Real-time Probabilistic Data Association over Streams*

Mert Akdere†
Google, Inc.
Mountain View, CA, USA
makdere@gmail.com

Jeong-Hyon Hwang
University at Albany, SUNY
Albany, NY, USA
jhh@cs.albany.edu

Uğur Çetintemel
Brown University
Providence, RI, USA
ugur@cs.brown.edu

ABSTRACT

The Probabilistic Data Association (PDA) problem involves identifying correspondences between items over data sequences on the basis of similarity functions. PDA has long been a topic of interest in many application areas such as real-time tracking and surveillance. Despite its significance, however, it has largely been ignored by the event-processing community. Our work rectifies this situation by studying PDA in the context of continuous event processing.

Specifically, we formulate PDA as a continuous probabilistic ranking problem with constraints and efficiently solve it using fast constraint resolution. Our solutions are built on a top-k approximation to the problem guided by resource-aware optimization techniques that adaptively utilize the available resources to produce real-time results. User-defined data association constraints are used to restrict the solution space and quickly eliminate inconsistent solution candidates. We also derive the runtime complexity of our solutions and experimentally evaluate these solutions using a prime PDA application: real-time tracking of moving objects within a camera network. Our evaluation results demonstrate the superiority of our solutions over traditional constraint programming formulations.

Categories and Subject Descriptors

H.2 [**Database Management**]: Systems

Keywords

data association; data streams; probabilistic databases; ranking; constraint resolution

*This work has been supported by the National Science Foundation under CNS-0721703 and CAREER award IIS-1149372.

†Work done while at Brown University.

DEBS'13, June 29–July 3, 2013, Arlington, Texas, USA.

1. INTRODUCTION

The data association problem refers to identifying correspondences between items over data sequences [19, 20]. A classical application area for data association is *tracking* [20], where the task is to map a given set of measurements to a set of tracks (or objects). Probabilistic data association (PDA) is a type of data association technique that uses similarity functions to create data association hypotheses together with association scores [2, 3]. In tracking applications, each hypothesis would represent a mapping of measurements to tracks.

PDA is an important task with many applications in diverse domains. As mentioned, a major target of PDA is the area of tracking and surveillance applications such as visual tracking where the task is to track people (or objects) in video streams [2], ocean surveillance (tracking of ships and submarines), and air defense and traffic control systems [3]. Document classification and topic intensity tracking is also an application of PDA where each document (e.g., e-mail) is classified into topics and the evolution of topic trends are of interest [1]. Yet another interesting application is found in the domain of law enforcement where the task is to associate crime incidents committed by the same individuals or group of individuals based on suspect traits and crime similarities [27].

PDA is often cited as one of the biggest challenges in tracking applications [19] and has long been a topic of interest. However, despite its importance and wide applicability in a variety of applications, it has largely been ignored by the event-processing community. In this paper, we explore the PDA problem from the perspective of probabilistic query processing [16]. Specifically, we consider the PDA problem as a constrained probabilistic ranking problem with possible worlds semantics [6, 7].

In the possible worlds semantics, relations are defined to be uncertain: an uncertain relation represents a set of possible relation instances. In most cases, this is achieved by adding an *existence probability / confidence* field to each relation. This added field represents the membership probability of a tuple in a relation. Based on this definition of uncertain relations, a probabilistic database is defined as a set of uncertain relations and can be viewed as a probability distribution over database instances. There is a close relationship between the data association problem and the possible worlds semantics: the PDA problem can be defined over an uncertain relation consisting of tuples describing data associations together with association scores. In this formulation,

each possible world instance represented by the uncertain relation would be a separate data association hypothesis.

We apply the above formulation to represent the PDA problem. Assuming a continuous influx of tuples that represent data associations and the probabilities of these associations, we consider the *PDA query* whose goal is to find the maximum probability possible world among those that can be defined according to the input tuples. We also present two efficient query execution methods that both emphasize efficient partial-exploration of the underlying possible worlds in the presence of association constraints. The first one is based on *Constraint Programming* (CP) [23] and leverages existing powerful constraint solvers [22] for efficiency and generality. The second approach is a form of *Ranked Search* that efficiently explores only the high-probability possible worlds, selectively pruning them with constraint elimination methods. The latter has the benefit of solving the PDA problem with low computational overhead.

Since exploring all possible worlds is intractable, our solutions use a *top-k approximation* approach which maintains only the k possible worlds with the highest probabilities. This approach has the benefit of limiting the constraint space and avoiding the decline of data association speed. In contrast to previous solutions that relied on pre-defined, fixed k values [2, 3], our solutions adjust the value of k to effectively balance processing overhead and association accuracy. These solutions tackle the challenges of detecting possible worlds that make limited contributions to association accuracy, as well as improving association accuracy while negligibly increasing computational overhead.

We illustrate these solutions using a running example, *visual tracking*, throughout the paper. While we heavily use visual tracking to explain our techniques with concrete examples and to demonstrate their potential impact in a real-world application, our goal is not to specifically solve this tracking problem per se. In what we describe below, the only application-specific component is the "cooking" functionality that processes the raw data (i.e., images in our example application) to extract the data on which our techniques operate. As such, our techniques are quite general and readily applicable to other PDA applications.

We note that our solutions go beyond the existing results on probabilistic query processing research in two aspects. First, probabilistic databases have limited support for the specification and efficient execution of domain constraints, which play a central role in PDA. Second, continuous PDA is fundamentally a stream processing problem as it involves temporal sequences. Prior work studied mostly static data and one-time queries (e.g., [7, 14]). To effectively support real-time tracking of moving objects, our solution has a time threshold that bounds the time to be spent on processing incoming data and strives to produce most accurate results within this time limit.

The paper makes the following contributions:

(i) Our high-level contribution is the first use of continuous, probabilistic query processing techniques on an important class of applications that involve PDA and that have so far remained outside the scope of the event-processing community.

(ii) We integrate efficient constraint resolution methods into probabilistic query evaluation. We first describe how we can use a constraint solver for this purpose. This is important in that a class of constraints that are already efficiently supported by existing solvers can be immediately used. Next, we propose a general search-based method for the resolution of generic constraints and then present methods to integrate efficient constraint resolution techniques into the solution. We also discuss the computational complexity of these methods.

(iii) We describe an adaptive parameter tuning technique that can be used to trade off processing overhead with association accuracy. With additional optimizations, we show that processing times can be significantly reduced with only modest accuracy losses.

(iv) We use a real-world PDA application, a visual tracking application that we deployed locally, for experimentally evaluating our techniques. The results reveal that our approach outperforms a widely cited generic solution for PDA [3]. (Our claim concerns the solution of the generic PDA problem and not the heavily optimized, specialized PDA solutions that exist for the tracking application.)

In the rest of the paper, we first describe the PDA problem in Section 2. Then, we present our solutions to the PDA problem in Section 3. Constraints for PDA and resolving these constraints are described in Section 4. Section 5 presents our technique for balancing accuracy and execution time. An experimental evaluation of our techniques is given in Section 6. Related work is in Section 7 and we conclude the paper in Section 8.

2. THE PDA PROBLEM

In this section, we first discuss and formulate the PDA problem, using a generic PDA application context. Next, we discuss the complexity of the PDA problem and its relationship with the PDA constraints. Finally, we present the visual tracking application which is used as the running example throughout the paper.

2.1 Generic PDA Application Model

We consider a generic PDA application in terms of the following main components: (i) data items, (ii) high-level objects and (iii) similarity functions. *Data items* are the basic units between which we would like to establish associations (i.e., links). In general, each *object* represents a set of associated data items and is uniquely identified by an *object id*. However, the complete semantics of objects, their representation and management (i.e., object creation/deletion) are application dependent. In a tracking application, each object could represent a person, with its data items identifying the appearances of the person. Similarly, in a document classification system objects and items could represent a list of topics and documents, respectively. In addition, applications could either choose to perform their own object management (e.g., by defining a priori a set of objects, such as a fixed set of document topics) or allow the system to self-manage the objects. In system-based object management, new objects are automatically created for the received data items and object deletion is performed when no item is assigned to an object for a user-specified time interval. For now we assume that there exists a fixed set of objects. Finally, a *similarity function* computes a similarity

value between a given object and a data item. Then, within this general setup, the primary goal of a PDA application is to find the maximum score associations, represented via objects, between data items based on the results obtained from the similarity functions.

2.2 Problem Definition

First, we consider the PDA problem on a non-streaming (i.e., traditional) probabilistic database. Later, we generalize it to the streaming scenario. In the non-streaming case, the input to the system is an uncertain relation U (see Figure 1) consisting of tuples of the form $<$ *item id*, *object id*, *probability* $>$. Each tuple of the relation U represents the association of a data item and an object together with a probability. Tuple probabilities are derived from the computed similarity values by normalization.

	item id	object id	probability	TupleID
Mutual Exclusion Group I	1	1	.3	t1
	1	2	.7	t2
Mutual Exclusion Group II	2	1	.4	t3
	2	2	.6	t4

PW	Probability
{t1,t3}	.12
{t1,t4}	.18
{t2.t3}	.28
{t2,t4}	.42

Figure 1: An example uncertain relation, U, representing the associations between two data items and two objects given for the non-streaming PDA problem

Each tuple is also part of a mutual exclusion group (MEG) [10, 15]. A MEG is a set of tuples from which only one can exist in a database instance. A separate MEG is formed for each data item to represent the basic constraint that each item is to be associated exactly with a single object. Then, the PDA problem is equivalent to finding the maximum probability possible world, W, represented by the relation U. Observe that W consists of a single assignment for each data item and represents the set of assignments for which the overall probability is maximized. Data items are associated with each other if they are assigned to the same object. In Figure 1, all of the possible worlds and their probabilities are listed for the given uncertain relation. In this case, the possible world $\{t2, t4\}$, which assigns both items to object 2 is the maximum probability possible world.

Current probabilistic databases allow a tuple to be part of only a single mutual exclusion group [10, 15] and as far as we know not much research has been done on the interplay of constraints and probabilistic databases. In this scenario (i.e., when each tuple is part of a single MEG and there are no additional constraints), the problem is solvable in linear time (in the number of tuples) by selecting the tuple with the maximum probability for each MEG. However, in many PDA applications, there is a number of additional constraints restricting the space of valid data associations and the complexity of the PDA problem depends on the given set of constraints. We discuss the problem complexity in Section 2.3 and present example constraints in the context of the visual tracking application in Section 2.4. With additional constraints, the PDA problem becomes a constraint optimization problem which can be formalized as follows:
PDA Problem (non-streaming): Given an uncertain relation U representing the associations between n data items and m objects,

$$maximize \prod C[i][X_i]$$
$$s.t.\ X_i \in D_i, \text{constraint\_list}$$

where X_i is a variable that represents the i^{th} data item from U and has the domain $D_i = \{1, 2, \ldots, m\}$, and $C[i][j]$ is the probability of the assignment of the i^{th} data item to the j^{th} object.

The solution to the constraint problem is a list of assignments of the form $X_i = x_i$ for $1 \leq i \leq n$. The assignment $X_i = x_i$ assigns the i^{th} data item to the x_i^{th} object. Hence, the solution also encodes the maximum score possible world that satisfies the given list of constraints.

In our system, we support generic boolean constraints on the set of assignments and user-defined function-based constraints as part of the *constraint_list*. For instance, a simple example constraint is $X_1 \neq X_2$ which disallows assigning items 1 and 2 to the same object. We discuss the set of supported constraints, constraint resolution techniques and example constraints for the visual tracking application in Section 4.
PDA Problem (streaming): Now, we consider the same problem in the context of probabilistic data streams. The streaming PDA problem is mainly a continuous / iterative version of the non-streaming PDA problem. We assume that a high-level PDA application continuously receives new data items. Moreover, upon the receipt of each set of new items, the high-level application computes the similarity values between all pairs of new items and existing objects.

In the streaming probabilistic data scenario, the tuples of the uncertain relation U are modified to be of the form $<$ *time point*, *item id*, *object id*, *probability* $>$. The new attribute *time point* is used to represent the iteration number (i.e., the points at which new tuples are appended to the database). For instance, in a visual tracking application, the video frame number could be used as the time point attribute. In this way, all of the data items extracted from the same frame would be identified with the same time point value. Finally, we can now define the streaming PDA problem as the problem of computing the set of maximum score assignments at each point in time for all of the received data items across all points in time.

2.3 Complexity of the PDA Problem

In Section 2.2, we showed that the PDA problem can be expressed as a constraint optimization problem. In this section, we discuss the complexity of the PDA problem based on the constraint optimization formulation. Constraint optimization problems (hence, also the PDA problems) are equivalent to constraint satisfaction problems (CSP) [28] to which an additional set of "soft" constraints (i.e., preferential constraints which need not be satisfied) are added to represent the score functions. A CSP is defined as a triple (X, D, C), where X is a set of variables, D is a set of domains and C is a set of constraints and the goal is to find an assignment for all of the variables that satisfy the given set of constraints.

CSPs are in general NP-complete and tractable only for a few very restricted cases such as the well-known 2SAT case, where the variables are all binary-valued and all of the constraints are binary (i.e., defined over 2 variables) [28]. Generally, constraint solvers apply search-based methods, such as the branch-and-bound algorithms, backtracking algorithms and heuristic methods [29] to solve the CSPs. In our constraint-programming based solution to the PDA problem (Section 3.2.1), we employ a constraint solver to demonstrate this approach. The exact complexity of a PDA

problem mostly depends on the given set of constraints. Therefore, it is essential to use efficient constraint resolution techniques whenever applicable. For this purpose, in our second solution to the PDA problem, the Ranked Search method (Section 3.2.2), we allow the integrated use of specific constraint resolution techniques. This is discussed in more detail in Section 4.

2.4 Running Example: Visual Tracking Application

In this section, we describe the visual tracking application which is used as the running example in the rest of the paper. In the visual tracking application, we consider the tracking of people and image features (e.g., corner features [21]) extracted from video streams. In the case of tracking people, using similarity functions based on image processing techniques, we compute the similarity score of a blob and a person. A blob is defined as a connected region of foreground pixels in an image and a person is represented as a series of blobs and related information. Therefore, the task of tracking people (i.e., high-level objects) reduces to the act of associating blobs (i.e., data items) across images. This process is illustrated in Figure 2 where blobs are denoted with rectangles and blob associations are shown with arrows. Tracking features is a similar task but performed on image features instead of blobs. Further information on the detection of blobs and features, and the camera network setup used in collection of video streams for experimentation are provided in Section 6.

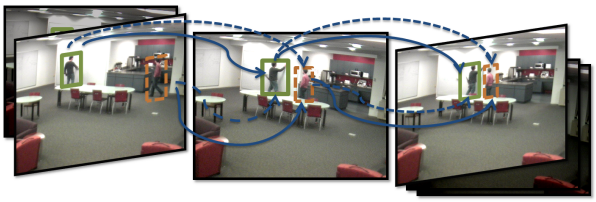


Figure 2: Illustration of tracking and data association using a video captured with our local camera network. Solid lines indicate the correct data associations.

We can represent the tracking application as shown in Figure 3. We identify each frame and person using a frame number (*Frame#*) and person id (*PersonID*). The blobs extracted from the same frame are assigned unique ids (*BlobID*). Each tuple describes the score (*MatchScore*) of associating a blob and a person. The association score is computed using similarity functions (discussed in Section 6). Also, we assigned each tuple with a unique *TupleID* for illustration purposes. Observe that, the *Frame#*, *BlobID*, *PersonID* and *MatchScore* fields correspond to the *time point*, *item id*, *object id* and *probability* fields discussed in Section 2.2, respectively. Within this setup, for each possible world derived from the given data stream, the probability of that possible world can be computed. For instance, in the possible world $\{t2, t3\}$, the first blob in frame 1 is person 2 and the second blob is person 1. The scores of tuples in the same mutual exclusion groups can generally be used to derive probabilities. In this example, we do not perform this step since it is not necessary for the PDA query. As mentioned before, the score of a possible world is defined to be the product of the scores of its assignments.

The data stream in Figure 3 is ordered according to the key (Frame#, BlobID, PersonID). This ordering occurs in-

Frame#	BlobID	PersonID	MatchScore	TupleID
1	1	1	.9	t1
1	1	2	.8	t2
1	2	1	.9	t3
1	2	2	.7	t4
2	1	1	.6	t5
...	...	...	...	...

PW[1]	Score
{t1,t3}	.81
{t1,t4}	.63
{t2,t3}	.72
{t2,t4}	.56

Mutual Exclusion Groups:
(t1,t2), (t3,t4)

Figure 3: Representation of a single-camera tracking application as a probabilistic data stream

herently as frames are received in-order from a camera and the processing follows the same order. A possible world for a given snapshot of the input stream at a frame number f, represents a complete data association for all of the received frames (from 1 to f). When a new frame, $f+1$, is received, a new possible world, including associations for frame $f+1$, is to be formed. Figure 3 represents a single-camera scenario. In the multiple-camera case, the input stream is augmented with a *CameraID* field, and the processing is done in terms of synchronized sets of frames.

In the above example, the mutual exclusion groups defined on Frame# and BlobID, are used to specify the underlying assumption that each blob is to be assigned to only a single person (e.g., $t1 \vee t2$). However, a number of other additional constraints could be specified as part of a tracking application. For instance, we could require a person (i) to be assigned at most one blob in a frame and (ii) not to be assigned blobs in cameras with non-overlapping views. Supporting such constraints efficiently is discussed in Section 4.

3. PDA QUERY EXECUTION

The PDA query is the database specification of the streaming PDA problem defined in Section 2.2. Hence, the goal of the PDA query is to continuously compute and output the most likely possible world (i.e., the data association hypothesis with the highest score) at each point in time. Each output possible world must represent a complete data association hypothesis covering all of the query points in time.

In the tracking application, the task is to track all of the people (or features) observable by the cameras. As discussed in Section 2.4, this task corresponds to a continuous data association problem on a probabilistic data stream (see Figure 3). In the context of the tracking application, we use the PDA query to perform the association between the people/features detected in the camera network. As mentioned before, we assume that all of the tuples with the same Frame# / time point, called a *frame set*, are appended to the input data stream. In the rest of the paper, we use the PDA query mainly within the tracking application framework to easily express our execution methods and related optimizations.

Finally, the PDA query supports a range of constraints which are commonly used in PDA applications. These include pairwise inequality constraints on association items, unique assignment constraints and other user-defined constraints; all supported through a single interface. Constraints and their enforcement are discussed in Section 4.

3.1 Top-k Approximation

A naive method to execute the PDA query would be to repeatedly enumerate all of the possible worlds from the

input stream and output the possible world with the maximum probability that satisfies the given set of constraints as the answer upon the arrival of each new frame set (or in a generic PDA application, upon the arrival of the tuples of a new point in time). However, the number of possible worlds is exponential in the number of association items, and thereby prohibits the naive enumeration approach. In addition, each possible world represents a complete data association hypothesis for all of the received frames. Therefore, the hypothesis space keeps growing exponentially as more frames (and consequently more data items) are received.

A more practical, but inexact (i.e., approximate) approach to executing the PDA query would be to incrementally produce the output hypothesis (i.e., possible world) at each frame set by extending the output hypothesis for the previous frame set. More specifically, at each frame set we would expand the previous output hypothesis with associations for the newly received data items. While this approach would be efficient, it would not be optimal to only build upon the single best hypothesis at each frame, because a lower ranking hypothesis may end up ranking higher as new "evidence" (i.e., frames) is observed. As the best hypothesis for a frame set may be significantly different from the previous best hypothesis, computing multiple hypotheses makes it more likely to find the best hypotheses in the next frame sets. In other words, keeping multiple hypotheses enables some degree of error recovery in the data associations of the earlier frames. Finally, another reason for computing multiple hypotheses is that the ranking results are based on uncertain information and therefore the distinction between the top ranked possible worlds is not absolute.

As such, the Multi-Hypothesis Tracking (MHT) model [2, 3, 4] has been the most widely adopted approach to real-time PDA. While multiple realizations of this model are possible, the common idea is to keep track of k-best hypotheses, where k is decided (often in an ad-hoc manner) subject to real-time processing constraints. We also adopt this popular and practical *top-k approximation* model as the basis of our solution; however, as we describe below, we extend and significantly improve upon the existing solutions. At each point in time, we incrementally compute the k-best hypotheses based on the most recent k-best hypotheses. Each hypothesis is formed from a combination of a previous hypothesis with a set of assignments for the most recent point in time.

3.2 Execution Methods

In this section, we discuss different top-k based execution methods for the PDA query. In each case, we assume that we are given a set of data associations corresponding to the most recent point in time, together with the previous k-best possible worlds to start with. Given this input, the expected output of the query is the new top-k possible worlds and their scores.

3.2.1 Constraint Programming

Our first solution is based on a constraint programming (CP) [23] formulation of the PDA problem. The main advantage of a CP-based approach is that it enables the use of existing powerful constraint solvers [22] to solve the PDA problem. In addition, a variety of constraint types, already efficiently supported by the constraint solvers, are immediately available for use. Given m objects (all of the objects belonging to a possible world) and n association items (blobs or features) received at this point in time, we define a CP for the PDA problem as follows:

CP for PDA problem:

$$maximize \prod C[i][X_i]$$

$$s.t.\ X_i \in D_i, \text{constraint\_list}$$

where $X_1, X_2, \ldots X_n$ are variables with domains $D_i = \{1, 2, \ldots m\} \cup \{m+i\}$ $(1 \leq i \leq n)$ and $C[i][j]$ is the score of assigning the i^{th} item to the j^{th} object.

This formulation is very similar to the constraint program given in Section 2.2 except for the variable domain definitions. In this case, the first m objects are previously existing objects, whereas the $(m+i)^{th}$ object represents the creation of a new object for the i^{th} data item. For the tracking application, new objects represent the possibility that a new person enters the camera view. The solution to the CP, an assignment for each X_i, specifies a complete assignment for the items at this point in time.

Given the previous k-best possible worlds (i.e., parent possible worlds), our CP-based method initially creates k separate CPs, one for each parent possible world. All of these CPs are then solved to obtain k new possible worlds, where each possible world is the best solution based on its parent possible world. More specifically, each CP is formed by combining the corresponding parent possible world and the new data associations obtained after the parent possible worlds are constructed. Then, these CPs and their solutions are inserted into a priority queue. Using the queue, we obtain a new set of k-best possible worlds. Our CP-based method bears some similarity to Murty's algorithm [5]. The key difference of our method is that it can effectively prune the search space, as explained below, using constraint resolution.

Algorithm 1 CP-based algorithm for PDA problem

```
1. state ← {(cp_1, soln_1), ..., (cp_k, soln_k)}
2. topk ← ∅
3. while |topk| ≤ k do
4.     (cp, s) ← state.pop()
5.     topk.insert(s)
6.     cp_new ← cp.update(s)
7.     s_new ← cp_new.solve()
8.     state.insert(cp_new, s_new)
```

Algorithm 1 shows the overall operation of our CP-based method. The priority queue mentioned above is denoted as *state* (see line 1). At each iteration of the while-loop (lines 3-8), the current best solution is taken from the priority queue (*state*) and added to the results list (*topk*). Furthermore, the CP of the best solution is modified to obtain the next best solution, which is then added to the priority queue. If the solution is a set of assignments $X_i = j_i$ for $1 \leq i \leq n$ and $j_i \in 1, \ldots, m+n$, then the modification to the CP consists of adding a new constraint, $\vee X_i \neq j_i$, to remove the current solution from the solution space.

As mentioned before, a constraint solver performs an efficient search on the space specified by the domains of the problem variables [22]. Hence, our CP method can be viewed as a search-based MHT implementation with embedded constraint resolution techniques. For comparison, we also implemented a search-based MHT algorithm as outlined in [4]

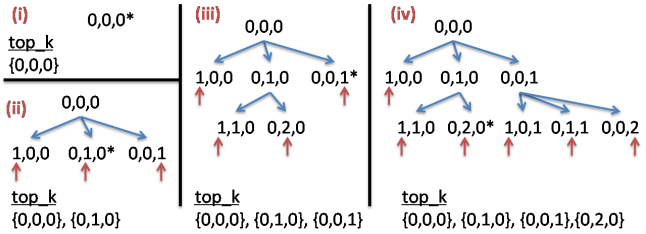


Figure 4: Example execution of *Ranked Search*

which performs a breadth-first search on the set of assignments using score-based bounds for pruning. However, in our experiments it was outperformed by our CP solution. Therefore, we do not provide the results for the previous search-based method in this paper.

3.2.2 Ranked Search

Consider the case where each tuple belongs to a single mutual exclusion group (MEG) and there are no additional constraints. Then, the possible world with the highest score consists of the maximum score assignment of a data item to an object for each MEG. If we sort the tuples for each MEG by score, the 0^{th}-index tuple (i.e., the highest scoring tuple) in each group will constitute the maximum score possible world. In this case, the solution is found immediately after sorting. Therefore, the run-time complexity of finding that solution is $O(nm \log m)$, where n is the number of items at the current point in time and m is the number of objects. $\{0, 0, 0\}$ in Figure 4 (i) represents the maximum score possible world which consists of the 0^{th} tuple for each of three 3 MEGs.

Given the above possible world, the next best possible world must consist of the 1^{st} tuple for one of the three MEGs and the 0^{th} tuple for the other MEGs. As Figure 4 (ii) shows, such a possible world has a single 1 together with 0s in its set of indices. Figures 4 (iii) and (iv) illustrate the process of finding the 3^{rd} and 4^{th} best possible worlds by incrementing one of the values in the index set.

In Figure 4, the possible world with the next highest score (denoted with a * sign) is added to the top-k list. The newly added world is also expanded using index increments to obtain a set of next possible worlds. In the figure, each index-set has a pointer to its leftmost modified index (see the up arrows). During each expansion, only the indices before the index pointer can be modified. This restriction guarantees the distinctness of the possible worlds that are examined. In other words, each branch of the search tree explores a distinct sub-space.

The expansion process mentioned above has the following properties:

(i) Let the sum of the values of an index-set be s. Then, the index-set is at level s of the tree and is reachable in s steps.

(ii) All possible values of the index-set are reachable from the root of the search and no value is produced more than once (based on the property that only the modifications before the index-pointer are allowed).

(iii) All ancestors of an index-set have less or equal values at each index position and therefore have higher score values.

The run-time complexity of our *Ranked Search* algorithm is $O(nm \log m + n(k-1))$. The second term represents the worst case scenario for the number of nodes expanded during the search process. The search tree in Figure 4 is used for illustration purposes only. In practice, a priority queue, enabling immediate access to the possible world with the best score, is used. The described *Ranked Search* algorithm only works for the case in which there is a single parent possible world at the previous point in time (i.e., $k = 1$). To handle the more general case with multiple possible worlds, we only need to modify the algorithm (i) to generate a separate root index-set for each parent possible world and (ii) to make index increments aware of the parent possible world by skipping the indices of the tuples not belonging to the parent possible world. The rest of the algorithm is unchanged and each index-set, regardless of its parent possible world, is inserted into the same priority queue.

Finally, constraints and their evaluation constitute an important part of the *Ranked Search* method. To support the use of constraints, we modify the algorithm so that it adds a possible world to the top-k list if and only if that possible world satisfies the constraints. As discussed before, in this case the complexity of the problem depends on the given set of constraints. We discuss how to efficiently resolve constraints using constraint-based pruning strategies in Section 4.

4. PDA CONSTRAINTS

We consider a variety of commonly used constraints in PDA applications. We support generic boolean constraints on data associations, and user-defined function-based constraints. In Section 4.1, we present generic constraint resolution techniques for boolean and function-based constraints. Then in Section 4.2, we discuss how to utilize existing specific constraint resolution techniques using as example a *bipartite matching* [25] algorithm for efficiently supporting a commonly used *unique assignment* constraint.

4.1 Generic Constraint Resolution

In the CP-based method, boolean constraints on assignments are embedded in the CPs. For instance, based on the physical locations of the cameras, we might know that two items, X_1 and X_2, cannot be the same object: $X_1 \neq X_2$. This is called a *pairwise inequality* constraint. On the other hand, the user-defined function-based constraints cannot be handled in this way. Such constraints are evaluated separately: each possible world, obtained as the solution to a CP, is checked for satisfying the function-based constraints. If the constraints are not satisfied, the CP is modified as described in Section 3.2.1 to obtain the next best solution.

In the Ranked Search method, both boolean and function-based constraints are evaluated together on possible worlds before being added to the top-k list. In the case where a possible world fails a boolean constraint, we only need to consider modifying the variables that are part of the constraint to obtain a solution. Therefore, during the expansion process in Section 3.2.2, the index incrementing operation is only performed on the variables of the failed constraint(s). For instance, in the case of the pairwise inequality constraint $X_1 \neq X_2$, the index incrementing operation will only change the variables X_1 and X_2. The other child possible worlds are pruned, without being formed, since it is certain that they cannot satisfy the constraints. To handle the user-defined

function-based constraints, we provide an interface that allows functions to return the set of assignments to change for satisfying the constraints. Given this information, function-based constraints can be handled similarly.

4.2 Specific Constraint Resolution

We observe that not all constraints can be handled efficiently using the generic constraint resolution methods described in Section 4.1. Specifically, in the cases where the violation of a constraint is caused by a large subset (or all) of the assignment items, the generic methods may take a long time for resolving the constraints. For this kind of constraints, it can be more efficient to use specific constraint resolution techniques. In this section, we discuss how we can utilize existing efficient solutions for such constraints in our framework.

Constraint solvers are already equipped with optimized algorithms for resolving specific constraints including *AllDifferent*, *Range*, *Sequence* constraints and others [22]. These constraint resolution techniques can be utilized by embedding the query constraints in the CPs. In this section, we consider the unique assignment constraint, which specifies that a given set of items are to be assigned to distinct objects. The unique assignment constraint can be solved efficiently using the *AllDifferent* constraint, which forces its argument variables to assume different values from one another. Observe that the unique assignment constraint can also be expressed as multiple pairwise inequality constraints in which case the solution will be much less efficient.

The Ranked Search method is ill-suited to efficiently solve the unique assignment constraint since its solution would be based on pairwise inequality constraints. The worst-case scenario would be the case of bipartite matching in which all of the variables are to be assigned different values. This is a common constraint in the case of feature tracking in which all of the features are to be matched against each other. However, there is an efficient algorithm, the Hungarian Method [25], that solves the bipartite matching problem in polynomial time. We can incorporate this algorithm within the Ranked Search method by replacing the index incrementing operation with the Hungarian method for the cases where the unique assignment constraint is violated. Observe that we can use this method to handle constraints on subsets of variables by solving a subproblem in which only the constrained variables are free and the rest of the variables are fixed. This hybrid constraint resolution technique enables the Ranked Search method to efficiently solve complex constraints using specialized algorithms. The Ranked Search algorithm will correctly compute the set of top-k possible worlds as long as the integrated constraint resolution methods are optimal.

5. OPTIMIZATIONS

5.1 Reducing the Computation Time

Until now, we assumed that the parameter k for a query has a user specified and fixed value (similar to top-k queries). In practice, however, it is not clear how to set this value. In addition, it is likely that different k values are better fitting for data association at different time points, thereby requiring an adaptive mechanism to adjust the value of k during runtime. The reason is that, depending on the score distribution, the top possible world scores could be very similar or really far apart. In the latter case, lower ranked possible worlds, though still within top-k, are unlikely to lead to top-k possible worlds at the next point in time and therefore will not contribute to the query result.

Based on this observation, we propose an adaptive mechanism that dynamically adjusts the k-value based on a statistical test, instead of using a fixed value, and reduces the computation time without significantly changing the query results. At a high level, we keep track of the change between the scores of the top possible world and the top i^{th} possible world. When this change has a low probability, according to the gathered statistics, we stop generating possible worlds (i.e., k is set to i) and return the result.

More specifically, we define a variable CR_i, the change rate of the top i^{th} possible world:

$$CR_i = \frac{p_i - p_{i+1}}{p_1}$$

where p_j is the score of the top j^{th} possible world. We assume that change rates are i.i.d. variables with mean μ and variance σ^2. Then based on the Chebyshev's Inequality [18] we have:

$$P(|CR_i - \mu| \geq c\sigma) \leq \frac{1}{c^2}$$

where $c > 0$ is a constant. We use a cumulative version of this inequality:

$$P(|\sum_{1}^{n} CR_i - n\mu| \geq c\sqrt{n}\sigma) \leq \frac{1}{c^2}$$

Here, $\sum_i^n CR_i$ is the cumulative change rate of the top n^{th} possible world and $\sum_i^n CR_i = \frac{p_1 - p_{n+1}}{p_1}$. Finally, we remove the absolute sign from the inequality to only consider the cases where the argument has a positive value. Otherwise the cumulative change rate is less than its expected value, which means that the possible world scores are close to each other. Therefore, given a probability threshold, $\frac{1}{c^2}$, our stopping condition is simply the given inequality test. This method is used together with a timeout mechanism or a maximum k value for the cases when the test always passes.

5.2 Improving the Association Accuracy

In this section, we first discuss how to measure the accuracy of the PDA query results and then present methods to increase accuracy without increasing the query processing times.

5.2.1 Accuracy Metrics

Given the ground truth (GT) data association for a PDA problem, there are various forms of accuracy metrics for data association [26]. An intuitive metric, variations of which are widely used by tracking applications, is the sequential accuracy, in which pairs of items associated with the same object in sequential time points / frames are identified and compared with the GT. If the GT also has the given items assigned to a single object, the item is said to be correctly associated, or a true positive (TP). If the association only exists in the query result but not in the GT, it is called a false positive (FP). Finally, if the association exists in the GT but not in the result, it is a false negative (FN). Then, the well-accepted precision and recall metrics of Information Retrieval can be applied: precision $= \frac{TP}{TP+FP}$ and recall $= \frac{TP}{TP+FN}$. Observe that, this metric focuses on individual item associations instead of continuous object tracking.

Hence, it is possible that each object is tracked correctly for a small number of frames but a high accuracy score is obtained. For instance, consider a single object video stream of 10 frames, for which 5 different 2-frame long objects are output as the tracking results (i.e., every two frames tracking fails and a new object is detected). In this case, the sequential accuracy would have 100% precision and 50% recall. For this reason, we can also use an object-based metric that compares the objects in the query result with the GT. In this case, we compare the first detected item for the object that an item is assigned to in the query result, with the first detected item of the corresponding object in the GT. TPs, FPs and FNs are defined similarly to the sequential accuracy metric.

Our top-k approximation method is based on the assumption that given relevant score assignments for data items (i.e., not conflicting with the GT), there is a strong correlation between high possible world scores and accuracy. This relationship is demonstrated experimentally in Section 6.

5.2.2 Hypothesis Sampling

In general, we expect the accuracy of the query results and the top possible world scores to increase with increasing values of k (until the maximum score possible worlds are obtained). Therefore, the first method of increasing accuracy is to increase the k-value. However, the value of k is limited due to computational reasons. Depending on the specified constraints and for large values of k, both the CP-based method and the Ranked Search method may take a long time to find the top-k possible worlds. In the worst case, an exponential number of possible worlds may need to be enumerated. In addition, a lower-ranked possible world in a frame set could lead to a high score possible world in later frame sets as the change in scores across frame sets is unlikely to be the same for all possible worlds. Given a fixed k value, such possible worlds could be prematurely eliminated. Finally, high score possible worlds tend to be very similar in terms of their assignments, which reduces the ability to recover from erroneous assignments.

We now add an additional *sampling step* to the Ranked Search method to address the mentioned issues. After the top-k possible worlds are found, we sample from the rest of the already computed possible worlds. The sampled possible worlds that satisfy the constraints are added to the top-k list. The sampling itself can be done uniformly at random or weighted by score. As future work, we plan to explore various sampling techniques and measure their effectiveness. Observe that the newly added possible worlds (i) are not necessarily high-score possible worlds and (ii) increase assignment *diversity* (as they originate from different parent possible worlds). Finally, notice that the sampling does not notably increase the computation time as it is performed over the already computed possible worlds; i.e., no additional iterations are performed to form new possible worlds.

6. EXPERIMENTS

6.1 Experimental Setup

In this section, we present an experimental evaluation of our algorithms using the visual tracking application described in Section 2.4.

6.1.1 Testbed

We have set up a 24 camera network (indoors) over a 100 Mbps wired LAN in our department as our testbed. In our setup, each camera [8] captures 5 frames (of size 640x480 pixels) per second (fps) and continuously forwards frames to a desktop computer in the same LAN for processing/storage. Although we had to limit the frame rate to avoid overloading the network, in our experiments we report processing times (per frame) that demonstrate that our system can easily cope with much higher data rates.

Video streams are processed by our computer vision (CV) library which extracts blobs and other image features and stores them in a central database. Our framework supports both real-time tracking on live videos and tracking on recorded videos using the central database (for debugging and retrospective analysis). For repeatability purposes, in the experiments reported here we use previously recorded videos. Our CV library, tracking application and all other tools are implemented in C++. We use the VXL library [9] for image processing and the ILOG [22] constraint solver to solve constraint programs.

6.1.2 Cooking Images: From Raw Data to Relations

We now describe how we "cook" the images within our visual tracking system. Notice that cooking is in general application-specific; we simply apply common practice and techniques and describe them for completeness and to facilitate repeatability.

Color-histogram and location-based matching [19] are the two similarity functions used to compute data association scores in this study. A color-histogram represents the color distribution of the pixels in a given image region. Our color-histogram implementation is a 3-dimensional histogram with 8 bins in each dimension based on the RGB color space [19]. The color-histogram matching function computes a score based on the similarity of the color-histogram of a data item with the color-histograms of an object. We use an intersection-distance based function for computing the similarity of two color-histograms. The intersection-distance function first normalizes each color-histogram and then sums the minimum of the values in the two histograms for each bin. On the other hand, for the location-based matching function, we model the motion of objects using a Kalman-filter [17] and use its prediction results in location estimation. In our experiments, matching score is computed as a weighted combination of the color-histogram and location-based similarity functions.

The objects considered for tracking in the experiments are people and foreground image features. There are a variety of image features available for tracking. In this study, we use a corner feature [21] provided by the VXL computer vision library. In addition, in a tracking application each object must be represented with a model based on the information required by the similarity functions. The representation model for objects is application dependent. In our implementation, the object representation models consist of a motion model and a bag of color-histograms.

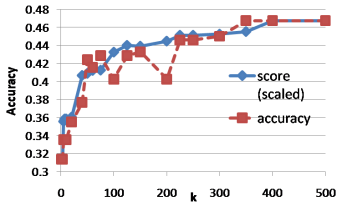
In our experiments, we normalized each similarity function to return a unit value. In addition, to avoid numerical underflows, we use the logarithm of each assignment score and maximize the sum of the log-scores instead of the product of the scores. The scores reported in the experiments are log-scores unless otherwise stated.

We note that it is common practice to derive probability values from the scores assigned by user-defined similarity functions in many real-world applications (such as handwriting recognition, document classification, and keyword-based Web search) in a manner similar to the way we describe it here for visual tracking. The discussion of better probabilistic models is outside the scope of this paper. Our approaches do not make any assumptions about how the probability values are derived and thus can work with alternate interpretations.

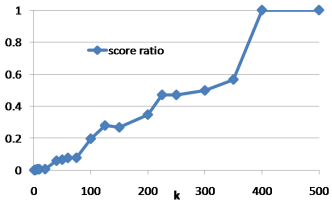
In Section 6.2, we present experiments on the relation between accuracy and possible world scores using synthetic datasets. Then in Section 6.3 we present experimental results using real datasets.

6.2 Tracking Accuracy

In this experiment, we use a single-camera synthetic video stream of 10 frames formed by rotating a single image 2 degrees clockwise (with respect to its upper left corner) per frame. 42 features, extracted from the foreground pixels in the original image, are rotated with the image. Hence, each feature performs a circular motion with the same angular speed. As a result, we know the exact feature locations (i.e., the ground truth) in all frames using which we can precisely compute tracking accuracy. This is a common "trick" used when evaluating tracking solutions.



(a) Accuracy and score



(b) Top score ratios

Figure 5: Score and Accuracy Relationship

We tracked the features using Ranked Search with different k values (from 1 to 500). The accuracy vs. score comparison for the top possible world obtained with each value of k is provided in Figure 5(a). The used accuracy metric is the object-based metric described in Section 5.2. The shown curve is the precision curve and in this case it is the same with the recall curve as we used bipartite matching and each false negative also caused a false positive (and vice versa). The score values are normalized to fit in the accuracy interval. The general trends (i) increasing score with increasing k value and (ii) increasing accuracy with increasing score are displayed. Based on this result, in the rest of the experiments, we use increasing score values as indicators for increasing accuracy.

We should point out that the association accuracy reported in our experiments is fundamentally a function of the quality and resolution of the underlying raw data and the similarity functions used. As such, the nominal accuracy values achieved are not an indication of how well our proposed algorithms work. These values should be interpreted in a relative manner; e.g., our algorithms should not unnecessarily degrade the best possible accuracy achievable in our setting in order to improve processing efficiency.

The maximum achieved object-based accuracy ($\approx 48\%$) is low. However, with the sequential accuracy metric, we obtained values between 90% for $k = 1$ and 93% for $k = 500$ on the same results. The high sequential accuracy shows that while frame-by-frame tracking is accurate, feature-to-object mappings are less accurate. The reason is the lack of prior information on the feature motion models. In addition, the color-histogram function was not used in this experiment since the features across frames are identical (the color-histogram would therefore provide certain information).

The scores shown in Figure 5(a) and the rest of the section are log-based. However, in Figure 5(b), we show how the actual score value (i.e., not log-based) changes with k-value for the same results in Figure 5(a). The values are normalized using the best score.

6.3 Tracking Methods and Performance

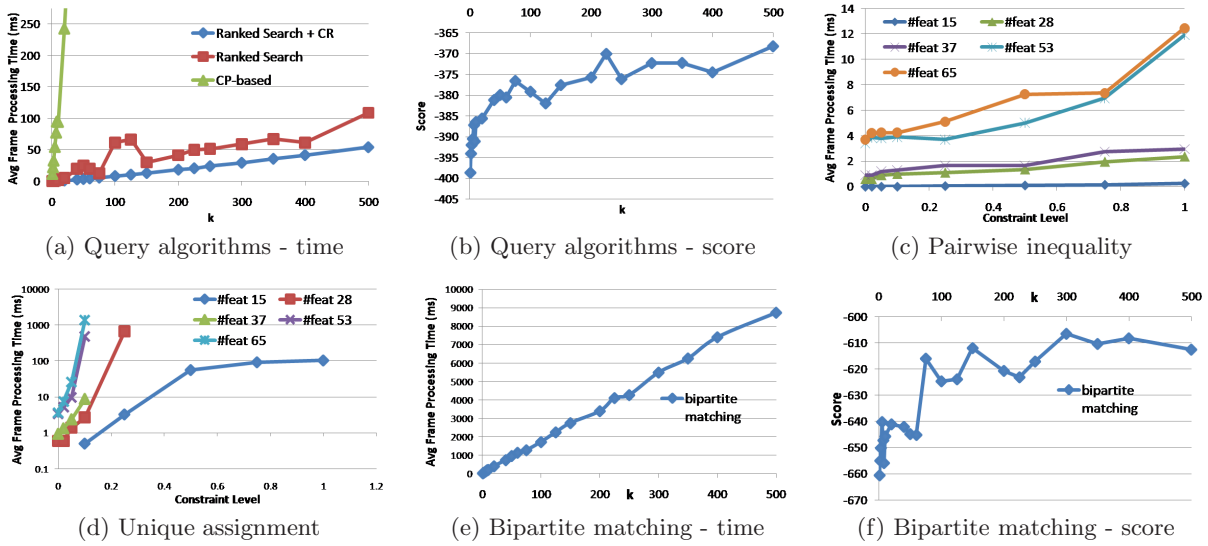
6.3.1 CP-based method vs Ranked Search

In this experiment, we compare PDA query execution methods using a video stream of 268 frames from 3 video cameras (804 total frames). In this case, we are tracking 2 people walking towards each other, crossing paths and leaving the camera views in separate directions. The query execution times for each method is shown in Figure 6(a). A uniqueness constraint (i.e., blobs in the same camera are assigned to different objects) is enforced. As mentioned in Section 3.2.1, our CP-based solution can be viewed as a search-based MHT implementation with embedded constraint resolution techniques. In contrast to previous MHT implementations [2, 3, 4, 5], our CP-based solution effectively prunes its search space using constraint resolution.

The CP-based method, despite using the *AllDifferent* construct (Section 4.2), is unable to handle large values of k and requires seconds for processing k values greater than 50. The Ranked Search method (without the constraint resolution techniques) performs much better. The best performance is obtained with the Ranked Search method using the constraint resolution (Ranked Search + CR) techniques. Observe that, while the Ranked Search method in general requires longer execution times for the larger k values, in some cases it can take longer times for smaller values of k as well. The reason is the lack of CR techniques which makes the execution time mainly dependent on the constraint instances. While there are only 2 people in this video, there are also noise blobs being observed due to imperfect detection of foreground pixels which makes the problem harder. The query results obtained with all the execution methods are the same and we show the best score obtained with each value of k in Figure 6(b).

6.3.2 Constraint Resolution

I. Generic Constraint Resolution: In this experiment, we analyze how the query execution times change with (i) the number of tracked objects and (ii) different levels and types



(a) Query algorithms - time (b) Query algorithms - score (c) Pairwise inequality

(d) Unique assignment (e) Bipartite matching - time (f) Bipartite matching - score

Figure 6: Comparison of PDA query execution methods and constraint resolution techniques

of constraints using a fixed value of $k = 10$ and generic constraint resolution techniques (Section 4.1). The test video contains 20 frames from a single camera.

Local Constraints: *Local constraints* are constraints that affect only a small subset of the variables [23]. As a local constraint example, we consider the pairwise inequality constraint (Section 4.1) (affects only 2 variables). In Figure 6(c), we show the average query execution times (per frame) for tracking different number of features (15, 28, 37, 53 and 65) under different constraint levels. The constraint level specifies the ratio of the features on which pairwise inequality constraints are placed. The results show that the increase in the number of features for low constraint levels does not notably affect the execution time. However, the execution time becomes more sensitive to the number of features as the constraint level increases.

Global Constraints: Now we consider the unique assignment constraint (Section 4.2) which is a *global constraint* (affects all variables). In Figure 6(d), we show the execution times for tracking different number of features with varying levels of the unique assignment constraint. In this case, the constraint level specifies the ratio of the features on which a unique assignment constraint is placed. When the constraint level is 1, the problem is equivalent to the bipartite matching problem. The results show that it is much harder to solve the unique assignment constraint with the generic constraint resolution methods. We can only obtain the full range of results for the case with 15 features. With a larger number of features, we ran out of memory before we could find the top-10 possible worlds.

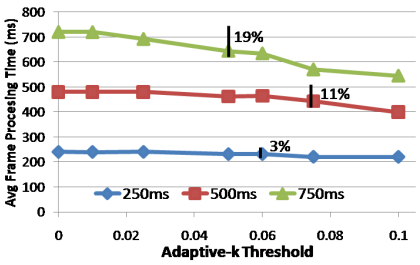
II. Specific Constraint Resolution - Bipartite Matching: In this experiment, we focus on the specific constraint resolution techniques within the context of bipartite matching (i.e., one-to-one matching of features to objects Section 4.2). The test video contains 23 frames with an average number of 38 features per frame. We execute feature tracking for different k values (from 1 to 500). The average query execution times are given in Figure 6(e). The execution time increases linearly with the value of k. We were unable to compute the top-10 possible worlds using generic CR techniques for more than 15 features and level-1 unique assignment constraint. With specific CR techniques, however, we could obtain these results for $k = 500$. In addition, we show the score curve for the best score obtained with each value of k in Figure 6(f).

6.3.3 *Adaptive-k Processing*

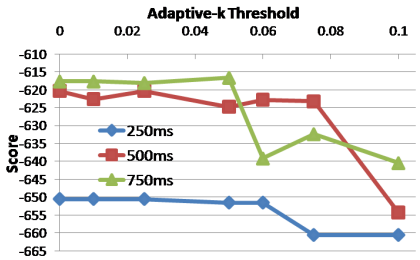
In this experiment, we analyze the effects of the adaptive-k mechanism using different threshold values and frame processing time limits, described in Section 5.1, on the score and execution time performance of the Ranked Search algorithm. We track the people in a video of 23 frames using the Ranked Search algorithm with the adaptive-k setting. The time limit specifies the maximum time allowed for processing a single frame, which is used to time out processing when the adaptive-k test is unable to stop processing. Average per frame query execution times are shown in Figure 7(a) for different values of the threshold and the time limit.

With a 250ms time limit, the adaptive-k mechanism is unable to notably reduce the computation time even with the larger values of the threshold. As discussed in Section 5.1, the threshold indirectly specifies the maximum cumulative drop in possible world scores before processing is stopped. The fact that the adaptive-k mechanism is unable to stop processing means that the possible worlds generated in 250ms are all within the score limits specified by the threshold.

For larger values of the time limit, the adaptive-k mechanism is able to reduce the average frame processing time. However, this is possible due to a trade-off between score and processing time. The score values obtained in the experiment are shown in Figure 7(b). For a 750ms time limit, we can use a threshold value of 0.05, to reduce the average computation time by 142ms ($\approx$ 19%) and still obtain a similar top possible world score. Similarly with 500ms time limit, we can use a threshold of 0.075 and reduce the average computation time by 57ms ($\approx$ 11%) without a significant loss in the best-score. The vertical lines in Figure 7(a) indicate this reduction in computation for these accuracy-cut based threshold values. For larger threshold values, the achieved best-score is smaller. The reason is that the possible worlds

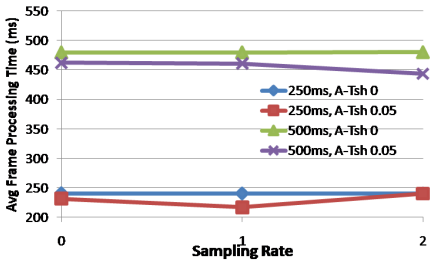


(a) Adaptive-k vs. time

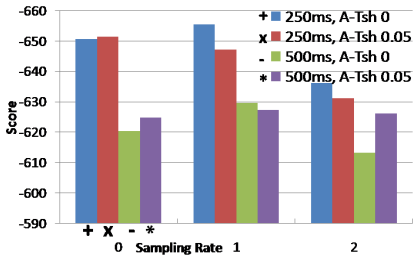


(b) Adaptive-k vs. score

Figure 7: Ranked Search with Adaptive-k



(a) Sampling - time



(b) Sampling - score

Figure 8: Hypothesis sampling

that generate these best-scores are prematurely eliminated during processing.

6.3.4 *Hypothesis Sampling*

In this experiment, we focused on the effects of sampling on per frame average execution time and score performance of the Ranked Search algorithm. For comparability purposes, we used the same 23-frame video with the Adaptive-k experiment (Section 6.3.3). There are three different dimensions for the algorithm settings used in this experiment: (i) frame processing time limit (250ms and 500ms) (ii) adaptive-k mechanism (used with a threshold of 0.05 or not used) and (iii) uniform sampling with different rates. The sampling is done as an additional step as discussed in Section 5.2.2. The experiment results for all combinations are shown in Figures 8(a) and 8(b).

The sampling rates, 0, 1 and 2, specify the ratio of sampled possible worlds to the already computed top-k possible worlds. For instance, with a sampling rate of 1, if top-10 possible worlds are computed using the Ranked Search algorithm, then 10 additional possible worlds are sampled from the rest of the computed possible worlds. In Figure 8(a), as predicted in Section 5.2.2, we show that sampling does not increase the execution time of the tracking algorithm. In Figure 8(b) we compare the best-score obtained for each setting of the tracking algorithm. The results show that in all cases a higher score is obtained with a larger time limit. However, it is still possible to improve the achieved score using sampling. For the sampling rate 2, the score is improved significantly for the 250ms time limit, approaching the score obtained with the 500ms time limit, both with the adaptive-k mechanism and without it. The relative score improvement for the 500ms case is less. Yet with a sampling rate of 2, a score of -610 is achieved, which is the best observed score in this case, and beats all non-sampling based scores and the scores achieved with 750ms time limit shown in the adaptive-k experiment.

7. RELATED WORK

Multi-Hypothesis Tracking (MHT) [2, 3, 4] is a widely used data association model with many applications, that forms multiple hypotheses for object tracking. Similar to our work, in MHT implementations [2, 5] a top-k approximation is used for finding high score data association hypotheses. In [5], an MHT implementation based on the bipartite matching algorithm is provided applying to one-to-one matching problems only. As in our CP-based method, the problem is repeatedly divided into separate problems which are solved optimally till top-k solutions are found. However, our techniques for improving the efficiency and accuracy, as well as our integrated constraint resolution framework, are novel and, as we demonstrated experimentally, provide significant benefits enabling real-time PDA with high accuracy.

Other data association methods include the Global Nearest Neighbors (GNN) [19] method, in which each data item is assigned to the object it most likely belongs to. However, in GNN items are assigned independently and without any constraints. In addition, GNN is unable to recover from erroneous assignments. Another data association method is the Joint Probabilistic Data Association Filter (JPDAF) [20], which unlike MHT and GNN is an object-based approach. In JPDAF, multiple items could be assigned to an object and each object model is updated based on a weighted combination of the data items assigned to it. Observe that, in JPDAF there is no specific data association, instead an object-based view, describing the model of each object, is provided to the user.

A top-k query on uncertain data computes the best k results according to a given rank function. In prior research on probabilistic databases, top-k queries were extensively studied [6, 10, 11, 13] and there has been different formulations of top-k queries targeting different applications. For instance, in [6], definition for two types of top-k queries are given: *U-Topk* and *U-k Ranks*. The U-Top$k$ query returns k tuples which has the highest probability of being the top-k tuples in all possible worlds. In the U-k Ranks query, the

result is a list of k tuples where the i^{th} tuple has the highest probability for being the i^{th} ranked tuple in all possible worlds. The PDA query introduced in this paper involves a top-k approximation and can be viewed as a continuous top-k query. However, our main goal is to find high score possible worlds at each frame instead of finding the best-k hypotheses for a single frame. In addition, the PDA query results are complete possible worlds, not partial assignments obtained from possible worlds.

In [12], authors consider the duplicate detection problem for data integration and cleaning. The proposed solution is to view the dirty database as a probabilistic database and the clean database is then defined as one of the instances of the probabilistic database. While this definition is similar to our formulation of the PDA problem, their goal (to answer queries over the clean database without computing it) and methods (involving query rewriting techniques) are different.

8. CONCLUSIONS

Continuous probabilistic data association is a data-driven stream processing challenge that arises in a variety of real-world applications. We described how continuous query processing techniques can support PDA, showing experimental evidence that they can improve upon the existing solutions in terms of generality and performance.

Our primary contributions include algorithms for probabilistic constrained ranking over data streams and a resource-aware tuning approach that produces real-time results with modest losses in accuracy. We use a visual tracking application over a smart camera network to evaluate our solutions.

There are several open directions for future work on PDA. In addition to live PDA, many applications would like to extract data associations over stored, historical data. Such a functionality can greatly benefit from standard tools such as indexing, as well as novel execution strategies that look for associations at different spatio-temporal granularities. Another promising direction is the integration of the cooking process with PDA processing over the relational representation, which was the topic of this paper. In our model, cooking is handled independently and separately by application-specific logic, after which our PDA algorithms kick in. Here, there are opportunities for new cross-layer optimizations by, for example, using on-demand computation of the similarity scores, i.e., by extracting only those image features required by the PDA algorithm.

9. REFERENCES

[1] Krause, A., et al. Data Association for Topic Intensity Tracking. ICML, 497-504, 2006.

[2] Cox, I. J. and Hingorani, S. L. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. Pattern Anal. Mach. Intell. 18(2), 1996.

[3] Reid D.B. An Algorithm for Tracking Multiple Targets. IEEE Trans. on Automatic Control 24(6), 843-854, 1979.

[4] Blackman, S. Multiple Hypothesis Tracking for Multiple Target Tracking. IEEE A & E Systems Magazine 19, 2004.

[5] Murty, K. G. An Algorithm for Ranking All the Assignments in Order of Increasing Cost. Operations Research 16, 1968.

[6] Soliman, M. A., Ilyas, Ihab F., and Chang, K. C. Top-k Query Processing in Uncertain Databases. ICDE, 896-905, 2007.

[7] Dalvi, N. and Suciu, D. Efficient Query Evaluation on Probabilistic Databases. The VLDB Journal 16, 2007.

[8] DCS-900 Camera. http://www.dlink.com/products/?pid=270.

[9] Vision-X Libraries. http://vxl.sourceforge.net.

[10] Hua, M., Pei, J., Zhang, W., and Lin, X. Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach. SIGMOD, 673-686, 2008.

[11] Re, C., Dalvi, N. N., and Suciu, D. Efficient Top-k Query Evaluation on Probabilistic Data. ICDE, 886-895, 2007.

[12] Andritsos, P., Fuxman, A., and Miller, R. J. Clean Answers over Dirty Databases: A Probabilistic Approach. ICDE, 2006.

[13] Ge, T., Zdonik, S., and Madden, S. Top-k Queries on Uncertain Data: On Score Distribution and Typical Answers. SIGMOD, 375-388, 2009.

[14] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. CIDR, 262-276, 2005.

[15] Benjelloun, O., et al. Databases with Uncertainty and Lineage. VLDB Journal 17(2), 243-264, 2008.

[16] Suciu, D., et al. Probabilistic Databases. Morgan & Claypool Publishers, 2011.

[17] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME 82(D), 35-45, 1960.

[18] Ross, S. A First Course in Probability. Prentice Hall, 2008.

[19] Forsyth, D. A. and Ponce, J. Computer Vision: A Modern Approach. Prentice Hall, 2002.

[20] Bar-Shalom, Y. Tracking and Data Association. Academic Press Professional, 1987.

[21] Smith, P., Sinclair., D. et al. Effective Corner Matching. BMVC, 1-12, 1998.

[22] IBM ILOG CP Optimizer. www.ilog.com

[23] Marriott, K., and Stuckey, P. J. Programming with Constraints: An Introduction. MIT Press, 1998.

[24] Kanagal, B. and Deshpande, A. Online Filtering, Smoothing and Probabilistic Modeling of Streaming Data. ICDE, 1160-1169, 2008.

[25] Papadimitriou, C. H., et al. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, 1998.

[26] Manohar, V. et al. Performance Evaluation of Object Detection and Tracking in Video. ACCV, 151-161, 2006.

[27] Brown, D. E. and Hagen, S. Data Association Methods with Applications to Law Enforcement. Decision Support Systems 34(4), 369-378, 2003.

[28] Apt, K. Principles of Constraint Programming. Cambridge University Press, 2003.

[29] Kumar, V. Algorithms for Constraint-Satisfaction Problems: A Survey. AI Magazine, 1992.