Risk Detection and Prediction from Indoor Tracking Data

Tanvir Ahmed[1], Toon Calders[2,3], Hua Lu[4], Torben Bach Pedersen[4]
[1]RadioAnalyzer ApS, Denmark
[2]University of Antwerp, [3]Université Libre de Bruxelles, Belgium
[4]Aalborg University, Denmark

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 discuss a risk detection and a risk prediction method for such kinds of indoor moving objects. We propose a data mining methodology for detecting risk factors from RFID baggage tracking data. The factors should identify potential issues in the baggage management. The paper presents the essential steps for pre-processing the unprocessed raw tracking data and discusses how to deal with the class imbalance problem present in the data set. Next, we propose an online risk prediction system for time constrained indoor moving objects, e.g., baggage in an airport. The target is to predict the risk of an object in real-time during its operation so that it can be saved before being mishandled. We build a probabilistic flow graph that captures object flow and transition times using least duration probability histograms, which in turn is used to obtain a risk score of an online object in risk prediction.

1 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., airports, office buildings, and shopping malls), airport baggage tracking, item movement tracking in supply chains, and package tracking in logistics systems. The research work presented in this paper in particular attempts to solve the airport baggage mishandling problem as the aviation industry suffers from enormous loss due to baggage mishandling. A recent report[1] shows that in 2013, 3.13 billion passengers traveled by airlines and among them around 21 million passengers and 21.8 million bags were affected by baggage mishandling that costs 2.09 billion USD to the airline industry. Common baggage mishandlings are: left behind at the origin or connecting airport (i.e., failed to catch the intended flight), bag loss, wrong bag destination, etc.

In airport baggage management a bag has to go through different steps in each airport from origin to final destination. Suppose that Nadia needs to travel from Aalborg Airport (AAL) to Brussels Airport (BRU) via Copenhagen Airport (CPH). First, Nadia has to check-in and hand over her bag to the check-in desk staff at AAL. Then the staff puts the bag on the conveyor belt for the automatic baggage sortation system. After passing all the steps inside AAL, the bag is loaded into the aircraft using the belt loader for the targeted flight. Upon arrival at CPH it is shifted to the transfer system. After all the required stages at CPH, the bag is loaded into the aircraft for BRU. After arriving at BRU, Nadia collects the bag from the arrival belt. During this journey,

[1]SITA Baggage Report 2014 `www.sita.aero/content/baggage-report-2014`

the bag has to go through up to 11 stages and there can be as many baggage handlers handling the bag at the different stages. Fig. 1 visualizes this airport baggage tracking scenario. The upper part of the figure shows the top level path of a bag that travels from AAL to BRU via CPH. 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.

At check-in, an RFID tag is attached to the bag. The memory inside the tag stores bag information including the bag identifier, flights, route legs, date of departure, etc. While passing different stages, whenever a bag enters into a reader's activation range, it is continuously detected by the reader with a sampling rate which generates raw reading records of the form: $\langle BagID,\ Location,\ Time,\ \{Info\}\rangle$, meaning that a reader at location *Location* detects a bag with ID *BagID* at timestamp *Time* and the tag stores the information *Info*. Considering only location and time related information, some examples of raw reading records are shown in Fig. 2a. In the table, *RID* represents the reading identifier. The massive baggage tracking data can be very useful for analyzing and finding interesting patterns. Combining the tracking data with other dimensions like route information, flights and punctuality, day hours, week day, transit duration, etc., can reveal risk factors that are responsible for baggage mishandling. As seen, a bag can have several readings at the same location and due to the rawness, it is also difficult to do further analysis. To overcome these problems and prepare the data for further analysis, we convert the raw reading records into stay records [3], described below.

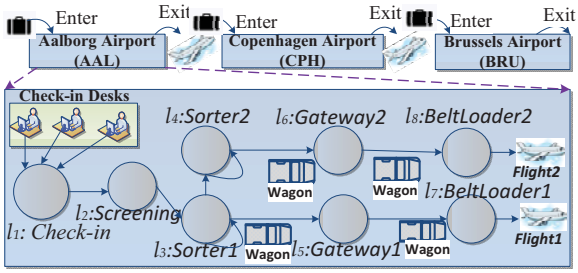

Figure 1: Baggage tracking in airport [2]

(a) Raw Reading Table

RID	BagID	Location	Time
R1	B1	AAL.Chkin1	1
R2	B1	AAL.Screen	4
R3	B1	AAL.Screen	5
R4	B1	AAL.sorter1	8
R5	B1	AAL.sorter1	9
R6	B1	AAL.Gate2	15
R7	B1	AAL.BltLd1	21
R8	B1	CPH.Trans1	70
R9	B1	CPH.Tran1	72
R10	B1	CPH.Sorter2	80
R11	B1	CPH.Sorter2	90
R12	B1	CPH.Sorter2	100

(b) Stay Records

Rec ID	Bag ID	FromLoc	ToLoc	t_{start}	t_{end}	Dur.
Rec1	B1	AAL.Chkin1	AAL.Screen	1	4	3
Rec2	B1	AAL.Screen	AAL.sorter1	4	8	4
Rec3	B1	AAL.sorter1	AAL.Gate2	8	15	7
Rec4	B1	AAL.Gate2	AAL.BeltLd1	15	21	6
Rec5	B1	AAL.BltLd1	CPH.Tran1	21	70	49
Rec6	B1	CPH.Trans1	CPH.Sorter2	70	80	10
Rec7	B1	CPH.Sorter2	CPH.Sorter2	80	100	20

(c) FlightLeg Records

Bag id	From Airpt.	To Airpt.	Is Tran.	Weekday	Flight Time	DurBef. Flight	IsLongStayFound	Delay InArr.	TotBag ThatHr	Status
B1	AAL	CPH	0	Monday	9-10	25	0	NULL	86	OK
B1	CPH	ARN	1	Monday	10-11	30	1	-5	70	Mishan.

Figure 2: Raw reading and relevant records [1]

Stay Records A stay record is of the form: $StayRecord\langle BagID,\ FromLocation,\ ToLocation,\ t_{start},\ t_{end},\ Duration,\ \{StayInfo\}\rangle$ which represents that a bag with *BagID* first appeared at *FromLocation* at time t_{start} and then first appeared at the next location *ToLocation* at time t_{end}. It took *Duration* time to go from the reader at *FromLocation* to the reader at *ToLocation*. The $\{StayInfo\}$ represents a set of other dimensional information related to the bag and to the transition (e.g., bag status, next flight schedule, origin and destination airports, and other flight-related information). The stay records compress the huge data volume of raw readings and also enable to find abnormally long time spans between locations that may lead to baggage mishandling. Fig. 2b shows the stay records for the raw records of Fig. 2a. In the table, *RecID* represents the stay record identifier. In Fig. 2b, *Rec1* represents that bag *B1* had a transition from *AAL.Checkin-1* to *AAL.Screening* and it took 3 time units for the transition. The stay records introduce a very important feature which is the *duration* information.

This paper addresses the baggage mishandling problem in two phases. First, it analyzes the problem in an offline scenario where it presents a data mining methodology for automatic extraction of risk factors from RFID baggage tracking data. Second, it addresses the issue in an online scenario where it develops an online risk prediction system for predicting the risk of a bag in real-time so that it can be saved from being mishandled. For example, from the data, one might be interested to know, *what is the probability that a bag will be mishandled if it has 35 minutes transit time at Copenhagen Airport on Sunday morning*? A real-time analysis request can be: *notify the baggage manager whenever a bag becomes risky during its processing time at Aalborg Airport*.

The rest of this paper is organized as follows. Sections 2 and 3 describe the mining risk factors from offline baggage tracking data and the online risk prediction system, respectively. Section 4 concludes and points to future work.

2 Risk Detection

In this section, we will discuss a methodology for building the best predictive model which will give us a set of rules or patterns that are highly correlated to the mishandled bags.

FlightLeg Records For finding risk factors, we are interested to find the baggage management performance at airport level rather than at the reader level. We take the duration feature including some other dimensions from the stay records and create a table called *FlightLeg Records*. For each flight of a bag we have one instance in the *FlightLegRecords* table. It captures some important features extracted from the *Stay Records* like {*IsTransit, DurationBeforeFlight, IsLongerStayFound, TotalBagInThatHour, Status*}. The attributes are discussed below.

(i) *FromAirport*: Departure airport of the corresponding flight. (ii) *ToAirport*: Destination airport of the corresponding flight. (iii) *IsTransit*: A Boolean value representing whether the record belongs to a transit bag or not. (iv) *Weekday*: Weekday of the corresponding flight. (v) *FlightTimeHour*: Departure hour of the corresponding flight. (vi) *DurationBeforeFlight*: Available time (in minutes) for the bag to catch the flight. For a non-transit record, it is calculated from the first reading time of the bag at check-in and the actual departure time of the flight. Conversely, for a transit record, it is calculated from the actual flight arrival time at the *FromAirport* and actual departure time of the next flight to the *ToAirport*. (vii) *IsLongerStayFound*: If any stay duration between readers at the *FromAirport* is longer than expected then it is *true*, otherwise it is *false*. Its value is determined by comparing the movements of baggage between readers at *FromAirport*. For each distinct transition between a pair of locations in the *Stay Records*, the bags that had followed the top 5% longest durations are considered as *longer than expected*. (viii) *DelayInArrival*: Delay in arrival (in minutes) of the arrival flight for the transit bag. Its value is calculated from the actual and scheduled arrival times of the flight in the transit airport (i.e., *FromAirport* is a transit airport). For the non-transit records *DelayInArrival* is *NULL*. (ix) *TotalBagInThatHour*: Number of bags read during the departure hour of the flight at *FromAirport*. (x) *Status*: Status of the bag i.e., 'OK' or 'Mishandled'. The status of the bag indicates whether the bag was mishandled or not in the *FromAirport*. The status of a bag is extracted from the reading records of the bag at the readers at *FromAirport*, flight timing, route information, etc. If a bag has any reading in the *FromAirport* after the corresponding flight departure time, then the bag is considered as left behind. Conversely, if a bag has any reading from an airport which is not in its planned route, then it is considered as wrong destination.

Fig. 2c shows an example content of *FlightLegRecords*. We use the *FlightLegRecords* for further analysis.

2.1 Solution Steps for Risk Detection

For detecting risks, we take the help of a classification algorithm. We use the *Status* attribute of the *FlightLegRecords* as the class column. However, baggage tracking data are highly imbalanced as the rate of mishandled bags is quite small ($<1\%$) compared to the correctly handled bags [1]. This imbalance presents difficulties to most data mining techniques. As a result, this *imbalance problem* should be handled wisely to overcome poor quality results otherwise produced by the classifier. The solution steps are given in Fig. 3. At first, the raw records are converted into the *FlightLegRecords*. The other steps are discussed next.

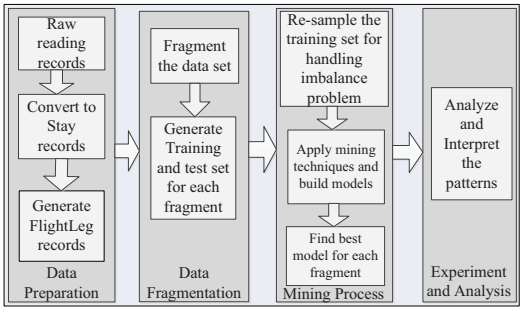

Figure 3: Outline of the steps

2.2 Data Fragmentation

Fragments The baggage management problem varies based on different important factors like whether the bag is in the transit airport or not, the duration of the transit, etc. Based on that, we have divided the data set into 5 fragments and applied data mining algorithms on each of the fragments for finding patterns specific to each fragment. Fig. 4 shows the different fragments of our data set, and the numbers inside the square bracket show the number of records and mishandling rate for the corresponding fragment in our experimental data [1].

Training and test set For each of the discussed fragments, we have one partition for training (P1), and another for testing (P2). In our experiments, all the records on or before a certain flight date are included in the training set and the rest is included in the test set. We did not rely on a standard cross-validation approach because bags that were on the same plane are more likely to have similar properties, as well as a similar class label. Therefore, spreading bags of the same flight over both the training and test set may cause a biased estimation of the performance due to overfitting. By dividing the data based on date, we can guarantee that the training set and the test set are independent, and we get an unbiased estimate of the performance of the mined models.

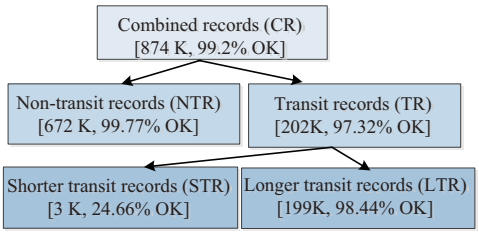

Figure 4: Fragments of the data set

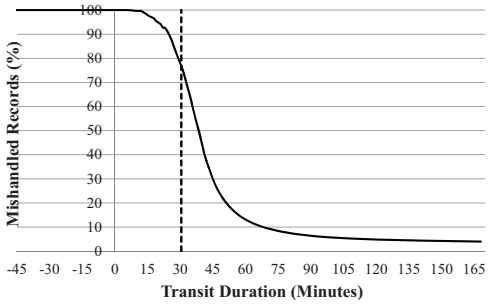

Figure 5: Mishandling vs. transit duration

2.3 Mining Process

Handling imbalance data. To remedy the imbalance problems, we use 2 different kinds of re-sampling for the training data(P1): ***Undersampling*** **(US)**: In this technique, a subset of *P1* is created by randomly deleting *OK* records until we reach equal number of records with class *OK* and class *MH*. ***Oversampling*** **(OS)**: In this technique, a superset of *P1* is created by copying some instances or generating new instances of *MH* records until we obtain an equal number of records for class *OK* and *MH*. We use Synthetic Minority Over-sampling Technique (SMOTE) [4] for getting *OS* data.

Mining Techniques We apply *Decision Tree (DT), Naive Bayes classifier (NB), KNN classifier (KNN), Linear regression (LIR), Logistics regression (LOR)*, and *Support vector machine (SVM)* on the training set *P1* of the combined records *CR* with the sampling strategies discussed above. We also do the same directly to P1 without re-sampling *(WS)*. Then we use different types of measures (discussed in the next paragraph) for finding the classification and sampling techniques that provide the best model for our data set. Subsequently, the chosen techniques are used for generating models for the remaining fragments. Before applying KNN, linear regression, logistics regression, and SVM, the structure of the input data table is changed as these algorithms do not work with categorical attributes. In these cases, we convert each value of a categorical attribute into a separate column and put Boolean 0 or 1 accordingly. An example of such conversion for Fig. 2c is shown in Table 1. For linear regression and logistic regression, the attributes with continuous values are normalized into [0,1]. Moreover, in our data set all the *FromAirport*s are within the *Schengen territory* (http://en.wikipedia.org/wiki/Schengen_Area). Unlike *FromAirport*s we have too many values in the *ToAirport* column which creates many branches in the decision tree and for other classification algorithms this column become useless. To make the *ToAirport* column useful and make the learned pattern interesting, we categorized the *ToAirport*s into three types: *Domestic, Schengen*, and *Others*.

Table 1: String values into columns of Fig. 2c

AAL	CPH	IsTransit	Monday	9-10	10-11	....
1	0	0	1	1	0	...
0	1	1	1	0	1	...

Finding the best model Typically the performance of a classifier is evaluated by its predictive accuracy. However, for an imbalanced data set the accuracy is not an appropriate measure, e.g., in our case an accuracy of 99% does not make sense, as it may misclassify all the examples as *OK (negative)* regardless of whether a record belongs to *Mishandled (positive)* or not. In our scenario, misclassifying a *Mishandled* bag as *OK* is more severe than misclassifying an *OK* bag as *Mishandled*. As such we are specifically interested in algorithms with a high recall on the *Mishandled* bags rather than in merely optimizing the accuracy of the classifier. We use the AUC of the ROC curve as the main measure for choosing the model that provides the best ranking. We also use precision-recall curves for finding which threshold provides higher precision for a good amount of recall. We perform a comprehensive experiment which can be found in [1]. Some results are presented in Section 4.

3 Online Risk Prediction (ORP)

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. For example, the model should be able to predict as early as possible whether a bag going through the baggage handling stages is at risk of being delayed.

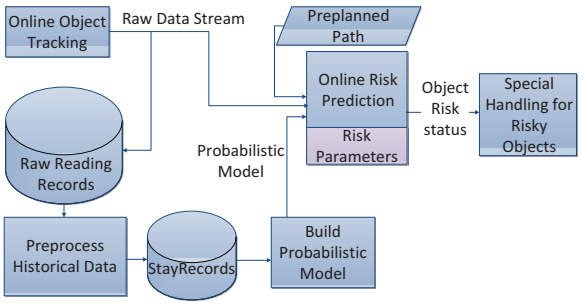

Figure 6: Overview of the ORP system [2]

3.1 Solution

The overall outline of the ORP system is shown in Fig. 6. The online tracking data stream is passed into two modules. One stores the data offline for future analysis and model building purpose and another uses it during the ORP process. The offline/historical records are converted into *StayRecords*, which are used for building the probabilistic model. The model, the raw data stream, and the preplanned paths 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.

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

Definition 1: ***Least Duration Probability (LDP).*** An LDP for a movement from l_i to l_j with threshold duration $d_k \in D_{i,j}$ is defined as

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

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

Definition 2: ***Least Duration Probability Histogram (LDPH).*** An 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.

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	1000	1000	1
	16		700	0.7
	20		300	0.3
	28		20	0.02
$l_2 \rightarrow l_3$	8	1000	1000	1
	10		580	0.58
	17		50	0.05
$l_3 \rightarrow l_5$	60	470	470	1
	70		250	0.53
	98		50	0.11
$l_5 \rightarrow l_7$	40	460	460	1
	50		250	0.54

Table 2: Transition summary (C stands for *Count*)

Table 2 shows an example summary of transitions for the path l_1 to l_7 generated from stay records. Fig 7 shows the LDPH for transition l_1 to l_2 shown in Table 2. In Fig. 7, LDP=0.7 represents that the probability of transition from l_1 to l_2 with a duration ≥ 16 is 0.7. It also shows that the LDP for duration 28 is very low (0.02).

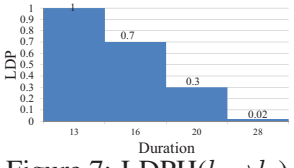

Figure 7: LDPH($l_1 \rightarrow l_2$)

Probabilistic Flow Graph (PFG). We build a probabilistic flow graph (PFG) from the data set, where each location is represented by a node and transitions between locations are represented by edges. The edges are labeled by their corresponding LDPHs. Fig. 8 shows the PFG constructed from the transition summary shown in Table 2. One LDPH of Fig. 8 is shown in Fig 7. The data for the rest of the LDPHs are available in the $LDP(tr, d_k^{\geqslant})$ column of Table 2.

3.2 Online Risk Prediction Steps

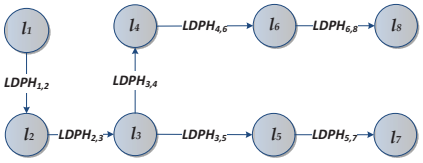

Figure 8: PFG

We consider a scenario, where the path of a given online object is predefined, e.g., in 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 an unusually longer duration for a transition, can also become risky. Moreover, the risk of an

object being delayed not only depends on the duration at a single location, but also at the subsequent locations. For example, an object might take a long duration from 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. To capture this, we aggregate the paths in the PFG into an *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 we have $i<n<=p$ (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. (2). In Eq. (2), $Count(S, d^{\geqslant})$ 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 3 shows the ALDPs and data for ALDPHs for the path from l_1 to l_7 in our example scenario.

$$ALDP(S, d^{\geqslant}) = \frac{Count(S, d^{\geqslant})}{Count(S)} \quad (2)$$

For risk prediction, we use an ALDP threshold *$ALDP_{th}(s_i)$* for each path sequence s_i for each of the preplanned paths. The $ALDP_{th}$ is converted into a *risk score threshold* RS_{th}, where $RS_{th}(s_i)$=*1-$ALDP_{th}(s_i)$*. After each transition of an object o, we get the total duration spent d by o to travel the path path sequence s_i and then we extract the *ALDP(s_i, d)* from the corresponding ALDPH. If d is not directly available in the ALDPH, we use linear interpolation to obtain the corresponding ALDP. After getting ALDP, we calculate the *risk score* of the object by *RS=1-ALDP*. If *RS*$\geq$*RS_{th}*, then o is marked as in risk. Also note that for the first transition, the values of ALDP and LDP are the same.

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

Table 3: Path summary

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 3. In the described scenario, o has to complete a transition to determine whether the object is at risk or not. To improve this scenario, we take the help of *$ALDP_{th}(s_i)$* to find the maximum acceptable stay duration ($Dur_{max}(s_i)$) of o to complete the path sequence s_i. Based on Dur_{max} a *time trigger (TT)* is set for o after each transition, which will trigger an alarm if o is not being read by the next planned location within the timestamp. The time trigger of o is set by, $TT[o]$ = t_{start}+Dur_{max}, where t_{start} is the timestamp when o was first tracked in the system and Dur_{max} is the maximum allowable duration for an object to complete its path sequence up to the next planned location. The overall processing steps of the ORP are shown in Fig. 9.

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 on the arrows represent the duration taken for the transitions. Let us consider *$ALDP_{th}$* for any path sequence is 0.2, thus RS_{th} = 1 - 0.2 = 0.8. Also, $Dur_{max}(l_1, l_2)$ by linear interpolation=$\lceil 20 + (0.3 - 0.2)/(0.3 - 0.02) \times (28 - 20) \rceil$=23. Consider that the object followed its preplanned path. For the first transition, LDP (l_1,l_2,$17^{\geqslant}$) by linear interpolation = 0.7-(0.7-0.3)/(20-16)$\times$(17-16) = 0.6, thus RS=1-0.6=0.4. As 0.4$\ngeq$0.8, o is not risky at this stage. Also, in terms of Dur_{max}, o is safe in this step as 17$\ngtr$23. For the second transition (i.e., up to l_3), the ALDP is 0.05 (as the total duration=17+17=34), thus RS = 0.95. As As 0.95$\geq$0.8, o is at risk at that point. After the next transition (i.e., up to l_5), the ALDP is 0.41, thus RS = 0.59$\ngeq$$RS_{th}$=0.8 (as total duration=17+17+60=94). The new score shows that the object recovered from its risky state as it was processed quickly between l_3 and l_5. 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 could result in a dense location or traffic jam. However, there are many applications where an object has to reach a particular location by a given timestamp. For example, in an airport a bag has to be loaded in the aircraft before the scheduled flight departure. So, the duration before 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, and vice versa. 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.

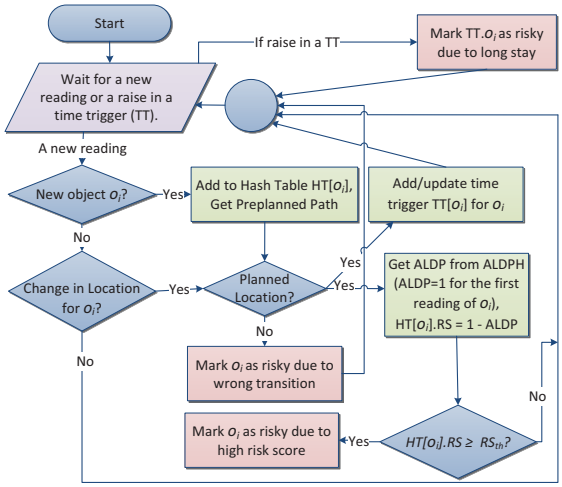

Figure 9: Online risk prediction steps

Let t_{enter} be the first time an object o was detected in the system and t_{final} be the maximum timestamp when o should reach its final reading point. 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. 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. (3), 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.

$$d_n(o) = \textit{Offset} + \sum_{i=1}^{k} dur_i, \textit{where offset} = (d_e - d_a) \tag{3}$$

For example, consider an object o_1 following its preplanned path: $l_1 \xrightarrow{17} l_2 \xrightarrow{17} l_3 \xrightarrow{94} l_5 \xrightarrow{50} l_7$. o_1 has a total of 200 seconds to reach l_7 from l_1. So, d_a=200 sec. From Table 3, d_e=134 (as ALDP for 134 is 0.51). Hence, *offset*=134-200 = -66. Now, for the first transition $l_1 \xrightarrow{17} l_2$, d_n=-66+17=-49. Therefore, from Table 2, the value of ALDP or LDP for the transition is 1. So, 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 if 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 for o_i. Besides, a higher value of *offset* will reduce the value of Dur_{maxN} for adjusting the riskiness of o. Finally, the corresponding time trigger is computed by $TT[o]_N$= 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 the positive class for classification. A prediction system giving too many false positives (FP) or false negatives (FN) 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, where the costs for the different kinds of errors are used for finding the threshold that maximizes the benefit. For example, 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 of human resources for the mistake. However, if an FN occurs, there will be a significant cost for delivery, insurance, and other operating costs. So, in the baggage tracking scenario, the cost for an FN is much larger as compared to that for an FP. During model building and testing, we use Eq. (4) for obtaining the total benefit for each of the generated thresholds and use the threshold that provides the maximum benefit. In Eq. (4), x is the 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. (4) can provide an idea how much money can be saved by using the ORP system.

$$\textit{Benefit}(x, y) = x \times \#\text{P} - (x \times \#\text{FN} + y \times (\#\text{TP} + \#\text{FP})) \tag{4}$$

We have reported a comprehensive experimental study in [2]. The 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. The risky objects are predicted early enough such that they can be saved before being mishandled.

4 Conclusion and Future Work

In this paper, we presented two approaches for analyzing the risk of indoor moving objects, primarily focusing on an RFID baggage tracking scenario. The first approach focused on the offline scenario where we proposed a data mining methodology for finding risk factors from the class imbalanced baggage tracking data. Our experiments [1] showed that the decision tree with under sampling provides the best model in our data set. The extracted patterns show that duration before flight is a critical factor and a bag is considered to be a high risk if it has less than 54 minutes in the transit airport. Moreover, departure airport, a longer stay at a location, and the busyness of the airport are also important factors. The second approach of this paper focused on the online scenario where we proposed an online risk prediction system for predicting risk of a bag in real-time so that it can be saved from being mishandled. Our experiments [2] showed that the proposed method can predict risk of a bag with more than 99% accuracy when it reaches in its bottleneck location (e.g., sorter). The proposed approach is also useful for other time critical multi-site based indoor tracking scenarios.

Several directions for future work exist. First, a more thorough study of the root causes for baggage mishandling, which are non-trivial, given the low probability of mishandled events. Second, the proposed online risk prediction technique can be extended to more general scenarios such as mixed indoor-outdoor object tracking. Third, it is useful to predict risks for the objects in nondeterministic scenarios, where the pre-planned paths are not available.

Acknowledgments

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*, pages 235–242, 2015.

[2] T. Ahmed, T. B. Pedersen, T. Calders, and H. Lu. Online risk prediction for indoor moving objects. In *MDM*, pages 102–111, 2016.

[3] 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.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.