Aalborg Universitet

AALBORG UNIVERSITY

Online Risk Prediction for Indoor Moving Objects

Ahmed, Tanvir; Pedersen, Torben Bach; Calders, Toon; Lu, Hua

Published in:
17th IEEE International Conference on Mobile Data Management

DOI (link to publication from Publisher):
10.1109/MDM.2016.27

Publication date:
2016

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Ahmed, T., Pedersen, T. B., Calders, T., & Lu, H. (2016). Online Risk Prediction for Indoor Moving Objects. In *17th IEEE International Conference on Mobile Data Management* (pp. 102-111). IEEE Computer Society Press. https://doi.org/10.1109/MDM.2016.27

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: August 23, 2025

Online Risk Prediction for Indoor Moving Objects

Tanvir Ahmed[§†] Torben Bach Pedersen[§] Toon Calders[†] Hua Lu[§]
[§]Department of Computer Science, Aalborg University, Aalborg, Denmark
[†]Department of Computer and Decision Engineering, Université Libre de Bruxelles, Brussels, Belgium
Email: {tanvir, tbp, luhua}@cs.aau.dk, toon.calders@ulb.ac.be

***Abstract*—Technologies such as RFID and Bluetooth have received considerable attention for tracking indoor moving objects. In a time-critical indoor tracking scenario such as airport baggage handling, a bag has to move through a sequence of locations until it is loaded into the aircraft. Inefficiency or inaccuracy at any step can make the bag risky, i.e., the bag may be delayed at the airport or sent to a wrong airport. In this paper, we propose a novel probabilistic approach for predicting the risk of an indoor moving object in real-time. We propose a probabilistic flow graph (PFG) and an aggregated probabilistic flow graph (APFG) that capture the historical object transitions and the durations of the transitions. In the graphs, the probabilistic information is stored in a set of histograms. Then we use the flow graphs for obtaining a risk score of an online object and use it for predicting its riskiness. The paper reports a comprehensive experimental study with multiple synthetic data sets and a real baggage tracking data set. The experimental results show that the proposed method can identify the risky objects very accurately when they approach the bottleneck locations on their paths and can significantly reduce the operation cost.**

I. Introduction

Technologies such as RFID and Bluetooth enable a variety of indoor, outdoor, and mixed indoor-outdoor tracking applications. Examples of such applications include tracking people's movement in large indoor spaces (e.g., airport, office building, and shopping malls), airport baggage tracking, item movement tracking in supply chains, and package tracking in logistics systems. During the movement of the objects, these tracking applications record the symbolic locations of the objects at different time points. For example, consider an RFID baggage tracking application where RFID readers are deployed at the different baggage handling locations such as check-in, screening, sorter, etc. Each reader has a very limited tracking range that covers a small portion of the location. If an object containing an RFID tag moves from the check-in to the sorter, this produces two consecutive tracking records of the object location in different places. Due to the limitations in indoor positioning technologies, the locations between these two records are not obtained. We call this type of tracking Symbolic Location Tracking (SLT). An SLT system can generate a massive volume of tracking data. This massive tracking data can be very useful for analyses such as finding risk factors, problem discovery, and decision-making. For example, in an airport baggage handling system, a bag can be left behind in the airport (i.e., failed to catch the intended flight) or can be sent to a wrong airport. In the baggage tracking system, the baggage tracking data can be used to extract interesting patterns and find the reasons for baggage mishandling. In a supply chain system, the item tracking data can be used for finding the factors that lead to an item being returned or get rotten. Some work has been carried out for the efficient management of such tracking data and to analyze them in the offline scenario [8], [13]. However, using such data for time-critical online applications, such as online bags risk prediction in the airports can be very useful for getting real-time notifications for immediate handling of the risky bags. Moreover, online items risk discovery in supply chain and production systems, online item risk prediction in logistics systems, traffic jam prediction, etc., can also benefit from the insights obtained from analyzing the online data. An example of a real-time analysis request can be: "notify the baggage management team whenever a bag becomes risky during its processing time at Aalborg airport". Another request can be: "which are the 5 current bags with the highest risk of not reaching their plane on time?".

The paper makes several contributions. First, to the best of our knowledge, this is the first paper to propose a method for online risk prediction for indoor moving objects. Second, we propose the concepts of least duration probability (LDP), aggregated LDP (ALDP), LDP histogram (LDPH), and ALDP histogram (ALDPH) where the histograms store probabilistic information about the transition times of the historical objects. We propose a probabilistic flow graph (PFG) and aggregated PFG (APFG) that capture the flows of objects from one symbolic location to another and the edges of the graphs contain corresponding LDPH and ALDPH, respectively. Third, an online risk prediction (ORP) algorithm is proposed that uses the PFG and APFG for obtaining a risk score for an online indoor moving object. The risk score is used for predicting the riskiness of the object during its processing. Fourth, as the total available processing time of an object is an important factor, we propose an approach for normalizing the available processing time (e.g., available processing time for a bag before its flight) with the stay durations of objects at different locations for obtaining a better risk score. Fifth, we present a cost model for obtaining the best risk score threshold that can maximize the overall benefit of identifying and removing the risky objects. Sixth, the paper reports a comprehensive experimental study with several synthetic data sets following different data distributions and a real baggage tracking data set. The results show that the proposed method can produce a very accurate risk score and identify the risky objects very precisely in different types of data distributions.

The remainder of the paper is organized as follows. Sec-

tion II presents the SLT systems and tracking data. Section III discusses the problem formulation. Section IV presents the solution and probabilistic flow graph. Section V presents online risk prediction steps and the algorithm. Section VI reports the experimental results. Section VII reviews related work. Section VIII concludes and points to future work.

II. Preliminaries

SLT Systems. In an SLT system, tracking devices are strategically deployed at different fixed symbolic locations, such as different doors in an office space, between sections in an airport, different locations in airport baggage management, etc. The objects contain tags or devices that can be tracked by the tracking devices. For example, in the case of RFID technology, RFID readers and RFID tags are used; in the case of Bluetooth systems, Bluetooth access points and Bluetooth devices are used. After deployment of the tracking devices, the positions are recorded in the database.

Fig. 1 shows an example of airport baggage tracking scenario. The upper part of the figure shows the top level path of a bag that travels from Aalborg Airport (AAL) to Brussels Airport (BRU) via Copenhagen Airport (CPH). The bag has to go through several baggage processing steps inside each airport. The bottom part of the figure shows the baggage processing stages inside AAL. The circles represent the baggage tracking locations where RFID readers are deployed for baggage tracking. Before handing over a bag into the system, an RFID tag with some encoded information about the bag and the route is attached to the bag. Suppose the bag is intended for *Flight1* and the preplanned path for the bag is: "*check-in→Screening→Sorter1→Gateway1→BeltLoader1*". Mismanagement or inefficiency at any one of these transitions may result in the bag being mishandled, i.e., the bag might miss the flight due to delay, or the bag might be sent to a wrong flight. While passing through the different locations, the bag enters the activation range of an RFID reader, it is continuously detected by the reader with a sampling rate, and it generates *raw reading records* with the form: $\langle Obj, Loc, t \rangle$. It means that a reader placed at location *Loc* detects a moving object *Obj* in its activation range at time t. An example set of raw reading records in an *SLT* system is shown in Table I.

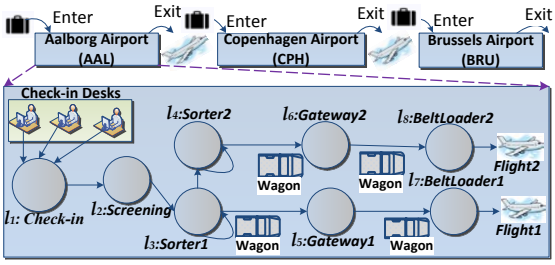


Fig. 1: Example SLT scenario in airport baggage tracking

StayRecords. As seen, the raw readings contain many redundant records. If an object stays for t time units under the activation range of a tracking device, it can generate

TABLE I: Raw Reading Data

Obj	$\langle$ *Obj, Loc, t* $\rangle$
o_1	$(o_1, l_1, 1)$ $(o_1, l_1, 3)$ $(o_1, l_1, 5)$ $(o_1, l_2, 12)$ $(o_1, l_2, 14)$ $(o_1, l_3, 25)$ $(o_1, l_3, 27)$ ...
o_1	$(o_2, l_1, 10)$ $(o_2, l_1, 12)$ $(o_2, l_2, 26)$ $o_2, l_3, 32)$ $(o_2, l_4, 39)$ $(o_2, l_4, 41)$ $(o_2, l_6, 46)$ $(o_2, l_8, 55)$ ...
...	...
o_{1000}	...

t/sampling rate records for that stay. Depending on the application scenario, the stay of an object under the activation range of a reader can vary. For example, in an airport baggage tracking scenario a bag continuously moves from one location to another and usually it stays for a very short period under the activation range. Besides, in a supply chain scenario an object can stay long (e.g., few hours, days) on a shelf and can stay for a long period under a reader. However, in any SLT application, an object moves from one symbolic location to another, and it is essential to know the total duration spent by the object between the locations. We create a table *StayRecord*$\langle Obj, L_{from}, L_{to}, t_s, t_e, Dur \rangle$, which represents that an object *Obj* first appeared at location L_{from} at time t_s and then first appeared at the next location L_{to} at time t_e. It took *Dur* time to go from the reader at L_{from} to the reader at L_{to} or in another way it spent *Dur* time between L_{from} and L_{to}. Table II shows an example of *StayRecord* table constructed for the raw reading records shown in Table I.

TABLE II: Stay records from Table I

Obj	*StayRecord*$\langle Obj, L_{from}, L_{to}, t_s, t_e, Dur \rangle$
o_1	$(o_1, l_1, l_2, 1, 12, 11)$ $(o_1, l_2, l_3, 12, 25, 13)$
o_2	$(o_2, l_1, l_2, 10, 26, 16)$ $(o_2, l_2, l_3, 26, 32, 6)$ $(o_2, l_3, l_4, 32, 39, 7)$ $(o_2, l_4, l_6, 39, 46, 7)$ $(o_2, l_6, l_8, 46, 55, 9)$
...	...
o_{1000}	$(o_{1000}, ..., .., ..., ..., ...)$ ...

III. Problem Formulation

We consider an application scenario where an object can be processed in a single system or multiple subsystems throughout its journey from the origin to the final destination. In the case of subsystems, when an object is registered in its origin, its identifier and the global route are shared within all the subsystems for further processing. Depending on the application scenario, an online risk prediction system can monitor the object throughout its entire journey from origin to final destination or it can monitor the object individually within each subsystem between its entry and exit times within that subsystem. For example, in Fig. 1, the global path of the object is *AAL→CPH→BRU*. The overall processing of the bag should be processed by the three subsystems, i.e., first at AAL, second at CPH, and third at BRU. Whenever the bag is first registered at AAL, its identifier, route and flight information is shared to CPH and BRU, so that they can recognize the bag when it appears to their systems. In this context, each subsystem has its separate online risk prediction system. Whenever the bag is first detected by an RFID reader at AAL, the bag becomes online to the local risk prediction system which starts monitoring the bag until it exits AAL, or

until it is confirmed that the bag misses its flight. Similarly, when the bag is detected at CPH, it becomes online at CPH and so on.

Definition 1. ***Online Object.*** An object is considered as an *online object* to a system/subsystem at time t, if t falls in the time interval $[t_{enter}, t_{exit}]$, where t_{enter} is the first time the object is tracked in a tracking device in the system/subsystem and t_{exit} is the last time the object is tracked by the last tracking device in the system/subsystem or the time within which the object is expected to exit the system/subsystem.

Problem Statement. Given a set of stay records R and a set of online moving objects O, we are interested in building a predictive model from R that can predict, as early as possible, whether an object $o_i \in O$ is at risk in real-time.

For example, in baggage tracking, the model should be able to predict whether a bag going through the baggage handling stages is at risk of being delayed at the airport and the prediction should be made as early as it sees the bag is being abnormally deferred compared to other bags.

IV. Solution

The overall outline of the data collection and risk prediction steps is shown in Fig. 2. The online object tracking data stream is passed into two sections. One of them stores the data offline for future analysis and model building purpose and another uses it during the *online risk prediction (ORP)* process. The offline/historical reading records are processed and converted into *StayRecords*. The *StayRecords* are used for building the probabilistic model. The model, raw data stream and the preplanned path of the objects are used by the ORP for deciding which objects are at risk. Finally, risky objects are notified by the ORP for special handling.

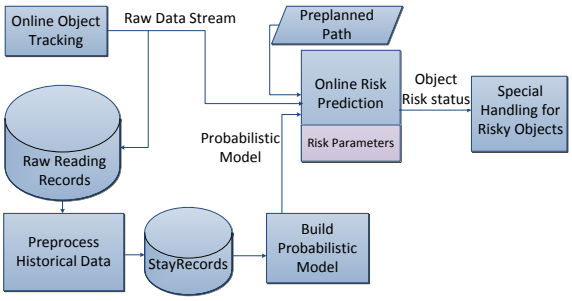


Fig. 2: Outline of the overall system

Let, L = $\{l_1, l_2, l_3, ..., l_n\}$ be the set of locations available in the data set. A set of durations taken by the transitions from location l_i to l_j be $D_{i,j}=\{d_1, d_2, d_3, \dots, d_n\}$.

Definition 2. ***Least Duration Probability (LDP).*** A least duration probability (LDP) for a movement from l_i to l_j with threshold duration $d_k \in D_{i,j}$ is defined as,

$$LDP(l_i, l_j, d_k^{\geqslant}) = \frac{Count(l_i, l_j, d_k^{\geqslant})}{Count(l_i, l_j)} \quad (1)$$

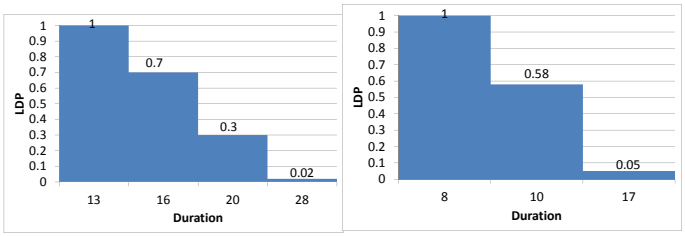
In Eq. (1), $Count(l_i, l_j, d_k^{\geqslant})$ is the total number of objects that took at least d_k duration from l_i to l_j and $Count(l_i, l_j)$ is the total number of objects that have a transition from l_i to l_j.

Definition 3. ***Least Duration Probability Histogram (LDPH).*** A least duration probability histogram (LDPH) for transitions from l_i to l_j is a histogram with transition durations $D_{i,j}$ on the X-axis, and LDPs for the transitions on the Y-axis.

TABLE III: Transition Summary (C stands for *Count*)

Transition $tr\ (l_i \rightarrow l_j)$	**Dur** (d_k)	$C(tr)$	$C(tr, d_k^{\geqslant})$	$LDP(tr, d_k^{\geqslant})$
$l_1 \rightarrow l_2$	13 16 20 28	1000	1000 700 300 20	1 0.7 0.3 0.02
$l_2 \rightarrow l_3$	8 10 17	1000	1000 580 50	1 0.58 0.05
$l_3 \rightarrow l_5$	60 70 98	470	470 250 50	1 0.53 0.11
$l_5 \rightarrow l_7$	40 50	460	460 250	1 0.54

Table III shows an example summary of transitions for the path l_1 to l_7 from the stay records in Table II. Fig 3 shows the different LDPHs for the transitions shown in Table III. In Fig. 3a, LDP=0.7 represents that the probability of transition from l_1 to l_2 with a duration $\geq$ 16 is 0.7. The figure also shows that the LDP for duration 28 is very low (0.02).



(a) LDPH for $l_1 \rightarrow l_2$ (b) LDPH for $l_2 \rightarrow l_3$

Fig. 3: LDPHs for the transitions $l_1 \rightarrow l_2$ and $l_2 \rightarrow l_3$ in Table III

Probabilistic Flow Graph (PFG). We use a probabilistic flow graph (PFG) for modeling the movement of objects from one symbolic location to another. The PFG is formally defined as a labeled directed graph $G = (L, E, D, H, lb_E)$, where:

1) L is the set of locations where each location is represented as a vertex in G.
2) E is the set of directed edges: $E = \{(l_i, l_j) \mid l_i, l_j \in L\}$.
3) D is the set of durations. $D_{i,j} \subseteq D$ represents the set of durations taken by objects for the transitions from l_i to l_j.
4) H is the set of LDPHs, where an $LDPH_{i,j} \in H$ is computed from the number of transitions from location l_i to l_j and the durations $D_{i,j} \subseteq D$.
5) lb_E is a function lb_E: $E \rightarrow H$ that labels an edge by an LDPH, $h \in H$. An edge $(l_i, l_j) \in E$ is labeled by an LDPH $LDPH_{i,j} \in H$, where $LDPH_{i,j}$ is the LDPH from l_i to l_j.

Fig. 4 shows the PFG constructed from the transition summary shown in Table III. Two LDPHs of Fig. 4 is shown in Fig 3. However, all the data for the rest of the LDPHs are available in the $LDP(tr, d_k^{\geqslant})$ column of Table III.

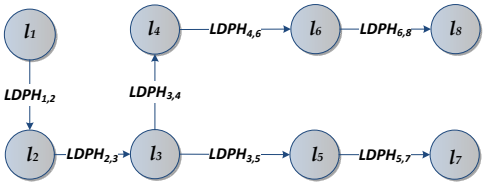


Fig. 4: Probabilistic flow graph (PFG)

V. Online Risk Prediction (ORP)

In an SLT system, the movements of the mishandled objects are expected to differ from the usual movement. They can take a wrong transition or can stay longer between planned locations. As the PFG is learned from the historical data, we use it for obtaining a probability score of an online object and use the score for predicting the unusual movements.

We consider a scenario, where the path of a given online object is predefined. For example, in the case of the baggage tracking, all the bags intended for a particular flight *SK123* should follow the same path sequence starting from the check-in desk up to the belt loader to the aircraft (e.g., $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_5 \rightarrow l_7$). If the object does not follow its preplanned path, it is triggered as risky. However, an object following its preplanned path, but taking a unusually longer duration for a transition can make the object risky. We use two different thresholds to decide the riskiness of an object. The method of finding the threshold is discussed at the end of this section. For each transition $l_i \rightarrow l_j$ in the PFG, we use an LDP threshold $LDP_{th}(l_i, l_j)$, that helps to get the maximum acceptable stay duration ($Dur_{max}(l_i, l_j)$) of an object between l_i and l_j. If an object spends equal or more than $Dur_{max}(l_i, l_j)$ between l_i and l_j, the object is considered at risk. Another threshold is called *risk score threshold* (RS_{th}). After each transition of an object o, its *combined duration probability (CDP)* for the so far traversed path is computed by multiplying the LDPs for the transitions as shown in Eq. (2). In Eq. (2), n is the total number of transitions of o and LDP'_i is the LDP for the stay duration of o obtained from the LDPH for o's i_{th} transition. The value of CDP is converted into *risk score (RS)* by, *RS = 1-CDP*. If $RS \geq RS_{th}$, we trigger that o is at risk. Maintaining the value of *RS* helps to find the top-k risky online objects in the system.

$$CDP(o) = \prod_{i=1}^{n} LDP'_i \tag{2}$$

Generally, when a PFG is learned from a large data set, the LDPHs should contain most of the possible stay durations for the upcoming new objects. However, if a new object o takes d duration for a transition $l_i \rightarrow l_j$ and d is not directly available in LDPH(l_i,l_j), the value of LDP (l_i, l_j, $d^{\geqslant}$) is computed in one of the following ways:

- If $d < d_{first}$ (the first entry of LDPH(l_i,l_j)), then LDP (l_i, l_j, $d^{\geqslant}$) = LDP(l_i, l_j, $d^{\geqslant}_{first}$).
- If $d > d_{last}$ (the last entry of LDPH(l_i,l_j)), then LDP (l_i, l_j, $d^{\geqslant}$) = LDP(l_i, l_j, $d^{\geqslant}_{last}$).
- For the other cases we use linear interpolation to obtain the value of LDP(l_i, l_j, $d^{\geqslant}$) from LDPH(l_i,l_j).

Furthermore, as the new online object becomes part of the historical data after its operation, its new duration is included in the PFG next time when a new model is built. Now coming to the ORP, if $LDP_{th}(l_i, l_j)$ is not directly available from LDPH(l_i, l_j), we use linear interpolation for computing the duration (i.e., the value in X-axis) for that LDP_{th} and use it as the Dur_{max} for that transition.

For example, consider an object o following a path: $l_1 \xrightarrow{17} l_2 \xrightarrow{17} l_3 \xrightarrow{60} l_5 \xrightarrow{40} l_7$. The labels in the arrows represent the duration taken for the transitions. Let us consider that the object followed its preplanned path. As the LDPH(l_1,l_2) has no entry for 17, its expected value by linear interpolation is, LDP (l_1,l_2,$17^{\geqslant}$) = 0.7-(0.7-0.3)/(20-16)×(17-16) = 0.6. The full CDP for the object = LDP(l_1, l_2, $17^{\geqslant}$) × LDP(l_2, l_3, $17^{\geqslant}$) × LDP(l_3, l_5, $60^{\geqslant}$) × LDP(l_5, l_7, $40^{\geqslant}$) = 0.6×0.05×1×1 = 0.03. Let us consider that RS_{th} = 0.8 and LDP_{th} for each of the transitions is 0.2. So, $Dur_{max}(l_1, l_2)$ by linear interpolation = $\lceil 20 + (0.3 - 0.2)/(0.3 - 0.02) \times (28 - 20) \rceil$ = 23. Similarly, $Dur_{max}(l_2, l_3)$=15. When o completes its first transition (i.e., $l_1 \xrightarrow{17} l_2$), it passes both of the LDP and RS checks for that transition as the spent duration $17 < Dur_{max}$ = 23 and RS = 1-CDP = 1-0.6 = $0.4 \not\geq RS_{th}$=0.8. So, the bag is not risky until the current state. When o reaches at l_3, the spent duration $17 \not< Dur_{max}$ = 15. Furthermore, the CDP of o up to this location is 0.03 (0.6×0.05). So, RS = 1-0.03 = $0.97 \geq RS_{th}$=0.8. So, o is considered as a risky object after this transition in terms of both Dur_{max} and RS_{th}.

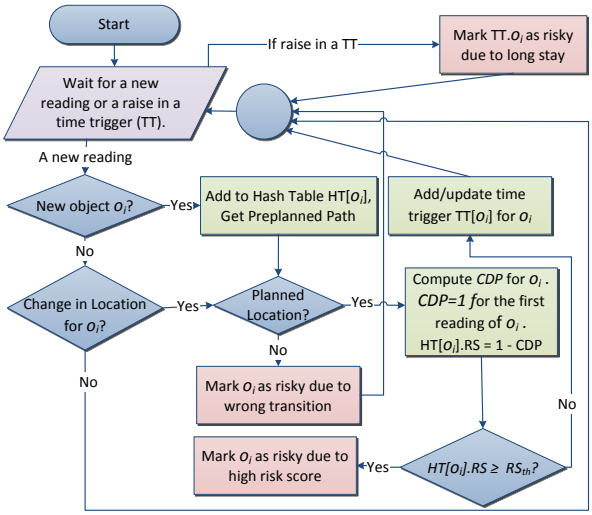


Fig. 5: Online risk prediction steps

The overall processing steps of the ORP are shown in Fig. 5. The process continuously waits for new readings. When a new reading arrives, it checks whether the object o_i in the reading is new. If o_i is new, it is inserted into a hash table (HT) and its preplanned path is retrieved from the system. If o_i does not follow its preplanned path, it is marked as risky due to the wrong location. However, if the path is correct, its CDP is initialized to 1. Based on the LDP_{th} of the current location l_{curr} and the planned next location $l_{p_{next}}$, a time trigger $TT_{o_i}(o_i, t_{start}, t_{e_{max}})$ is added with o_i, where t_{start} is

the first reading time of o_i at l_{curr} and $t_{e_{max}}$=t_{start}+Dur_{max}. As mentioned earlier, the value of Dur_{max} is extracted from the corresponding LDPH. If o_i remains between l_{curr} and $l_{p_{next}}$ until the clock time reaches $t_{e_{max}}$, the time trigger TT_{o_i} is raised and the trigger marks o_i as risky. Coming back to the starting point, if the new reading contains an old object, the process checks whether the object has changed its location. If it is in the same location, the process continues waiting for a new reading and a raise of a time trigger. However, if the object changes its location, its planned location is checked and based on that its further processing such as CDP computation, RS checking, time trigger update, etc., is performed. The time trigger allows a fast notification about a risky object as the process does not have to wait to complete the transition.

Algorithm 1 shows the processing of the ORP. It takes a hash table, RS_{th}, LDP$_{th}$ list, and PFG as the input and updates the hash table as the result. The algorithm continuously waits for a new reading or a raise of a time trigger (lines 1-2). If a new reading arrives, based on the data in the new reading, it updates the hash table HT. First, it checks whether the object o_i in the new reading is newly arrived in the system (line 5). If o_i is new, it is inserted in HT (line 6), and its preplanned path is checked (lines 6-9). If o_i is in the planned location, it is initialized to a safe object (lines 11-12). Based on the next planned location and LDP_{th}, its Dur_{max} is extracted from the PFG (lines 13-14), maximum clock time threshold $t_{e_{max}}$ is calculated (line 15), and a time trigger is registered for o_i (line 16). If o_i is not new, it is checked whether o_i has changed its location (line 17). If o_i has not changed its location, the algorithm continues waiting for a new reading or a raise in a time trigger. Conversely, if o_i changes its location in the new reading, its planned path is checked (lines 18-21). If it is in the planned path, o_i's time trigger is updated with the new information (lines 22-25). After that, o_i's stay duration for the transition is computed, CDP and RS are calculated, and the RS is checked with the RS_{th} (lines 26-31). Besides, if a time trigger is fired, the corresponding object is notified as risky (lines 33-35).

Recovery Scenario. The PFG cannot capture the possibility of a recovery of an object from its risky state. For example, an object might take a long duration between location l_1 to l_2 that makes it risky. However, it might be handled very quickly in its next transition l_2 to l_3 that recovers the object from being mishandled. To capture this, we modify the PFG into *aggregate probability flow graph (APFG)*. Here, we additionally maintain an *aggregate LDPH (ALDPH)* for each path sequence S=$l_i l_{i+1} l_{i+3} ... l_n$, where l_i must be the first tracking location of at least one object in the data set and i<n<=p (the length of the path sequence). An *ALDPH (S)* contains all the *aggregate LDPs (ALDP)* for S. An *ALDP(S,$d^{\geqslant}$)* represents the probability of taking at least a duration of d by an object for completing the path sequence S. The value of an ALDP for the path sequence S with a total duration d is computed by Eq. (3). In Eq. (3), $Count(S, d^{\geqslant})$ is the number of objects taking at least a d duration to complete the path sequence S and $Count(S)$ is the number of objects traveling through path sequence S. Table IV shows the ALDPs and data for ALDPHs for the path from l_1 to l_7 in our example scenario.

Algorithm 1: ORP(HashTable HT, RS_{th}, LDPThresholdList LDP_{th}, PFG) **Result:** *Hash Table with Risk Status*

```
1  while true do
2      wait for a new reading or a raise in a time trigger;
3      if a new reading rr arrives then
4          o_i ← rr.Obj; l_cur = rr.Loc;
5          if HT[o_i] = NULL then
6              HT.Insert(o_i); PP[o_i]←PlannedPath(o_i);
7              if l_cur ≠ PP[o_i].POP() then
8                  HT[o_i].status ← "Risky";
9                  HT[o_i].Reason←"WrongTran"; continue;
10             HT[o_i].Loc_cur ← HT[o_i].Loc_prev ← rr.Loc;
11             HT[o_i].t_s←rr.t;HT[o_i].CDP←1;HT[o_i].RS←0;
12             HT[o_i].status←HT[o_i].Reason←"NotRisky";
13             l_pnext←HT[o_i].Loc_pNext←PP[o_i].POP();
14             Dur_max←PFG.GetDur(LDP_th, l_cur, l_pnext));
15             MaxTimeEnd t_emax ← rr.t+Dur_max;
16             TT[o_i] ← TimeTrigger(o_i, rr.t, t_emax);
17         else if HT[o_i].Loc_prev ≠ l_cur then
18             l_pcur ← PP[o_i].POP();
19             if l_cur ≠ l_pcur then
20                 HT[o_i].status ← "Risky"; HT[o_i].RS ← 1;
21                 HT[o_i].Reason←"WrongTran"; continue;
22             l_pnext←HT[o_i].Loc_pNext←PP[o_i].POP();
23             Dur_max←PFG.GetDur(LDP_th, l_cur, l_pnext));
24             MaxTimeEnd t_emax ← rr.t+Dur_max;
25             TT[o_i] ← TimeTrigger(o_i, rr.t, t_emax);
26             SpentDuration dur ← rr.t - HT[o_i].t_s;
27             HT[o_i].Loc_cur←l_cur; HT[o_i].t_s←rr.t;
28             HT[o_i].CDP←HT[o_i].CDP×LDP(HT[o_i].Loc_prev,
               l_cur, dur^≥); HT[o_i].RS ← 1- HT[o_i].CDP ;
29             if HT[o_i].RS ≥ RS_th then
30                 HT[o_i].status ← "Risky";
31                 HT[o_i].Reason ← "High RS";
32             HT[o_i].Loc_prev ← l_cur;
33     else if a time trigger is raised for the object o_j then
34         HT[o_j].status ← "Risky";
35         HT[o_j].Reason ← "Long Stay Triggered";
```

$$ALDP(S, d^{\geqslant}) = \frac{Count(S, d^{\geqslant})}{Count(S)} \tag{3}$$

The processing of the ORP with the APFG is very similar to the algorithm discussed above with some additional conditions and operations. First, after each transition, in addition to the CDP computation, the ALDP for the traveled path is extracted from the corresponding ALDPH. If CDP<$ALDP$, then CDP is updated with the value of ALDP to make the score less risky. Also note that for the first transition, the values of ALDP and LDP are the same. Second, instead of using LDP_{th} for each transition, we maintain an *ALDP$_{th}$(s_i)* for each path sequence s_i for each of the preplanned paths. In our example scenario, for the path from l_1 to l_7, there will be ALDP thresholds for each of the path sequences mentioned in Table IV. Third,

TABLE IV: Path Summary

Path (S)	**Dur** (d)	***Count***(S)	***Count(S,***$d^{\geqslant}$***)***	***ALDP(S,***$d^{\geqslant}$***)***
$l_1 \rightarrow l_2 \rightarrow l_3$	21	600	600	1
	23		500	0.83
	24		300	0.5
	30		150	0.25
	33		30	0.05
$l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_5$	81	470	470	1
	83		415	0.88
	84		245	0.52
	94		195	0.41
	100		120	0.26
	121		50	0.11
	128		20	0.04
$l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_5 \rightarrow l_7$	123	460	460	1
	131		290	0.63
	134		235	0.51
	144		145	0.32
	150		110	0.24
	171		40	0.09
	178		10	0.02

the concept of $Dur_{max}(l_i, l_j)$ is changed to $Dur_{max}(s_i)$, where Dur_{max} represents the maximum allowable duration for an object to complete the path sequence s_i. The value of $Dur_{max}(s_i)$ can be extracted from the *ALDPH(s_i)* based on the *ALDP$_{th}$(s_i)*. Fourth, the concept of the time trigger is updated with the concept of ALDP and its structure is changed to TT_{o_i} *(s_i, $l_{p_{next}}$, t_{start}, $t_{e_{max}}$)*, where s_i is the so far completed path sequence by o_i, $l_{p_{next}}$ is the next planned location, t_{start} is the timestamp when o_i first tracked in the system, and $t_{e_{max}}$=t_{start}+$Dur_{max}(s_{p_{next}})$, where $s_{p_{next}}$ is the path sequence up to $l_{p_{next}}$ (i.e., $s_i \rightarrow l_{p_{next}}$). Then the rest of the procedure is the same as the above algorithm.

Consider the example discussed above where an object o followed a path: $l_1 \xrightarrow{17} l_2 \xrightarrow{17} l_3 \xrightarrow{60} l_5 \xrightarrow{40} l_7$. In the above example, o was marked as risky when it completed the path up to l_3. From Table IV, the ALDP up to l_3 is 0.05 (as the total duration=17+17=34). In terms of both LDP and ALDP, o is at risk at that point. After the next transition (i.e., up to l_5), the ALDP is 0.41, thus RS = $0.59 \ngeqslant RS_{th}$=0.8 (as total duration=17+17+60=94). However, the value of CDP up to l_5 is 0.6×0.05×1 = 0.03. The value of CDP either decreases or remains same while multiplying new LDPs. The new score shows that the object recovered from its risky state as it was processed quickly between l_3 and l_5. So, we update the CDP with the value of ALDP and mark o as not risky. When o moves further, the time trigger for o is also updated based on the traversed path sequence, preplanned path, and $ALDP_{th}$.

Time Constrained ORP. Generally, a slow processing of an object at a location makes the object risky. This slow processing could also result in a dense location or traffic jam that could hamper the processing of the upcoming objects. However, there are many applications where an object has to reach a particular location within a given timestamp. For example, in the baggage tracking, a bag has to be loaded in the aircraft before the scheduled flight departure. So, the available duration before the flight departure is an important factor for baggage risk prediction. If a bag starts its processing well in advance before the flight departure, it is less risky, even if it stays longer for a transition. Conversely, a bag having a short duration before the flight makes it risky, even it is processed relatively quickly in its transitions. So, the stay duration should be normalized with the available processing time and use the normalized duration for taking the corresponding ALDP to reflect the actual riskiness of the object.

Let us consider, t_{enter} be the first time an object o detected in the system and t_{final} be the maximum timestamp when o should reach its final reading point/location. So, the total available duration for o is d_a=t_{final}-t_{enter}. The expected average duration of travel of an object is extracted from the *ALDPH* for the full preplanned path of the object. Let d_e be that expected duration extracted from the *ALDPH* with *ALDP* = 0.5. After each transition of o, its normalized total stay duration for the so far traversed path is computed by Eq. (4), where *dur*$_i$ is the stay duration of o for its i_{th} transition. In the equation, the value of *offset* is computed initially by subtracting the value of d_a from d_e. Then, after the k_{th} transition of o, its total travel time up to that transition is added to the offset for obtaining the normalized duration. So, instead of taking the ALDP directly for the total duration d_t, we take the ALDP for d_n. Depending on the value of d_a and d_e, the value of *offset* as well as d_n can be negative. As discussed earlier about picking the LDP from an LDPH for a given duration, the value of ALDP for d_n is also taken in the same way from the corresponding ALDPH.

$$d_n(o) = \textit{Offset} + \sum_{i=1}^{k} dur_i, \textit{where offset} = (d_e - d_a) \quad (4)$$

For example, consider an object o_1 following its preplanned path and the stay durations for the transitions are: $l_1 \xrightarrow{17} l_2 \xrightarrow{17} l_3 \xrightarrow{94} l_5 \xrightarrow{50} l_7$. o_1 has a total of 200 seconds to reach l_7 from l_1. So, d_a=200 sec. From Table IV, d_e=134 (as ALDP for 134 is 0.51). So, *offset*=134-200 = -66. Now, for the first transition $l_1 \xrightarrow{17} l_2$, d_n = -66 + 17 = -49. So, from Table III, the value of ALDP or LDP for the transition is 1. It shows that instead of taking the actual LDP for duration 17 (which was 0.6 as computed earlier), we take the LDP for normalized duration. As o has plenty of time to reach l_7, the normalization makes the object less risky. After the next transition to l_3, d_n = -66+17+17 = -32. So, the ALDP after normalization is 1. Before the normalization, the ALDP was 0.05. However, after normalization the score says that the object is completely safe until that transition. Similarly, when o_1 reaches at l_7, the total stay duration is 178 and the normalized duration is 112. Thus, without normalization the ALDP is 0.02 and with normalization ALDP is 1. It shows that, even o_1 takes long for its transitions, the normalization marks it as a safe object as it has a long available time to reach its destination.

Adjusting *Dur$_{max}$* and Time Trigger. During processing of the ORP, $Dur_{max}(s_i)$ is adjusted to the concept of normalization. The normalized maximum allowable duration of an object o for completing its path sequence s_i is computed by, *Dur$_{maxN}$*(s_i, o) = *Dur$_{max}$*(s_i)−*offset*. As seen, if the value of *offset* is negative, then *Dur$_{maxN}$* allows more time to

o_i. Besides, the higher value of *offset* will reduce the value of Dur_{maxN} for adjusting the riskiness of o. Finally, $t_{e_{max}}$ in the corresponding time trigger is computed by, $t_{e_{max}}$= t_{start}+Dur_{maxN} and is used for the risk prediction.

Finding the best thresholds. The optimal threshold depends on the particular goal of the system. We consider mishandled as a positive class for classification. A prediction system, giving too many false positives (FP) (i.e., predicting correctly handled objects as the mishandled objects) or false negatives (FN) (i.e., predicting mishandled objects as the correctly handled objects) can make the system useless or not interesting. So, there should be a defined acceptable metric for deciding the optimal operational threshold. We define a benefit function based on the operation cost, where the costs for the different kinds of errors are used for finding the threshold that maximizes the benefit. For example, In the case of baggage tracking, if a bag is predicted as mishandled, it requires a special manual handling so that the bag can reach the aircraft before the flight. If an FP occurs, there will be a waste in the human resource cost for the mistake. However, if an FN occurs, there will be a significant cost to deliver the bag to the passenger's address and insurance and other operating costs are involved for such mistakes as well. So, in the baggage tracking scenario, the cost for an FN is much more compared to that for an FP. During model building and testing (discussed further in Section VI), we use Eq. (5) for obtaining the total benefit for each of the generated thresholds and use the threshold that provides the maximum benefit. In Eq. (5), x= cost for handling a mishandled object (i.e., positive case (P)), y=cost for handling a predicted mishandled object (i.e., TP and FP), and #P is the total number of positive cases in the data set. So, Eq. (5) can provide an idea how much money can be saved by using the ORP system.

$$\textit{Benefit}(x, y) = x \times \#\text{P} - (x \times \#\text{FN} + y \times (\#\text{TP} + \#\text{FP})) \quad (5)$$

VI. Experimental Evaluation

The PFG and APFG are implemented using a set of SQL statements and the prediction is implemented in C#. For all SQL queries, we use a leading RDBMS. The experiments are conducted on a laptop with an Intel Core i7 2.7 GHz processor with 8 GB RAM. The operating system is Windows 7 64 bit.

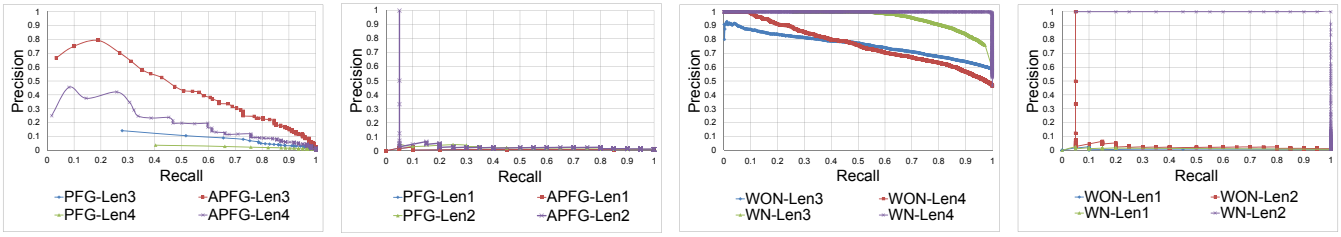
A. Data Sets Descriptions

We use both synthetic and real data for experimenting the different aspects of the proposed systems. The real data set reflects a specific scenario and contains a lot of erroneous readings, miss readings, and other anomalies. So, only experimenting with such real data cannot provide the other aspects of the prediction systems. Furthermore, synthetic data can be generated in different ways to see how the models perform with different ratios and distributions. During model building and prediction, it is assumed that the data set is cleaned.

Synthetic data sets. We generate 5 different data sets for the airport baggage tracking scenario, where bags follow the paths shown in the floor plan in Fig 1. There are two preplanned paths, *P1*: $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_5 \rightarrow l_7$, and *P2*: $l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \rightarrow l_6 \rightarrow l_8$. There are 20 flights a day and each flight departs after every 30 to 60 minutes. The first flight of a day starts at 8:00 am. In each of the data sets, there are a total of 5K flights carrying 100K bags. Each of the flights has 20 registered bags. Each data set contains approximately 450K stay records. Flight IDs and bag IDs are generated sequentially and the flight with even IDs are allocated to path P1 and others to P2. Bags are checked in at the earliest 3 hours and the latest 30 minutes before the flights. For each possible transition, bags follow a realistic range of duration with different distributions that will be discussed next. In our example scenario, the transitions $l_1 \rightarrow l_2$ and $l_2 \rightarrow l_3$ have less influence on baggage mishandling, whereas the sorters (i.e., $l_3 \rightarrow l_5$, $l_3 \rightarrow l_4$, and $l_4 \rightarrow l_6$) have higher influence. So, we put relatively smaller time intervals for those transitions. The duration ranges (in seconds) for different paths are, P1: $l_1 \xrightarrow{25-60} l_2 \xrightarrow{50-300} l_3 \xrightarrow{120-9000} l_5 \xrightarrow{400-500} l_7$, and for P2: $l_1 \xrightarrow{25-60} l_2 \xrightarrow{50-300} l_3 \xrightarrow{120-2500} l_4 \xrightarrow{120-5400} l_6 \xrightarrow{400-500} l_8$.

Varying the distributions of the durations for the transitions and durations before the flights, we generate 5 different data sets (DS). Each data set is divided into a training set (TRS) and a test set (TSS) containing 70K and 30K bags, respectively. It is also made sure that the bags for the same flight are not be distributed between training and test set as it might give a biased estimation due to overfitting. We also use validation sets from the TRS for cross validation while finding the best value for RS_{th} that will be discussed later in this section. The data sets are described below:

- **DS_1:** Transition durations and durations before flights are uniformly distributed. DS_1 contains 53% mishandled bags.
- **DS_2:** Transition durations follow a normal distribution and durations before flights follow uniform distribution. DS_2 can show the effect in the models when the transition durations are normally distributed compared to the uniform distribution of DS_1. DS_2 contains 54% mishandled bags.
- **DS_3:** Transition durations follow a log-normal distribution and durations before flights follow uniform distribution. As a log-normal distribution creates long tail, it generates less mishandled bags compared to DS_1 and DS_2. This distribution reflects a more realistic scenario of airport baggage tracking. The data set contains 12% mishandled bags.
- **DS_4:** Transition durations follow a log-normal distribution with different μ and σ compared to DS_3 and durations before flights follow a normal distribution. The main intention is to reduce the mishandling rate to below 2%. It also can expose how good the models are when the mishandling rate is very low. DS_4 contains 1.42% mishandled bags.
- **DS_5:** The distributions of durations are similar to DS_4. However, the bags flow from the opposite direction. So, in DS_5, *P1*=$l_7 \rightarrow l_5 \rightarrow l_3 \rightarrow l_2 \rightarrow l_1$, and *P2*=$l_8 \rightarrow l_6 \rightarrow l_4 \rightarrow l_3 \rightarrow l_2 \rightarrow l_1$. As seen, the change in direction of the path brings the sorter in the earlier step. In the sorter bags generally spend most of its operational time and considered as the bottleneck of the system. The data set can show how bringing bottleneck



(a) P1-PFG vs. P1-APFG for DS_4 (b) P_{r2}-PFG vs. P_{r2}-APFG, DS_R (c) With vs. without norm., DS_1 (d) With vs. without norm., DS_R

Fig. 6: PR curves for comparing PFG vs. APFG, and with (WN) vs. without normalization (WON) while using APFG

earlier in the path can affect the models. DS_5 contains 1.53% mishandled bags.

Real data sets (DS_R). We use a small real RFID baggage tracking data set from the departure system of an airport $A1$. For the reason of confidentiality, the airports' names are not disclosed. The bags are originated from $A1$ to the destination airport $A2$. In $A1$, there are 6 RFID readers deployed. Four of them are for departure system that includes Check-in, Sorter, Gateway1, and BeltLoader and 2 of them for arrival system. From the data set we derived three different preplanned paths, P_{r1}: Check-in→Sorter→Gateway1→BeltLoader, P_{r2}: Check-in→Sorter→Gateway1, and P_{r3}: Check-in→Sorter→BeltLoader. After removing many noisy records, we have a total of 20.4K bags for 2.5K different flights. There are only 75 mishandled (MH) bags which are only 0.35% of the total bags. The details for the training set are: total 15.9K, MH 29 (0.18%), P_{r1}-[total 1.5K, MH 7], P_{r2}-[total 10.1K, MH 4], P_{r3}-[4.3K, MH 18]. The test set details are: total 4.6K, MH 43, P_{r1}-[total 1.5K, MH 20], P_{r2}-[total 2.6K, MH 20], P_{r3}-[total 0.5K, MH 3].

B. Test Cases

We build PFGs and APFGs from all the mentioned data sets and tested them from various perspectives. The PFGs and APFGs are built from the combined records (i.e., containing all the paths in the data sets) called C-PFG, and C-APFG, respectively. We also separately build PFGs and APFGs with the records of each different path, e.g., P1-PFG, P1-APFG for P1 in synthetic data, P_{r1}-PFG, P_{r1}-APFG for P_{r1} in real data, etc. The PFGs and APFGs are tested on the test set for the relevant paths. We test them without normalizing (WON) and with normalizing (WN) the durations before flights.

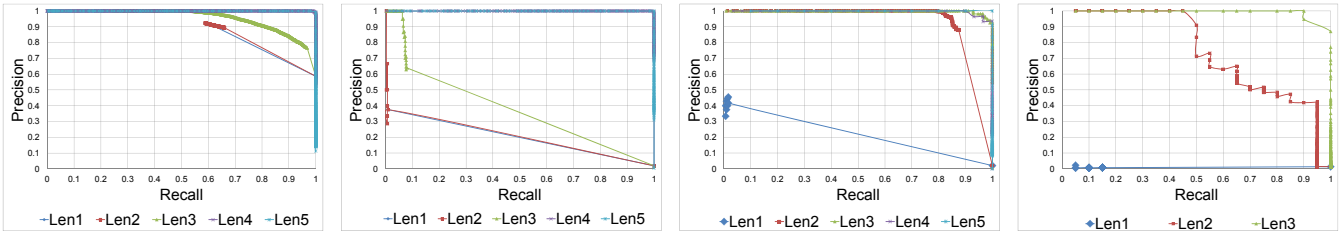
We apply the PFGs and APFGs on all the bags of the TSS and for each bag, we obtain a risk score for each of its transitions based on their transition duration. For each of the generated risk scores r, we compute the *recall*, where *recall(r)*= $\frac{\text{\# of mishandled bags having risk score} \geq \text{r}}{\text{\#of mishandled bags}}$. Conversely, for each r, we also compute the *precision*, where *precision(r)*= $\frac{\text{\# of mishandled bags having a risk score} \geq \text{r}}{\text{\#of bags having a risk score} \geq \text{r}}$. In our scenario, a perfect precision score of 1 means that all the classified mishandled bags are also actually mishandled. However, this precision score says nothing about whether all mishandled bags are predicted correctly. Conversely, a perfect recall score of 1 means that all the actually mishandled bags are classified as mishandled. However, this recall says nothing about how many correctly handled bags are wrongly predicted as mishandled. We draw *precision-recall (PR) curves* that represent how precision and recall changes with the risk scores. The perfect point in a PR curve is (1,1) that represents the predictions for the mishandled bags are perfect and any correctly handled bags are not predicted as mishandled.

We generate PR curves for the various test cases discussed above and report only the cases that are interesting to analyze. For all the test cases we analyze the PR curves for the different transition lengths, e.g., for path P1 there are 4 different lengths of transitions l_1 to l_2 (*Len1*), l_1 to l_3 (*Len2*), ..., and l_1 to l_7 (*Len4*). Similarly, P2 has 5 different lengths of transitions. For the real data, P_{r1} has 3 different transitions, P_{r2} and P_{r3} have 2 different lengths of transitions. The PR curves for the different test cases are reported in Fig. 6 and 7. The PR curves are analyzed from the different perspectives and explained next. The PR curves for DS_2 and DS_3 are not reported as they show the same behavior as others.

C. Analyzing the PR Curves

PFGs vs. APFGs. The PR curves for comparing the PFGs and APFGs are shown in Fig. 6a and 6b. Fig. 6a reports the PR curves generated by applying P1-PFG and P1-APFG on the TSS for P1 in DS_4. It reports the results for Len3 and Len4. PFG-Len3 can be compared with APFG-Len3 and so on. The results show that the APFGs always provide higher precisions compared to the PFGs. As we found same type behavior for the other DSs, they are not reported due to space limitation. Fig. 6b shows the similar experiments with DS_R. For Len1 PFG and APFG provide the same score. So, the lines are overlapped. However, for Len2, the APFG provides relatively better results. Overall from the experimental results, it is clear that the APFGs can better capture the riskiness and recovery of the objects and better differentiate between the correctly and incorrectly handled bags. In rest of the experiments, we report only the results with APFG.

With and without normalization. The PR curves for comparing the results with and without normalizing the duration before flights are presented in Fig. 6c and 6d. In Fig. 6c, P2-APFG is applied on the records for P2 in DS_1. In Fig. 6d, we use the same APFG used for Fig. 6b. In all cases, the results show that the normalizing boosts the performance. In all the cases except WN-Len1 (Fig. 6d), we can get almost a perfect classification, i.e., close to full precision with 100% recall when normalizing. The results for WN-Len1 can be understood better in the next paragraph.



(a) C-APFG for DS_1 tested on P2 (b) C-APFG for DS_4 tested on P2 (c) C-APFG for DS_5 tested on P2 (d) C-APFG, DS_R tested on P_{r1}

Fig. 7: Comparing the effect of path lengths and location types on APFGs while applying with normalization

Influence of path length and location type. Fig. 7 reports the PR curves for showing the effect of path length on the classification performance. Overall, the results show that the performance gets better with increasing the path length. In the case of DS_1 (Fig. 7a) and DS_4 (Fig. 7b), Len1 and Len2 have less influence on a baggage mishandling. These transitions also take very short durations. As a result, the performance is poor up to those transitions. However, in the case of DS_5, which contains the same distribution as DS_4, with the direction of the path is reversed, the performances from Len2 and afterward are close to perfect classification. The main mishandling occurs in the sorting system as a bag takes longer for completing its sortation. So, it shows that the model can classify mishandled bags very accurately when they come in the bottleneck in their path and the result continues getting better as objects move forward. The result with DS_R also shows the similar behavior (Fig. 7d).

Combined model vs. specialized model for each path. Comparing the results of Fig. 6c with Fig. 7a shows that testing the bags with path P2 by the C-APFGs and P2-APFGs provides the same result. In all the experimental cases we found that testing the bags with combined model and specialized model provide the same results. However, it is also true that the ALDPHs of a C-APFG become specialized for the different paths when they start following different path sequences. In our generated data set, P1 and P2 have a common path from l_1 to l_3. After that, they follow different path sequences. So, it will be best to use only combined model, instead of building many models for different paths.

Effect of data distribution and mishandle ratio. In general, the data distributions do not change the overall behavior of the models. In DS_1, the mishandling rate is balanced. So, it starts giving very good precision from Len1 (Fig. 7a) compared to the other cases where the mishandling rate is extremely low (Fig. 7b to 7d). However, the models built from all our different data set provide very good results. It also shows the proposed APFG with normalization can perform very well in an imbalanced class situation and does not get affected by the class imbalance problem.

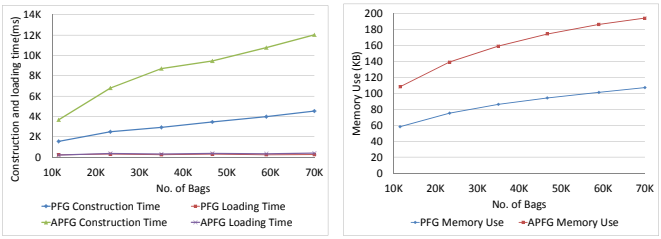
Finding the best RS_{th}. We use DS_4 in this experiment. TRS of DS_4 is divided into 10 folds for the standard k-fold cross validation. Each fold contains 17.4K bags, where almost half of them belong to P1. It is also made sure that the bags from the same flight are not distributed to multiple folds. We do not use DS_R to show this experiment as DS_R is small and dividing it into multiple folds will make it even smaller for learning. Iteratively, we learn APFGs from 9 folds and use the 10th fold for testing. So, finally we have the test results for 10 APFGs. We use Eq. (5) with x=\$96 (according to [1]) and y=\$15 (salary of a baggage handler is app. \$13/hour) for obtaining the benefit for the different RSs in the results. For optimizing the RS_{th} for each path length, we take the average of the risk scores that provide the highest benefit in each of the 10 models. Then the average RS is used as the RS_{th} for predicting the riskiness of the actual test bags. The test results are analyzed from two different perspectives. In scenario1 (SC1), the predicted mishandled bags are not removed from the system unless they automatically disappear when they are really mishandled. It can show the actual benefit at different path lengths. In scenario2 (SC2), at different path lengths, the predicted mishandled bags are removed from the system such that they cannot be seen in the subsequent locations in their path. It can show the total benefit if bags are saved whenever it is detected as mishandled. The selected RS_{th}s, benefits, confusion matrices, and how early the bags are saved before the flight (SBF) for the bags with preplanned path P2 are reported in Table V. As there are 286 mishandled bags, the total cost without using the ORP will be \$27456. The benefits with SC1 shows that handling bags only at L_4 can save 82% of the total mishandling cost, whereas, in SC2 $\frac{168+0+1464+21141+0}{27456} \times 100 = 83\%$ of the total cost can be saved. It also shows that 99% of the mishandled bags are predicted within Len4 and at least 25 minutes before the flights.

TABLE V: Results based on the selected RS_{th}

Pathlen	RS_{th}	SBF	Bnft-SC1	Bnft-SC2	pred->	P-SC1	N-SC1	P-SC2	N-SC2
Len1	0.998	29.58	168	168	Act-P	3	283	3	283
					Act-N	5	14709	5	14709
Len2	0.94	28.23	66	0	Act-P	1	285	0	283
					Act-N	1	14713	0	14709
Len3	0.28	28.7	1647	1464	Act-P	22	264	19	264
					Act-N	9	14705	5	14704
Len4	0.479	25.3	22518	21141	Act-P	278	3	261	3
					Act-N	0	14714	0	14704
Len5	0.498	10.3	4374	0	Act-P	54	11	0	3
					Act-N	0	14714	0	14704

Scalability. We use DS_4 for building PFGs and APFGs for showing the scalability regarding their construction time and memory use. We use a set of SQL queries with some DDL and DML operations for building PFGs and APFGs and they are stored in the database tables. Before prediction, a C# program loads the PFG and APFG into main memory. The full PFG and APFG construction and loading times are reported in Fig. 8a. In the case of SQL query times, we clear the cache

after executing each operation. In all cases, we run the queries and code 3 times and report the rounded average time. In both cases, the results show that the construction time increases almost linearly with the number of bags. We also report the memory use of LDPHs and ALDPHs for the different numbers of bags. It shows that the size grows linearly with the number of bags. It also shows that the total size of ALDPHs is on average 84% higher than the total size of LDPHs. However, the total size of the ALDPHs is very small, only 194 KB for 70K bags. So, it is feasible even for a larger data set.



(a) Construction Time (b) Memory Usage

Fig. 8: PFG and APFG construction times and memory usage.

VII. Related Work

Related work falls into two main categories. One is to pre-process raw indoor tracking data and another one is to perform data mining on such tracking data. The RFID data management challenges and solutions are discussed in [6], [15]. Data warehousing, mining, and workflow analysis are proposed for RFID-based item tracking in the supply chain systems in [8], [9]. The authors convert the raw RFID records into cleansed record containing the first and last reading times of an object under the readers activation range. In the present paper, we use stay records [2], as it can capture the total stay duration between locations. Graph-based model for indoor tracking is discussed in [3], [11]. In the present paper, we extend graph models for capturing the object flows with new probabilistic concepts such as LDP, ALDP, and histograms.

Data mining is performed on the tracking data for finding frequent spatio-temporal sequential patterns [5], [10], typical movements of objects in indoor space [14], frequent trajectory patterns for activity monitoring [12], and frequent walk in RFID-equipped warehouse [4]. Interesting spatio-temporal rule mining applications, techniques and issues are discussed in [7]. The present paper introduces a new perspective which is for risk prediction in indoor moving object. In [1], RFID baggage tracking data are analyzed for mining risk factors in the offline scenario. The present paper focuses on an online risk prediction scenario that require more fine grained features such as object transitions at the reader level and duration for each of the transitions. Further, this paper is more general as the used features are common in many symbolic indoor and mixed indoor-outdoor tracking applications.

VIII. Conclusion and Future Work

We proposed detailed steps and probabilistic models for predicting the risk of online indoor moving objects. We converted the historical raw tracking records into stay records and used them for constructing the probabilistic flow graphs called PFG and APFG. The graphs capture the probabilistic information about the transition times by using histograms called least duration probability histogram LDPH and aggregated LDPH (ALDPH). The flow graphs are used for obtaining risk score of an online indoor moving object and for predicting risks. A comprehensive experiment with synthetic and real data showed that the proposed risk prediction method can differentiate risky objects from the correctly handled objects very accurately when the objects approach the bottleneck locations on their paths. We also proposed a cost model for object mishandling and the experiments showed that using APFG with the proposed normalization can significantly save the operation cost. The result also showed that the risky objects are predicted early enough such that they can be saved from being mishandled.

In future work, the proposed techniques can be expanded to more general scenarios such as mixed indoor-outdoor object tracking. Further, predicting risks for the objects in nondeterministic scenarios, where the paths of the objects are unknown in advance, can be another future direction.

Acknowledgment

This work is supported by the BagTrack project funded by the Danish National Advanced Technology Foundation under grant no. 010-2011-1.

References

[1] T. Ahmed, T. Calders, and T. B. Pedersen. Mining risk factors in RFID baggage tracking data. In *MDM (1)*, pages 235–242, 2015.

[2] T. Ahmed, T. B. Pedersen, and H. Lu. A data warehouse solution for analyzing RFID-based baggage tracking data. In *MDM (1)*, pages 283–292, 2013.

[3] T. Ahmed, T. B. Pedersen, and H. Lu. Finding dense locations in indoor tracking data. In *MDM (1)*, pages 189–194, 2014.

[4] Z. Berenyi and H. Charaf. Utilizing tracking data in RFID-equipped warehouses. In *ICC*, pages 169–173, May 2008.

[5] H. Cao, N. Mamoulis, and D. W. Cheung. Mining frequent spatio-temporal sequential patterns. In *ICDM*, pages 82–89, 2005.

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

[7] G. Gidófalvi and T. B. Pedersen. Spatio-temporal rule mining: Issues and techniques. In *DaWaK*, pages 275–284, 2005.

[8] H. Gonzalez, J. Han, and X. Li. Flowcube: Constructuing RFID flowcubes for multi-dimensional analysis of commodity flows. In *VLDB*, pages 834–845, 2006.

[9] H. Gonzalez, J. Han, and X. Li. Mining compressed commodity workflows from massive RFID data sets. In *CIKM*, pages 162–171, 2006.

[10] Y. Huang, L. Zhang, and P. Zhang. A framework for mining sequential patterns from spatio-temporal event data sets. *TKDE*, 20(4):433–448, 2008.

[11] C. S. Jensen, H. Lu, and B. Yang. Graph model based indoor tracking. In *MDM*, pages 122–131, 2009.

[12] Y. Liu, Y. Zhao, L. Chen, J. Pei, and J. Han. Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays. *TPDS*, 23(11):2138–2149, 2012.

[13] H. Lu, C. Guo, B. Yang, and C. S. Jensen. Finding frequently visited indoor POIs using symbolic indoor tracking data. In *EDBT*, pages 449–460, 2016.

[14] L. Radaelli, D. Sabonis, H. Lu, and C. S. Jensen. Identifying typical movements among indoor objects - concepts and empirical study. In *MDM (1)*, pages 197–206, 2013.

[15] F. Wang and P. Liu. Temporal management of RFID data. In *VLDB*, pages 1128–1139, 2005.