisfies all constraints.

The existence of constraints leads to the uniqueness of our problem. Samples must satisfy all the constraints to become qualified. Note that in most sampling problems, we prefer independent samples, that is, the current draw of a sample is independent from the previous draw. In our scenario, however, the sampling techniques which generate independent samples (e.g., importance sampling) may suffer from low sampling efficiency due to the loss of correlation between adjacent samples. Figure 3.8 illustrates how the correlation between samples can be utilized to improve the sampling efficiency. The qualified sampling space is a subset of the complete sampling space. Suppose sample point A is the current sample in the qualified sampling space. For an independent sampler, the next sample could be any point in the complete sampling space. However, the next sample is useful only if it happens to fall into the qualified sampling space. On the contrary, if a MCMC-based sampler is employed, the next sample will be chosen according to the proposal distribution at point A, i.e., the next sample will be in the area denoted by the dotted circle centering at point A. Therefore, compared to other independent samplers, the probability that the next sample generated by MCMC falls into the qualified sampling space is considerably increased.

Note that although MCMC improves sampling efficiency, a sample generated by MCMC may not necessarily be a qualified sample. As in Figure 3.8, point B is a sample generated by MCMC after point A. However, point B is outside the qualified sampling space. Consequently, we have to sample again to acquire point C, which is a qualified sample, and then add it into the Markov chain as the next state (i.e., the Markov chain moves from point A to point C).

3.4.3 Metropolis-Hastings and Gibbs Sampling

The Metropolis-Hastings (MH) sampler and the Gibbs sampler are the two most common MCMC samplers. MH conducts a sequence of random walks using a proposal distribution and decides whether to reject the proposed moves using the rejection sampling. In the applications of Bayesian inference, the normalizing constant is usually extremely difficult to

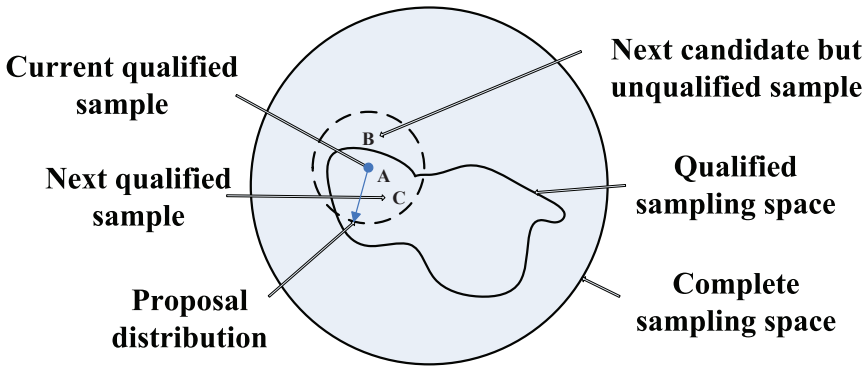


Figure 3.8: Taking advantage of the correlation between samples to improve sampling efficiency.

compute. MH avoids the computation of the constant. It approximates the posterior by using only the ratio of the posterior, where the constant is canceled out.

Recall that the random vector representing the locations of objects is denoted as $\hat{H}$ and the posterior distribution is $post(\hat{H}|\mathbb{Z})$. Suppose $\hat{H}_{t-1}$ is the immediate previous state before the state $\hat{H}_t$ in the formed Markov chain. According to the MH algorithm, at first, a sample, $\hat{H}_q$, is drawn from a proposal distribution, $q(\hat{H}_q|\hat{H}_{t-1})$, i.e., $\hat{H}_q$ is a random deviation from $\hat{H}_{t-1}$. In our research, we use a uniform proposal distribution whose support is defined as the step length. The proposal sample $\hat{H}'$ can be denoted as $\hat{H}_{t-1}+ \hat{H}_q$. Then MH accepts $\hat{H}'$ as the next state $\hat{H}_t$ with the probability of $\frac{post(\hat{H}'|\mathbb{Z})}{post(\hat{H}_{t-1}|\mathbb{Z})}$.

Here we compare MH sampler with the Gibbs sampler in brief. The Gibbs sampler requires that conditional (marginal) distributions for each variable are known and easy to sample from. MH relies on the ratio of the posterior, and does not require to sample from any distribution. Because we have already derived the closed form of the posterior as Equation 3.4 and are able to calculate likelihoods easily according to the proposed n-state model, it will be much more straightforward to use MH sampler rather than the Gibbs sampler in our design.

3.4.4 Metropolis-Hastings sampler with Constraints

Although the naive MH algorithm can evaluate the posterior by forming a Markov chain in the sampling space, it does not take constraints into account. If we impose constraints to samples, many of them should be rejected because they are inapplicable, i.e., they are not qualified samples.

To incorporate constraints in sampling, we propose a Metropolis-Hastings sampler with Constraints (MH-C). With MH-C, each zone is associated with multiple variables called *resource descriptors*. The current value of a *resource descriptor* represents how much the associated resource is still available. Suppose we have a variable, denoted as des_{zone_i}, to keep track of the current available vacancy in zone i. The initial value of des_{zone_i} is set to the maximal capacity of zone i. Thus, whether an object j with the volume vol_{object_j} is able to be stored in zone i can be examined by:

$$des_{zone_i} = des_{zone_i} - vol_{object_j} \tag{3.25}$$

The proposed resource allocation is feasible only if des_{zone_i} is no less than zero. Otherwise, we have to re-sample until a new allocation meets all the constraints. Consequently, the problem of whether an allocation is feasible (or compatible) can reduce to the problem of monitoring the value of each descriptor.

With MH-C, because each sample is a D_{object}-dimensional vector, a proposal sample is generated iteratively dimension by dimension. If any descriptor for the current allocation is less than zero, there will be no chance for the current partial sample to become a qualified sample. Therefore, we can discard the current value and then choose another value for that dimension by re-sampling. As far as the proposal distribution is concerned, we construct a random walk chain by choosing a uniform proposal distribution within the step length. A detailed description of MH-C algorithm is illustrated in Algorithm 3 and the related notations are summarized in Table 3.3.

Symbol	Meaning
$\mathbb{Z}$	The raw data matrix from RFID readers
$\mathbb{S}$	The sample set
$\vec{C}$	The current sample in the Markov chain
$\vec{P}$	The proposal sample in the Markov chain
C_j	The j^{th} dimension of $\vec{C}$
P_j	The j^{th} dimension of $\vec{P}$
E	The number of effective samples
B	The number of samples in the burn-in phase
S	The step length for the uniform proposal distribution
D_{object}	The total number of monitored objects
D_{zone}	The total number of zones
$Jitter$	A random number between 0 and 1
$Rand(a, b)$	Generate a random integer between a and b based on uniform distribution
$Post(\hat{H}\|\mathbb{Z})$	The posterior probability of the sample $\hat{H}$ given raw data $\mathbb{Z}$

Table 3.3: Symbolic notations utilized in Algorithm 3.

In Algorithm 3, line 1 initializes the sample set and takes the RFID raw data. Line 2 loads the n-state detection model of readers with the objective of computing the likelihood. Line 3 initializes all the resource descriptors and line 4 randomly chooses a qualified sample as the first state of the Markov Chain. Lines 6 to 17 generate a random sample as the proposal sample dimension by dimension (object by object). Lines 8 to 13 correspond to the sampling process. First, we obtain a random integer based on the current value and the random proposal value. The correct range of the value on each dimension of the sample vector, represented as h_i, is [1, D_{zone}]. Therefore, if the proposal value P_j overflows in the range, we need to make the value reflect into the range. Then, we check all the related

descriptors to make sure their updated values are no less than zero. If any value is less than zero, it means that the current allocation will violate the corresponding constraints. Consequently, we go back to line 8 to re-sample until an allocation on that dimension is feasible, as shown in line 15. Note that our algorithm guarantees each proposal sample is also a qualified sample. After a complete proposal sample is generated, lines 18 to 21 accept this proposal sample as the next state of the Markov chain with the probability of the posterior ratio of the proposal sample over the current sample. Line 22 adds the next state into the sample set. Line 23 resets all the resource descriptors to make sure that the examination on descriptors for the next proposal sample is correct. Note that line 5 is to set the upper bound of the sampling loop to $E + B$ in order to guarantee that the final number of samples in sample set is $E + B$. It is because the first B samples should be excluded as burn-in samples and consequently only the remaining E samples will be taken into account to reconstruct the posterior.

3.5 Real-Time Cleansing on Data Streams

$$\begin{aligned} post(\hat{H}|\mathbb{C}) &= post(h_1, h_2, \ldots, h_n| \begin{bmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{bmatrix}) \\ &\propto p(\begin{bmatrix} c_{11} & \cdots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nm} \end{bmatrix} |h_1, h_2, \ldots, h_n) \cdot p(h_1, h_2, \ldots, h_n) \end{aligned} \tag{3.26}$$

$$post(\hat{H}|\mathbb{C}) = \alpha \cdot \prod_i p(c_{i1}|h_i) \cdot p(c_{i2}|h_i) \cdot \ldots \cdot p(c_{im}|h_i) \cdot \prod_j p(h_j) \tag{3.27}$$

$$p(c_{ij}|h_i) = \frac{N!}{c_{ij}! \times (N - c_{ij})!} \times p(z_{ij}|h_i)^{c_{ij}} \times (1 - p(z_{ij}|h_i))^{N - c_{ij}} \tag{3.28}$$

In many real-world applications, RFID raw data is incrementally becoming available as data streams and the locations of objects may change over time [14, 61, 62]. In support of

real-time object monitoring where RFID data are generated quickly and automatically, we propose a Streaming Bayesian Inference (SBI) approach. SBI combines Bayesian inference with MH-C and employs a *counted matrix* in the *sliding window* to cleanse the RFID data and predict the locations of tagged objects in a real-time fashion.

3.5.1 Streaming Bayesian Inference

First of all, we assume that there exist a *sliding window* during which the locations of all objects can be considered as unchanged and the process of posterior reconstruction can be finished during each *sliding window*. In the previous section, we illustrated that MH-C is a very efficient way to draw samples from the posteriors. As demonstrated in Figure 3.10, in most cases, MH-C can generate the reconstruction of posteriors for all objects in no longer than 20 seconds. Therefore, it is reasonable to set the size of the *sliding window* according to the running time of MH-C. On the other hand, our *sliding window* based approach can meet the common requirement in data stream processing algorithms which dictates that the time complexity must be linear.

Once the size of the *sliding window* is determined, the posterior distributions of locations of all objects within a certain *sliding window* can be computed based on Bayesian inference, as shown in Equation 3.26. While the non-streaming Bayesian inference approach directly utilizes the RFID raw data matrix as the input (as illustrated in Equation 3.1), the streaming Bayesian inference method takes the *RFID counted matrix* as the input as shown in Equation 3.26. In Equation 3.26, $\mathbb{C}$ is the *counted matrix* which is generated by counting the numbers of successfully readings (i.e., "1's") for each combination of objects and readers over all the raw data matrices during a *sliding window*, i.e., $c_{ij} = \sum_t {z_{ij}}^t$, where ${z_{ij}}^t$ denotes the reported value of z_{ij} at the time point t during the sliding window. Besides, if we assume that each reader detects different objects independently and the prior distribution of each object does not depend on that of others, we can obtain Equation 3.27, where α is a normalizing constant. In other words, given a *counted matrix* with respect to a certain

sliding window, we can compute the posterior of each sample, $\hat{H}$, within this *sliding window*, $Post(\hat{H}|\mathbb{C})$ by using Equation 3.27. Furthermore, noticing that the read operation of each reader at different time points is independent from each others', we can conclude that c_{ij} in fact follows a binomial distribution, i.e., c_{ij} *w.r.t.* $h_i \sim B(N, p(z_{ij}|h_i)))$, where N is the total number of read operations within the *sliding window*. Therefore, $p(c_{ij}|h_i)$ can be computed according to the binomial distribution as shown in Equation 3.28. Notice that $p(z_{ij}|h_i)$ in Equation 3.28 can be estimated based on our n-state detection model efficiently.

3.5.2 Algorithm Description

The complete algorithm of our SBI method is formalized in Algorithm 4. In Algorithm 4, Line 1 generates the counted matrix, $\mathbb{C}$, by counting the numbers of successful readings, i.e., $c_{ij} = \sum_t {z_{ij}}^t$. Subsequently, all samples are generated using MH-C. To be specific, line 2 randomly chooses a qualified sample within the support of $Post(\hat{H}|\mathbb{C})$ as the first state of the Markov Chain. Lines 3 to 4 correspond to the sampling process. Line 4 draws a proposal sample, $\hat{H}'_i$, according to MH-C. In line 5, $Post(\hat{H}'_i|\mathbb{C})$ is computed using Equation 3.27 and Equation 3.28 according to the binomial distribution $B(N, p(z_{ij}|h_i))$, and the n-state detection model. Line 6 calculates the posterior ratio of the proposal sample over the current state, and then accepts $\hat{H}'_i$ as the next state with the acceptance ratio of $\frac{Post(\hat{H}'_i|\mathbb{C})}{Post(\hat{H}_{i-1}|\mathbb{C})}$.

3.6 Experimental Validation

In this section, we applied our method to a warehouse application, i.e., each object corresponds to a case and each zone corresponds to a rack. To capture the likelihood, without loss of generality, we assumed that the 3-state detection model was adopted in this application. We implemented MCMC (we use the terms MH-C and MCMC interchangeably in this section) and our Streaming Bayesian Inference (SBI) approach to evaluate their performances. For comparison with our MCMC-based solution, we extended the SIS-based approach in [67] to incorporate duplicate readings (i.e., their work does not consider duplicate

readings and only focuses on the distribution of the missing cases). For the experiments on SBI(N), where N denotes the number of arrivals of raw data at different time points during each *sliding window*, we varied N from 1 (SBI (1)), 3 (SBI (3)) to 5 (SBI (5)). In particular, when N equals to 1, SBI(N) degrades to our MCMC-based solution (the non-steaming method). Therefore, we employed SBI (1) as a baseline to show the advantage of our SBI-based approaches.

3.6.1 Data Representation

In this dissertation, we store the true distributions of objects and RFID raw readings at each time point in matrices. For simplicity of presentation, suppose we have six tagged objects in the target area. A possible true distributions matrix is shown in Table 3.4(a) where the rows represent cases and columns represent racks. To be specific, if a case is on a rack, the corresponding position is 1 and 0 otherwise. Therefore, according to Table 3.4(a), the first case is on the first rack, the second case is on the fourth rack and so on. On the other hand, a possible RFID raw reading matrix is shown in Table 3.4(b) in the same format. By comparing the two matrices, we can easily see that there are notable differences (existence of noise and duplicate readings) between these two matrices. Specifically, duplicate readings of the fourth and fifth case (consecutive 1's) occur in the raw reading matrix (data).

$$\underset{\text{(a)}}{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}} \quad \underset{\text{(b)}}{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}}$$

Table 3.4: The matrix of: (a) true distribution and (b) raw readings.

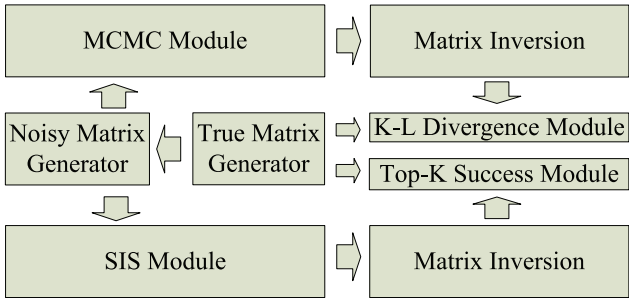


Figure 3.9: The simulator structure.

3.6.2 Simulator Implementation

Our simulator consists of seven components as displayed in Figure 3.9. The true matrix generator randomly produces distribution matrices as true distributions. On the contrary, noisy matrix generator provides the noisy matrices as the RFID raw data in the same format. Then MCMC and SIS modules reconstruct the distribution for each case using the noisy matrix as the input. Our simulator generates the synthetic RFID raw data with duplicate readings according to the physical characteristics of RFID readers [31]. The 3-state detection model was used to capture the likelihood. In addition, we assume as the prior distribution that each case exists on each rack with the same probability. The parameters used in our simulations are shown in Table 3.5. All our experiments were conducted on a Linux machine with an Intel Pentium 4 2.4GHz processor and 2GB of memory.

In order to investigate the impact of redundant readings on reconstruction accuracy, we define the *data redundancy degree* as follows:

Definition The *data redundancy degree* is the probability that a reader successfully detects an object not in the zone associated to that particular reader.

The *data redundancy degree* can reflect the probability that redundant readings occur in RFID raw data, i.e., redundant readings (false positives) are actually the successful detection of objects which are not in the zone that a certain reader belongs to. Taking the 3-state model as an example, we can utilize the read rate in the minor detection region to define

the data redundancy degree. A larger data redundancy degree indicates a higher probability that a reader can detect objects in the neighboring zones (or racks).

We employed K-L divergence, the top-1 success rate, and the top-2 success rate to evaluate the reconstruction accuracy. K-L divergence is a metric commonly used to evaluate the difference between two distributions. The definition of the top-k success rate is as follows:

Definition The *top-k success rate* is a percentage of the number of cases whose true locations match the top k predicted locations of the reconstructed distribution over the total number of cases.

Our simulator calculated the K-L divergence from the reconstructed distribution to the true distribution, i.e., the smaller value of K-L convergence indicates the higher accuracy of the reconstructed distribution. Specifically, the recovered matrices (reconstructed distributions) were inverted by the matrix inversion module to facilitate the computation of K-L divergence. The K-L divergence module was used to compare the reconstructed distributions with the true distributions and compute the corresponding K-L divergence values for MCMC and SIS. On the other hand, the top-k analysis module was responsible for calculating the top-k success rate.

For each experimental result in this section, we randomly generated the true distribution matrix and the corresponding RFID raw reading matrix 100 times. During each run of distribution reconstruction, we recorded the sampling time, and computed the average K-L divergence, the top-1 success rate and the top-2 success rate of all the 5000 cases (i.e., each result is obtained by averaging over 5000 tagged objects in the whole target area).

3.6.3 The Performance Analysis of MH-C

In this section we focus on the performance of MCMC and SIS with respect to reconstruction efficiency and accuracy.

Parameter	Value
D_{object}	5000
D_{zone}	200
B	50
S	30
Vol_{object}	1
$Capacity_{zone}$	50

Table 3.5: The parameters for our simulations.

The Reconstruction Efficiency

Here we investigated the performance of MCMC and SIS in terms of the average sampling time. Compared to SIS, the average sampling time of MCMC is remarkably reduced over different number of qualified samples as illustrated in Figure 3.10. For example, with 5000 qualified samples the sampling time of MCMC is 11.58 seconds while the sampling time of SIS is 230.78 seconds. This is because MCMC takes advantage of the current qualified sample to generate the next qualified sample (i.e., keeping the relevance of samples). Consequently, MCMC takes less time than SIS to come up with the same number of qualified samples.

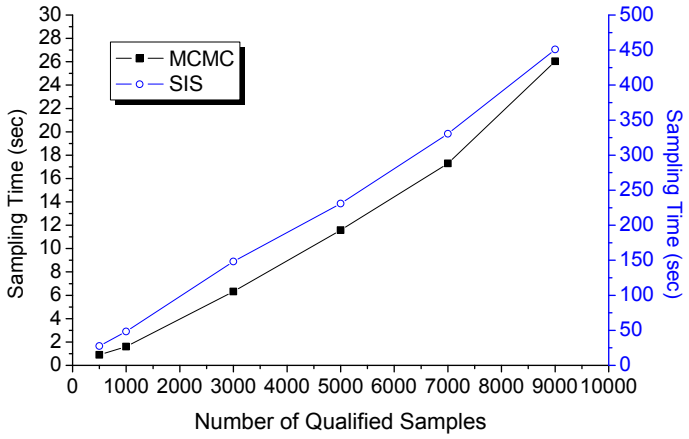
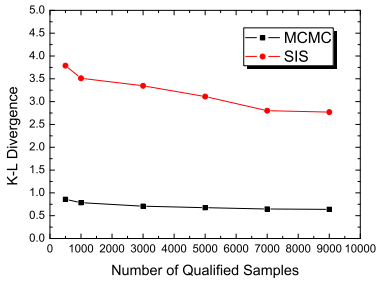


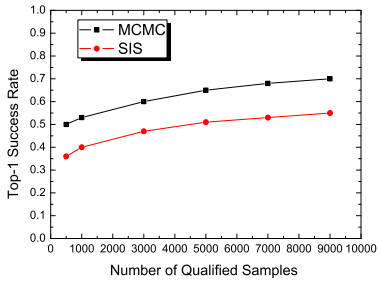
Figure 3.10: MCMC versus SIS on sampling time.

The Reconstruction Accuracy

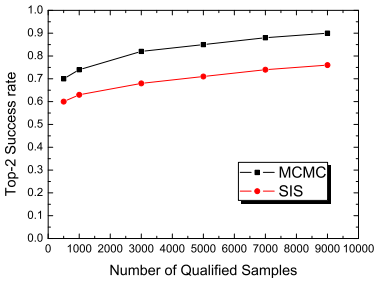
In this experiment, we varied the number of qualified samples, data redundancy degree, and the number of managed racks per reader to investigate their effects on the reconstruction accuracy.



(a) Average K-L divergence of 5000 cases.

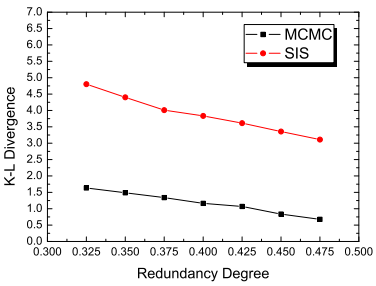


(b) Top-1 success rate of 5000 cases.

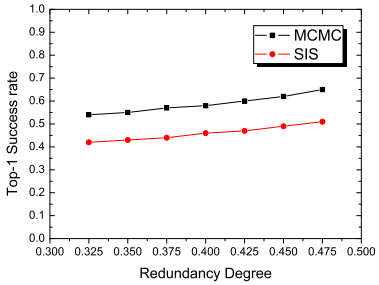


(c) Top-2 success rate of 5000 cases.

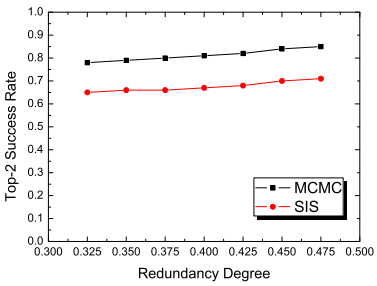
Figure 3.11: The impact of the number of qualified samples on accuracy for MCMC and SIS.



(a) Average K-L divergence of 5000 cases.



(b) Top-1 success rate of 5000 cases.

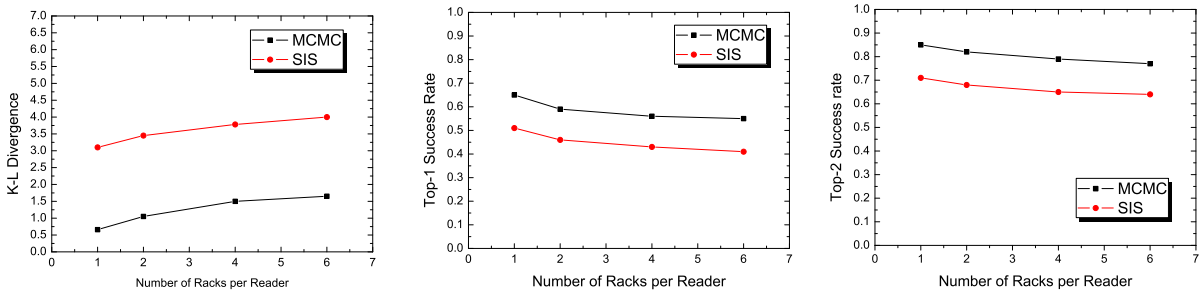


(c) Top-2 success rate of 5000 cases.

Figure 3.12: The impact of RFID data redundancy degree on accuracy for MCMC and SIS.

The Impact of the Number of Qualified Samples

We first increased the number of qualified samples from 500 to 9000 to investigate the performance of MCMC and SIS on reconstruction accuracy. Here we assumed that the read rate in the major detection region is 95% and the number of racks managed by a reader is 1. As demonstrated in Figure 3.11(a), with the increase of the number of qualified samples, the K-L divergence values of both approaches kept decreasing. However, MCMC always



(a) Average K-L divergence of 5000 cases.

(b) Top-1 success rate of 5000 cases.

(c) Top-2 success rate of 5000 cases.

Figure 3.13: The impact of the number of managed racks per reader on accuracy for MCMC and SIS.

outperformed SIS with all experimented sample numbers. Particularly, when we drew 500 qualified samples, the K-L divergence of MCMC was 0.86 while the K-L divergence of SIS was 3.78. When we picked 9000 qualified samples, the K-L divergence of MCMC reduced to a remarkable value 0.64 comparing with 2.77 of the SIS solution. Also, as far as the top-1 success rate is concerned, with the increase of the number of qualified samples, the top-1 success rate of MCMC increased from 0.50 to 0.70 while the top-1 success rate of SIS extended from 0.36 to 0.55 as shown in Figure 3.11(b). In addition, the top-2 success rate of MCMC raised from 0.70 to 0.89 while the top-2 success rate of SIS changed from 0.60 to 0.76 as demonstrated in Figure 3.11(c).

The Impact of the Data Redundancy Degree

Next, we studied the impact of the *data redundancy degree* on the performance of MCMC and SIS in terms of the reconstruction accuracy. We varied the data redundancy degree from 0.325 to 0.475, corresponding to the read rate in the major detection regions from 65% (the least reliable reader) to 95% (the most reliable reader). For each experiment, we drew 5000 qualified samples and assumed that the number of racks managed by a reader is 1. Figure 3.12 illustrates the results. With the enlargement of the data redundancy degree, both the performances of MCMC and SIS on reconstruction accuracy are elevated. Specifically, as demonstrated in Figure 3.12(a), MCMC always maintained a lower K-L

divergence value than SIS, reflecting a more precise prediction. Furthermore, as shown in Figure 3.12(b), the top-1 success rate of MCMC increased from 0.54 to 0.65 with the increase of the data redundancy degree while the top-1 success rate of SIS expanded from 0.42 to 0.51. Figure 3.12(c) demonstrates how the top-2 success rate increased when we raised the data redundancy degree for both MCMC and SIS.

The Impact of the Number of Managed Racks per Reader

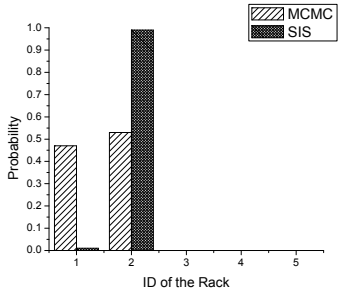
We evaluated the performance of MCMC and SIS by varying the number of managed racks per reader. In order to deploy readers in a warehouse more efficiently, users may want to assign multiple racks to be managed by a single reader. Taking into account the fact that the overall detection region of a regular RFID reader has little chance to be more than 20 feet [31], we changed the number of racks managed by a reader from 1 to 6. As Figure 3.13(a) demonstrates, when each rack had its own reader, the K-L divergence values of MCMC and SIS were 0.68 and 3.11, respectively. When a reader monitored more racks, both the K-L divergence values of MCMC and SIS deteriorated. When a reader was responsible for detecting cases on six racks, the K-L divergence values raised to 1.66 of MCMC and 4.01 of SIS. Moreover, as demonstrated in Figure 3.13(b), with the enlargement of the number of managed racks per reader the top-1 success rate of MCMC decreased from 0.65 to 0.55. On the other hand, the top-1 success rate of SIS dropped from 0.51 to 0.41. Also, Figure 3.13(c) depicts how the top-2 success rate of MCMC and SIS decreased correspondingly.

3.6.4 Visualization of Query Results

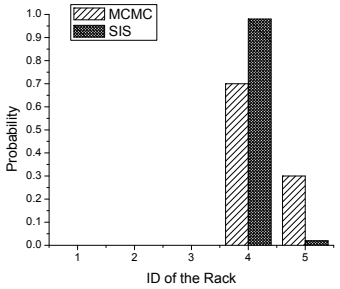
The Location Query

The results of the location queries returned by MCMC and SIS for the first six cases are demonstrated in Figure 3.14. For the first case, the correct location is the first rack. MCMC predicted a probability of 0.47 on that rack while SIS predicted a probability of only 0.01. Similarly, for the second case, the true location is the fourth rack. MCMC predicted

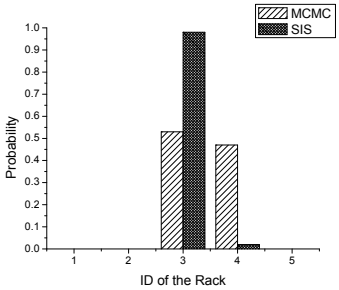
a probability of 0.70 on that rack while SIS predicted a probability of 0.98. For the third case, the accurate location is the third rack. MCMC predicted a probability of 0.53 on that rack while SIS predicted a probability of 0.98. For the fourth case, the exact location is the second rack. MCMC predicted a probability of 0.51 on that rack while SIS predicted a probability of 0.32. For the fifth case, the precise location is the fifth rack. MCMC predicted a probability of 0.53 on that rack while SIS predicted a probability of 0.99. At last, for the sixth case, the correct location is the first rack. MCMC predicted a probability of 0.20 on that rack while SIS predicted a probability of only 0.05. In summary, MCMC tends to generate a much smoother probability distribution than SIS. Consequently, MCMC provides a superior overall prediction of the distribution of all the objects monitored by an RFID system compared with SIS.



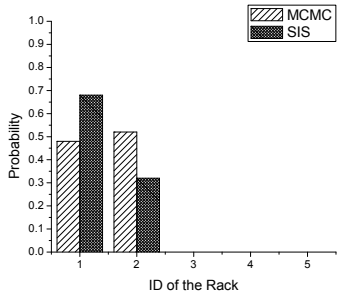
(a) First case (true location: rack 1).



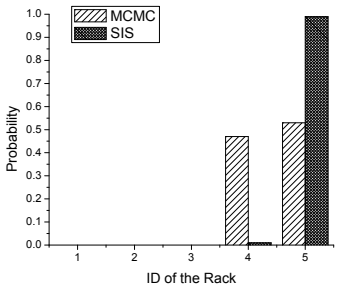
(b) Second case (true location: rack 4).



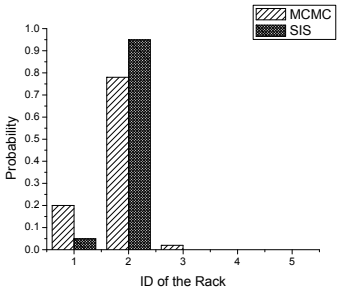
(c) Third case (true location: rack 3).



(d) Fourth case (true location: rack 2).



(e) Fifth case (true location: rack 5).

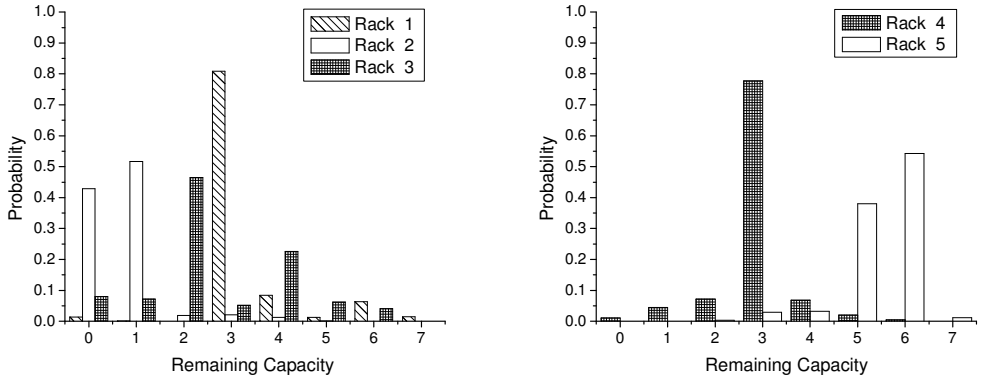


(f) Sixth case (true location: rack 1).

Figure 3.14: The results of the location queries answered by MCMC and SIS.

The Remaining Capacity Query

We applied the available length on a rack as the acquirable volume of that rack to demonstrate the remaining capacity query. The results of the remaining capacity queries answered by MCMC for the first five racks are demonstrated in Figure 3.15. Note that the remaining capacity on each rack is 3 because each rack exactly accommodates 4 cases according to the true distribution matrix used for this experiment (the first six rows of the matrix are as shown in Table 3.4(a)). As illustrated in Figure 3.15, the remaining volumes on racks 1 and 4 were reported correctly. On the contrary, the remaining volumes on racks 2 and 3 were underestimated and the remaining volume on rack 5 was overestimated.



(a) Racks 1, 2 and 3 (true value: 3).

(b) Racks 4 and 5 (true value: 3).

Figure 3.15: The results of the remaining capacity queries answered by MCMC.

3.6.5 The Performance Analysis of SBI

In this section we evaluated the performance of SBI (1), SBI (3) and SBI (5) with respect to reconstruction efficiency and accuracy.

The Reconstruction Efficiency

Here we investigated the performance of SBI (1), SBI (3) and SBI (5) in terms of the average sampling time. When N increases, the average sampling time of SBI (N) extends almost linearly with the same number of qualified samples, as illustrated in Figure 3.16. For

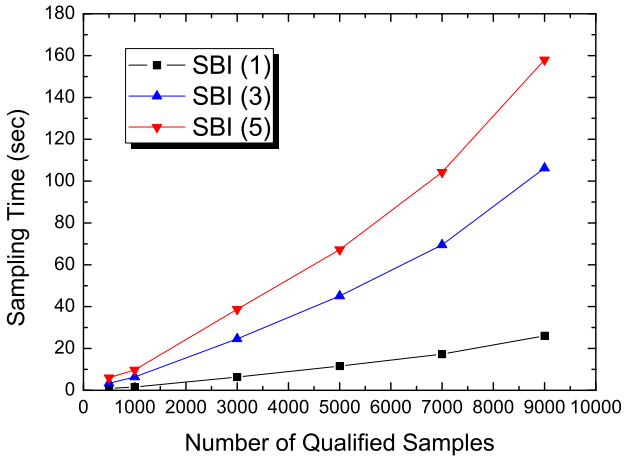


Figure 3.16: Performance comparison of SBI (1), SBI (3) and SBI (5) on sampling time.

example, with 5000 qualified samples the average sampling time of SBI(1) is 11.58 seconds while the average sampling time of SBI (3) and SBI (5) is 45.10 seconds and 67.25 seconds, respectively. This is because a larger N requires more calculations to estimate the likelihood of each sample based on the Binomial distribution, $B(N, p(z_{ij}|h_i))$, thus leading to more time spent on computing the posterior of each sample.

The Reconstruction Accuracy

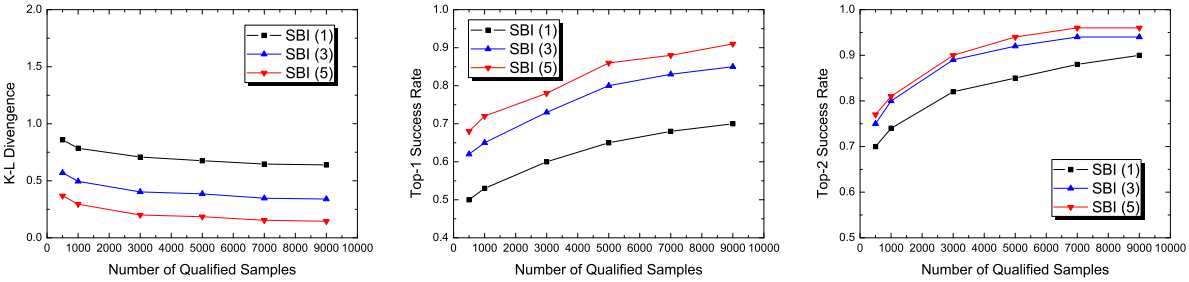
In this experiment, we focus on the performance of SBI (1), SBI (3) and SBI (5) on reconstruction accuracy.

The Impact of the Number of Qualified Samples

We first studied the relationship between the number of qualified samples and the reconstruction accuracy under SBI (1), SBI (3) and SBI (5). Here we assumed that the read rate in the major detection region is 95% and the number of racks managed by each reader is 1. As shown in Figure 3.17(a), with a larger N, the K-L divergence of SBI (N) kept decreasing. Particularly, when we drew 5000 qualified samples, the K-L divergence of SBI (1) is 0.68. On the contrary, the corresponding K-L divergence of SBI (3) and SBI (5) is 0.39 and 0.19, respectively. Furthermore, when we chose a larger N, the top-1 success rate

of SBI (N) will be elevated accordingly as illustrated in Figure 3.17(b). For instance, with 5000 qualified samples the top-1 success rates of SBI (3) and SBI (5) are 0.80 and 0.86, respectively while the top-1 success rate of SBI (1) is 0.65. In addition, as Figure 3.17(c) demonstrates, the top-2 success rate can also be improved by taking a larger N. For example, with 5000 qualified samples the top-2 success rates of SBI (1), SBI (3) and SBI (5) are 0.85, 0.92 and 0.94, respectively. In summary, SBI (3) and SBI (5) always outperformed SBI (1) in terms of accuracy because they can estimate the posterior of each sample more precisely by taking into consideration a larger number of RFID raw readings of each object at a particular location. Therefore, a more precise localization can be achieved by inferring jointly on these readings generated at different time points.

The Impact of the Data Redundancy Degree



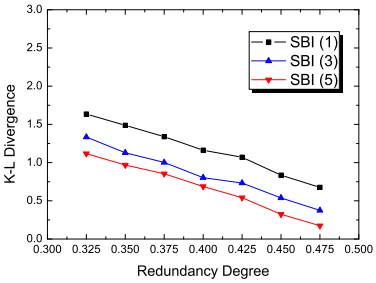
(a) Average K-L divergence of 5000 cases.

(b) Top-1 success rate of 5000 cases.

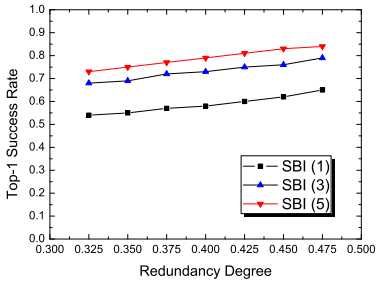
(c) Top-2 success rate of 5000 cases.

Figure 3.17: The impact of the number of qualified samples on accuracy for SBI (1), SBI (3) and SBI (5).

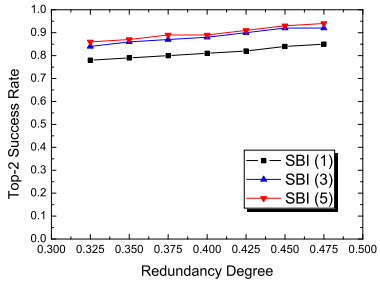
Next we investigated the performance of SBI (1), SBI (3) and SBI (5) on the reconstruction accuracy by varying the *data redundancy degree*. For each experiment, we picked 5000 qualified samples and assumed that the number of racks managed by a reader is 1. As demonstrated in Figure 3.18(a), with the enlargement of N the K-L divergence of SBI (N) kept decreasing, reflecting a more precise prediction. In particular, when the data redundancy degree is 0.475, the K-L divergence of SBI (3) and SBI (5) is 0.38 and 0.17, respectively while the K-L divergence of SBI (1) is 0.68. Moreover, as shown in Figure 3.18(b), when N



(a) Average K-L divergence of 5000 cases.



(b) Top-1 success rate of 5000 cases.



(c) Top-2 success rate of 5000 cases.

Figure 3.18: The impact of RFID data redundancy degree on accuracy for SBI (1), SBI (3) and SBI (5).

increased from 1, 3, to 5, the top-1 success rate of SBI (1), SBI (3) and SBI (5) is elevated accordingly. For example, with the data redundancy degree of 0.475, the top-1 success rate of SBI (1) is 0.65. On the other hand, the corresponding top-1 success rates of SBI (3) and SBI (5) jumped to 0.79 and 0.84, respectively. Besides, Figure 3.18(c) demonstrates how the top-2 success rate can be raised by choosing a larger N.

Algorithm 3 Metropolis-Hastings Sampler with Constraints

Set $\mathbb{S} = \emptyset$ and take raw data $\mathbb{Z}$
Load the n-state detection model
Initialize all the resource descriptors to their maximal capacity.
Initialize $\vec{C}$ by randomly choosing a qualified sample within the support of $Post(\hat{H}|\mathbb{Z})$ as the starting point.
for $Cycle = 2$ to E+B **do**
  for $j = 1$ to D_{object} **do**
    repeat
      $P_j = C_j$+ Rand(-S,S)
      {Generate a new integer based on the current value and a proposal value within the step length}
      if $P_j < 1$ **then**
        $P_j = 1$+ (1-P_j)
        {Overflow and Reflection}
      end if
      if $P_j > D_{zone}$ **then**
        $P_j = D_{zone}$-(P_j-D_{zone})
        {Overflow and Reflection}
      end if
    until The value of any resource descriptor related to the referred zone is no less than zero after the proposed allocation on the current object is committed
    $j \leftarrow j + 1$
  end for
  Generate a random number between 0 and 1: $Jitter$
  if $Jitter \leq \min(1, \frac{Post(\vec{P}|\mathbb{Z})}{Post(\vec{C}|\mathbb{Z})})$ **then**
    $\vec{C}$=$\vec{P}$
    {Metropolis-Hastings}
  end if
  Add $\vec{C}$ into $\mathbb{S}$ as the next sample
  Resetting all the resource descriptors
  $Cycle \leftarrow Cycle + 1$
end for

Algorithm 4 Streaming Bayesian Inference (SBI) Method

Require: Raw data matrices $\mathbb{Z}^{t_1}$, $\mathbb{Z}^{t_2}$, ..., $\mathbb{Z}^{t_N}$ at time points, t_1, t_2, ..., t_N, within a *sliding window* W and the n-state detection model

1: Compute counted matrix, $\mathbb{C}$, by counting the numbers of successfully readings in $\mathbb{Z}^{t_1}$, $\mathbb{Z}^{t_2}$, ..., $\mathbb{Z}^{t_N}$ within W

2: Randomly choose a qualified sample, $\hat{H}_0$, within the support of $Post(\hat{H}|\mathbb{C})$ as the the first state of the Markov Chain.

3: **for** each cycle i **do**

4: Draw a proposal sample $\hat{H}'_i$

5: Compute $Post(\hat{H}'_i|\mathbb{C})$ according to Equation 3.27 and Equation 3.28

6: Accept $\hat{H}'_i$ as the next state in the Markov chain, $\hat{H}_i$, with the probability of $\frac{Post(\hat{H}'_i|\mathbb{C})}{Post(\hat{H}_{i-1}|\mathbb{C})}$ {Metropolis-Hastings Sampler with Constraints}

7: **end for**

Ensure: All the accepted states of the resulting Markov chain, $\hat{H}_0$, $\hat{H}_1$, $\hat{H}_2$, ... , $\hat{H}_i$, ... , $\hat{H}_n$ {By counting over samples, we reconstruct posteriors of locations of all objects w.r.t. the sliding window w}

Chapter 4

Related Work

4.1 Spatial Databases

4.1.1 Nearest Neighbor Query

The nearest neighbor query is a very important query type for supporting GIS applications. With the R-tree family [7, 24, 51] of spatial indices, depth first search (DFS) [47] and best first search (BFS) [25] have been the prevalent branch-and-bound techniques for processing nearest neighbor queries. The DFS method recursively expands the intermediate nodes for searching NN candidates. At each newly visited index node, DFS computes the ordering metrics for all its child nodes and applies pruning strategies to remove non-promising branches. When the search reaches a leaf node, the data objects are retrieved and the NN candidates are updated. On the other hand, the BFS method employs a priority queue to store nodes to be explored through the search process. The nodes in the queue are sorted according to their minimum distance (MINDIST) to the query point. During the search process, BFS repeatedly dequeues the top entry in the queue and enqueues its child nodes with their MINDIST into the queue. When a data entry is dequeued, it is included in the result set.

Recently nearest neighbor search solutions have been extended to support queries on spatial networks. Jensen et al. [32] proposed data models and graph representations for NN queries in road networks and designed corresponding solutions. Papadias et al. [43] presented solutions for NN queries in spatial network databases by progressively expanding road segments around a query point. A network Voronoi diagram based solution for NN search in road network databases was proposed in [37]. Sharifzadeh et al. [54] extended

the Voronoi diagram based approach for spatial data streams by using approximate Voronoi cell computation. On the contrary, in order to speed up the NN search, Samet et al. [50] proposed a solution to explore the entire spatial network by pre-computing the shortest paths between all the vertices in the network and using a shortest path quadtree to capture spatial coherence. Using their approach, the shortest paths between various vertices can be only computed once to answer different NN queries on a given spatial network. However, the above pre-computation based approaches suffer from high overhead and adapt poorly to network updates. To overcome this pitfall, Lee et al. [39] presented an efficient and flexible query framework, *ROAD*, based on *search space pruning* by using *shortcuts* for accelerating network traversals and *object abstracts* for guiding traversals.

4.1.2 Route Planning Query

In many GIS applications (e.g., logistics and supply chain management), users have to plan a trip to a number of locations with several sequence rules and the goal is to find the optimal route that minimizes the total traveling distance. One related query type is named the optimal sequenced route (OSR) query proposed by Sharifzadeh et al. [52]. OSR query retrieves a route of minimum length starting from a given source location and passing through a number of locations (with different types) in a particular order (sequence) imposed on all the POI types. In [53], a pre-computation approach was provided to answer OSR query by taking advantage of a family of AW-Voronoi diagrams for different POI types. However, because the searches in [53] are based on NN queries, the authors need to investigate the performance of their method in terms of the traveling distance besides the response time. A multi-type nearest neighbor (MTNN) query solution was proposed in [41] by Ma et al. Given a query point and a collection of locations (with difference types), a MTNN query finds the shortest path for the query point such that only one instance of each type is visited during the trip. MTNN can be treated as an extended solution of OSR by exploiting a page-level upper bound. On the contrary, Li et al. [40] designed solutions for another new query type –

Trip Planning Queries (TPQ). With TPQ, the user specifies a set of POI types and asks for the optimal route from her starting location to a specified destination which passes through exactly one POI in each POI type. Notice that compared to a OSR query, there is no order imposed on the types of POIs to be visited in a TPQ query. Terrovitis et al. [58] illustrated a-autonomy shortest path and k-stops shortest path problems for spatial databases. Given a source point and a destination point, the first query retrieves a sequence of points from the database where the distance between any two consecutive points in the path is not greater than a. The second query searches for the optimal path from a origin to an end which passes through exactly k intermediate points in the database. Tian et al. [59] proposed skyline path queries in road networks based on multiple route search criteria (i.e., shortest traveling distance and shortest traveling time). By taking into account the probability that each POI type satisfies the user's particular need, an interactive approach was proposed in [34]. In order to capture the characteristics of ever-changeling road networks, Tian et al. [60] proposed the continuous min-cost path query and presented a system, *PathMon*, to monitor min-cost routes in dynamic road networks. However, all the aforementioned solutions cannot support MRPSR queries.

Another related problem to MRPSR is the sequential ordering problem (SOP) [19] and it is stated as follows. Given a graph G with n vertices and directed weighted edges, find a minimal cost *Hamiltonian path* from the start vertex to the terminal vertex which also observes precedence constraints. Nevertheless, a Hamilton path is not required in MRPSR and the types of visited locations are considered by our solution.

4.2 RFID Databases

Many systems have been developed to manage data with uncertainty. The usual approach to address uncertainty is to augment the classical relational model with attribute-level or tuple-level probability values [3, 5, 6, 13, 15]. On the other hand, a more general approach based on sampling is proposed for managing incomplete and uncertain data [11, 28, 67]. The

idea is simple and intuitive: we construct random samples while observing the prior statistical knowledge and constraints about the data. Thus, each sample is one possible realization in the space of uncertainty, and the entire set of samples reveals the distribution of the uncertain data we want to model. Queries and inferences are then conducted against this distribution.

An important application that drives the recent surge of interest in managing incomplete and uncertain data is RFID data management. Chawathe et al. [9] proposed a system architecture of a distributed RFID system and discussed related data management challenges such as inferences and online warehousing. An expressive temporal data model for RFID data is defined by Wang and Liu [65] to support tracking and monitoring queries. Gonzalez et al. [23] designed a warehousing model which provides RFID data compression and path-dependent aggregates. Based on the proposed warehousing framework, they also developed techniques for summarizing and indexing RFID data and various query methods. Because RFID raw data usually contains anomalies [20], solutions have been proposed to clean the input data from readers. Jeffery et al. [29] presented a framework for building data cleaning infrastructure to support pervasive applications. An adaptive smoothing filter (SMURF) for RFID data cleaning was proposed in [31]. SMURF focuses on a sliding-window aggregate that interpolates for lost readings. Cocci et al. [14] provided a data interpretation and compression scheme over RFID streams. A probabilistic model [61] was provided to capture the dynamics of the reader and object, and noisy readings in RFID stream processing. Tran et al. [63] presented PODS, a system to support stream processing for captured uncertain data using continuous random variables based on Gaussian Mixture Models (GMM). An approximate framework to evaluate complex queries involving conditioning and aggregation operations on uncertain data streams was proposed in [62]. Our work, however, focuses on how to leverage spatio-temporal data redundancy to elevate the localization accuracy for all the tagged objects in a target area for real-time RFID tracking and monitoring applications.

The works in [35,36,67] are the most relevant research to this dissertation. For correcting erroneous RFID raw data, Khoussainova et al. [35,36] presented a system for correcting input data errors automatically using application defined global integrity constraints. The system corrects the input data by inserting missing tuples when necessary and assigning to each one the probability that it is correct for groups of conflicting tuples. This maximum entropy based solution is practical. However, it is unable to capture all application related prior knowledge and dependency compared with sampling-based approaches. Useful information can be recovered from noisy RFID data by exploiting constraints and sequential importance sampling methods [67]. Nevertheless, the work in [67] failed to consider the duplicate readings caused by the overlapped detection regions of RFID readers.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Geographic information systems are getting increasingly sophisticated and route queries with traveling rules represent a significant class of spatial queries. Existing solutions only focus on trips with a complete POI category sequence or without any sequence. However, GIS users usually want to set a number of traveling preferences when they plan their trips. In this dissertation we propose the MRPSR query and design three fast approximation algorithms to efficiently compute routes which can fulfill all the traveling rules with a near-optimal travel distance based on the underlying road networks. With extensive simulations, we show that our techniques can generate satisfying trips which are very close to the shortest routes with remarkable short response time.

The data reported by RFID devices are known to be unreliable. In this dissertation we propose a Bayesian inference based framework for cleansing RFID raw data which can take full advantage of the spatio-temporal redundancy in the raw data. Our approach takes into account prior knowledge (prior distribution and the likelihood) to evaluate location queries for all tagged objects in a target area. Specifically, we propose the n-state model to capture likelihood and validate that the 3-state model can maximize the system performance. Moreover, we devise MH-C to efficiently sample from the posterior distribution under environmental constraints. To deal with the RFID raw data available as data streams, we present our streaming based solution, the streaming Bayesian inference method, in support of real-time tracking and monitoring. Finally, we demonstrate the advantages of our approach in terms of efficiency and accuracy by extensive simulations.

5.2 Future Work

For future work, we intend to extend our proposed algorithms in support of MRPSR queries on dynamic road networks in which traffic information (e.g., travel time, traffic congestion, etc.) is incrementally becoming available as a data stream. As for our study on RFID databases, we plan to extend our n-state model to cover 2D RFID reader array deployment and identify the obstacles when integrating a 2D n-state model with our Bayesian inference based framework.

Bibliography

[1] Digital Chart of the World Server.

[2] U.S. Geological Survey.

[3] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A System for Data, Uncertainty, and Lineage. In *VLDB*, pages 1151–1154, 2006.

[4] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2):5–43, 2003.

[5] P. Andritsos, A. Fuxman, and R. J. Miller. Clean Answers over Dirty Databases: A Probabilistic Approach. In *ICDE*, page 30, 2006.

[6] L. Antova, C. Koch, and D. Olteanu. Query Language Support for Incomplete Information in the MayBMS System. In *VLDB*, pages 1422–1425, 2007.

[7] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, 1990.

[8] C. Beeri, Y. Kanza, E. Safra, and Y. Sagiv. Object Fusion in Geographic Information Systems. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, pages 816–827, 2004.

[9] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. E. Sarma. Managing RFID Data. In *VLDB*, pages 1189–1195, 2004.

[10] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The multi-rule partial sequenced route query. In *GIS*, page 10, 2008.

[11] H. Chen, W.-S. Ku, and H. Wang. Cleansing uncertain databases leveraging aggregate constraints. In *ICDE Workshops*, pages 128–135, 2010.

[12] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun. Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing. In *SIGMOD Conference*, pages 51–62, 2010.

[13] R. Cheng, S. Singh, and S. Prabhakar. U-DBMS: A Database System for Managing Constantly-evolving Data. In *VLDB*, pages 1271–1274, 2005.

[14] R. Cocci, T. T. 0002, Y. Diao, and P. J. Shenoy. Efficient Data Interpretation and Compression over RFID Streams. In *ICDE*, pages 1445–1447, 2008.

[15] N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. *The VLDB Journal*, 16(4):523–544, 2007.

[16] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR*, pages 317–328, 2005.

[17] A. Doucet, S. Godsill, and C. Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10:197–208, 2000.

[18] D. W. Engels and S. E. Sarma. The Reader Collision Problem. In *IEEE SMC*, 2002.

[19] L. F. Escudero. An Inexact Algorithm for the Sequential Ordering Problem. *European Journal of Operational Research*, 37(2):236–249, 1988.

[20] C. Floerkemeier and M. Lampe. Issues with RFID Usage in Ubiquitous Computing Applications. In *Pervasive*, pages 188–193, 2004.

[21] M. J. Franklin, S. R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong. Design Considerations for High Fan-In Systems: The HiFi Approach. In *CIDR*, pages 290–304, 2005.

[22] B. George, S. Kim, and S. Shekhar. Spatio-temporal Network Databases and Routing Algorithms: A Summary of Results. In *Proceedings of the 10th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 460–477, 2007.

[23] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and Analyzing Massive RFID Data Sets. In *ICDE*, page 83, 2006.

[24] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD'84, Proceedings of Annual Meeting*, pages 47–57, 1984.

[25] G. R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *ACM Trans. Database Syst.*, 24(2):265–318, 1999.

[26] J. Ho, D. W. Engels, and S. E. Sarma. HiQ: A Hierarchical Q-learning Algorithm to Solve the Reader Collision Problem. In *SAINT Workshops*, pages 88–91, 2006.

[27] E. Horowitz, S. Sahni, and S. Anderson-Freed. *Fundamentals of Data Strucures in C.* W. H. Freeman, 1993.

[28] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. MCDB: A Monte Carlo Approach to Managing Uncertain Data. In *SIGMOD*, pages 687–700, 2008.

[29] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive*, pages 83–100, 2006.

[30] S. R. Jeffery, M. J. Franklin, and M. N. Garofalakis. An Adaptive RFID Middleware for Supporting Metaphysical Data Independence. *VLDB J.*, 17(2):265–289, 2008.

[31] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, pages 163–174, 2006.

[32] C. S. Jensen, J. Kolárvr, T. B. Pedersen, and I. Timko. Nearest Neighbor Queries in Road Networks. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, pages 1–8, 2003.

[33] A. B. Kahn. Topological Sorting of Large Networks. *Commun. ACM*, 5(11):558–562, 1962.

[34] Y. Kanza, R. Levin, E. Safra, and Y. Sagiv. An interactive approach to route search. In *GIS*, pages 408–411, 2009.

[35] N. Khoussainova, M. Balazinska, and D. Suciu. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. In *MobiDE*, pages 43–50, 2006.

[36] N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic Event Extraction from RFID Data. In *ICDE*, pages 1480–1482, 2008.

[37] M. R. Kolahdouzan and C. Shahabi. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*, pages 840–851, 2004.

[38] W.-S. Ku, R. Zimmermann, H. Wang, and C.-N. Wan. Adaptive Nearest Neighbor Queries in Travel Time Networks. In *Proceedings of the 13th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, pages 210–219, 2005.

[39] K. C. K. Lee, W.-C. Lee, and B. Zheng. Fast object search on road networks. In *EDBT*, pages 1018–1029, 2009.

[40] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On Trip Planning Queries in Spatial Databases. In *Proceedings of the 9th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 273–290, 2005.

[41] X. Ma, S. Shekhar, H. Xiong, and P. Zhang. Exploiting a Page-Level Upper Bound for Multi-Type Nearest Neighbor Queries. In *Proceedings of the 14th ACM International Symposium on Geographic Information Systems (ACM-GIS)*, pages 179–186, 2006.

[42] J. Myung, W. Lee, J. Srivastava, and T. K. Shih. Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification. *IEEE Trans. Parallel Distrib. Syst.*, 18(6):763–775, 2007.

[43] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query Processing in Spatial Network Databases. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 802–813, 2003.

[44] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A Deferred Cleansing Method for RFID Data Analytics. In *VLDB*, pages 175–186, 2006.

[45] R. Reddy. To dream the possible dream. *Commun. ACM*, 39(5):105–112, 1996.

[46] S. M. Ross. *Introduction to Probability Models, Ninth Edition*. Academic Press, 2006.

[47] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 71–79, 1995.

[48] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[49] H. Samet. Issues, Developments, and Challenges in Spatial Databases and Geographic Information Systems (gis). In *Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, page 1, 2001.

[50] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD Conference*, pages 43–54, 2008.

[51] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *Proceedings of 13th International Conference on Very Large Data Bases (VLDB)*, pages 507–518, 1987.

[52] M. Sharifzadeh, M. R. Kolahdouzan, and C. Shahabi. The Optimal Sequenced Route Query. *The VLDB Journal*, accepted for publication, 2008.

[53] M. Sharifzadeh and C. Shahabi. Processing optimal sequenced route queries using voronoi diagrams. *GeoInformatica*, 12(4):411–433, 2008.

[54] M. Sharifzadeh and C. Shahabi. Approximate voronoi cell computation on spatial data streams. *VLDB J.*, 18(1):57–75, 2009.

[55] S. Shekhar, M. Coyle, B. Goyal, D.-R. Liu, and S. Sarkar. Data Models in Geographic Information Systems. *Commun. ACM*, 40(4):103–111, 1997.

[56] L. Sullivan. RFID Implementation Challenges Persist, All This Time Later. *InformationWeek*, October 2005.

[57] Y. Tao, D. Papadias, and Q. Shen. Continuous Nearest Neighbor Search. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB)*, pages 287–298, 2002.

[58] M. Terrovitis, S. Bakiras, D. Papadias, and K. Mouratidis. Constrained Shortest Path Computation. In *Proceedings of the 9th International Symposium on Advances in Spatial and Temporal Databases (SSTD)*, pages 181–199, 2005.

[59] Y. Tian, K. C. K. Lee, and W.-C. Lee. Finding skyline paths in road networks. In *GIS*, pages 444–447, 2009.

[60] Y. Tian, K. C. K. Lee, and W.-C. Lee. Monitoring minimum cost paths on road networks. In *GIS*, pages 217–226, 2009.

[61] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy. Probabilistic Inference over RFID Streams in Mobile Environments. In *ICDE*, 2009.

[62] T. T. L. Tran, A. McGregor, Y. Diao, L. Peng, and A. Liu. Conditioning and Aggregating Uncertain Data Streams: Going Beyond Expectations. 2010.

[63] T. T. L. Tran, L. Peng, B. Li, Y. Diao, and A. Liu. PODS: A New Model and Processing Algorithms for Uncertain Data Streams. In *SIGMOD Conference*, pages 159–170, 2010.

[64] J. Waldrop, D. W. Engels, and S. E. Sarma. Colorwave: An Anticollision Algorithm for the Reader Collision Problem. In *ICC*, pages 1206– 1210, 2003.

[65] F. Wang and P. Liu. Temporal Management of RFID Data. In *VLDB*, pages 1128–1139, 2005.

[66] R. Want. The Magic of RFID. *ACM Queue*, 2(7):40–48, 2004.

[67] J. Xie, J. Yang, Y. Chen, H. Wang, and P. S. Yu. A Sampling-Based Approach to Information Recovery. In *ICDE*, pages 476–485, 2008.

[68] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee. Location-based Spatial Queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 443–454, 2003.