UNIVERSITY OF CALIFORNIA,
IRVINE

Variational Message-Passing: Extension to Continuous Variables and Applications in Multi-Target Tracking

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Andrew J. Frank

Dissertation Committee:
Alexander Ihler, Co-Chair
Padhraic Smyth, Co-Chair
Rina Dechter

2013

DEDICATION

To my wonderful wife and parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

I would like to thank my advisors, Alex Ihler and Padhraic Smyth, for their guidance and support on this long journey. They offered me freedom to explore and pressed me to grow. Together they have shaped the way I think about research and machine learning; although I have absorbed only a portion of the wisdom they shared over the years, they have taught me a great deal.

I would also like to thank Rina Dechter, my third committee member. I was fortunate to take Rina's belief networks class my first quarter at UCI, and she kindled my interest in what would become the focal point of my research.

I would like to thank Jim Randerson for his fruitful collaboration. In addition to being an excellent researcher in his own field, Jim was instrumental in bridging the gap between our two disciplines.

I am grateful to the School of ICS for supporting me with a Dean's Fellowship. The remainder of my research was funded by grants from the Office of Naval Research (MURI grant N00014-08-1-1015) and the NSF (grant IIS-1065618).

Of course, my experience at UCI has been largely shaped by my peers. There are too many to name, so I will just acknowledge the students of the Smyth, Ihler, Dechter, Baldi, and Welling labs as the finest bunch of friends and coworkers one could hope for.

Finally, I would like to thank my wife, Dr. An Tyrrell, who has been a constant source of love, support, and happiness in my life. And my parents, Andy and Joan, who taught me that nothing is out of reach.

CURRICULUM VITAE

Andrew J. Frank

EDUCATION

Doctor of Philosophy in Computer Science **2013**
University of California, Irvine *Irvine, CA*

Masters of Science in Computer Science **2009**
University of California, Irvine *Irvine, CA*

Bachelor of Science in Computer Science **2007**
Washington University in St. Louis *St. Louis, MO*

Bachelor of Science in Electrical Engineering **2007**
Washington University in St. Louis *St. Louis, MO*

ABSTRACT OF THE DISSERTATION

Variational Message-Passing: Extension to Continuous Variables and Applications in Multi-Target Tracking

By

Andrew J. Frank

Doctor of Philosophy in Computer Science

University of California, Irvine, 2013

Alexander Ihler and Padhraic Smyth, Co-Chairs

This dissertation focuses on both the application and development of variational inference algorithms for probabilistic graphical models. First, we propose a new application of graphical models and approximate inference in the multi-target tracking domain. By constructing a factor graph representation of the track-oriented multiple hypothesis tracker, we enable the application of variational inference algorithms to efficiently estimate marginal probabilities of possible tracks. We then show that that these track marginals are the key ingredient in a multi-target generalization of the standard expectation-maximization algorithm used for parameter estimation in single-target tracking. The resulting online estimation algorithm makes the tracker robust to parameter misspecification and can improve performance in settings with non-stationary target dynamics. Next, we develop a general framework to extend algorithms for approximate marginalization in discrete systems to work with continuous-valued graphical models. We extend the particle belief propagation algorithm, which uses importance sampling to lift the sum and product operations of belief propagation from a variable's continuous domain into an importance-reweighted particle domain. We demonstrate that this framework admits other variational inference algorithms such as mean field and tree-reweighted belief propagation, and that they confer similar qualitative benefits to continuous-valued models as in the discrete domain.

Chapter 1

Introduction and Background

Statistical modeling has emerged as one of the most successful tools for modern machine learning applications. Fundamentally, machine learning is about extracting insight from data and making predictions. Statistics provides a rigorous and flexible framework for *learning* structured representations of raw data, and probabilistic *inference* provides the corresponding tools to use these structured representations to extract insight and make predictions. This dissertation focuses on inference, proposing a new strategy for approximate inference in continuous-valued models and applying existing algorithms to the problem of multi-target tracking.

Graphical models [40] serve as a common language for expressing statistical modeling assumptions. In addition to facilitating communication and exploration of modeling techniques, this common language serves as a substrate on which general purpose inference algorithms can operate, partially decoupling the domain-specific model-building process from the domain-independent learning and inference processes. As a result, graphical models have enabled successful machine learning applications in areas ranging from text analysis [13, 46] to neural imaging [86].

Chapters 2 and 3 of this dissertation propose a novel application of graphical models and approximate inference algorithms to the problem of multi-target tracking. Specifically, we focus on the *track-oriented multiple hypothesis tracker* (TOMHT) [45, 10], which is typically viewed as an approximate *optimization* algorithm for identifying the single most likely set of target trajectories (tracks) from a collection of sensor data. We formulate the modeling assumptions behind the TOMHT in the language of graphical models, revealing a rich conditional independence structure that admits efficient application of approximate *marginalization* algorithms like belief propagation [56]. In experiments on simulated sensor data, we investigate the use of several approximate inference algorithms for the purpose of estimating the marginal probabilities of possible tracks. We then develop an expectation-maximization (EM) algorithm for learning parameters of the tracking model. This learning algorithm requires track marginals as inputs, and we show that approximate marginals computed via belief propagation are sufficiently accurate in the context of this learning algorithm to improve tracker performance.

Chapter 4 transitions away from multi-target tracking and focuses instead on a methodological extension of approximate inference algorithms. As previously mentioned, one of the most attractive aspects of the graphical model framework is that it enables application of general purpose inference algorithms in a domain independent manner. Consequently, considerable work has gone into advancing the state of the art for approximate inference in graphical models [37, 82, 75, 79].

Unfortunately, many of these more advanced methods are applicable only to graphical models with discrete random variables. This restriction effectively creates a wall between discrete and continuous-valued graphical models, splitting the spaces of models and inference algorithms into two incompatible sets and detracting from the promise of graphical models as tools for truly general purpose inference. To this end, we propose a framework based on the particle belief propagation algorithm [35] for extending these algorithms to continuous-

valued graphical models. Our experiments show that the extensions of these recent discrete inference algorithms to continuous models confer the same benefits as in the discrete domain.

The contributions of this dissertation share a common theoretical foundation of probabilistic graphical models and approximate inference algorithms. In the remainder of this chapter, we first provide a structured summary of the dissertation's main contributions and then review the shared theoretical foundations, introducing notation that will be drawn on in subsequent chapters.

1.1 Contributions

This section summarizes the main contributions of this dissertation, organized by chapter:

Chapter 2

- We present a novel factor graph formulation of the TOMHT's probabilistic model.
- We propose variational message-passing on the factor graph as a novel method for approximating track marginal probabilities in the TOMHT.
- We conduct an empirical evaluation of two approaches to approximating track marginal probabilities – our new approach using variational message-passing, and an existing approach based on the k-best data association hypotheses.

Chapter 3

- We develop a novel approximate EM algorithm for estimating parameters of the TOMHT model. The E-step of this algorithm uses approximate track marginals, computed as

in Chapter 2.

- We demonstrate experimentally that the EM algorithm makes the TOMHT robust to parameter misspecification, even when using approximate track marginals estimated via BP.

- We compare performance of the EM-enabled TOMHT using three different track marginal estimators, and show that the BP-based marginal estimator results in better performance than other estimators of comparable speed.

- We demonstrate that online EM can improve tracker performance relative to the best static-parameter tracker when the true target dynamics model changes over time.

Chapter 4

- We introduce a general framework for extending variational message-passing algorithms to work on continuous graphical models via particle BP.

- For the case of tree-reweighted BP, we demonstrate experimentally that its qualitative characteristics carry over directly to continuous problems.

- We provide a finite sample analysis of the weighted mini-bucket primal bound on the partition function.

- In a simulated sensor self-localization problem, we demonstrate that tree-reweighted particle BP represents uncertainty more accurately than particle PB when the true marginal distributions are multimodal.

1.2 Probabilistic graphical models

Graphical models [40] have grown to be the lingua franca of large-scale probabilistic modeling. Their use offers two main benefits. First, they compactly summarize a large number of modeling assumptions. This aids in communication between researchers, allowing one to understand the essence of a complex probabilistic model at a glance. Second, as we will see in the next section, they provide a common structure on which generic inference algorithms can operate. This section briefly motivates the use of probabilistic modeling, and then develops factor graphs, a specific type of graphical model.

1.2.1 Probabilistic modeling

Many deterministic real-world phenomena can be well approximated by random processes. As an example, consider the process of digitizing a black and white photograph. Let $\boldsymbol{x}$ denote the resulting $m \times n$ image produced by an ideal scanner that creates perfectly faithful copies of its input. Further, let $x_s \in [0, 255]$ denote the intensity value of the s^{th} pixel of $\boldsymbol{x}$, with s ranging from 1 to mn. In practice our scanner is unlikely produce $\boldsymbol{x}$, but $\boldsymbol{x}$ is the image we would *like* to obtain. Now consider a more realistic scenario, where the output image is a corrupted representation of the original photograph. Specks of dust in the scanner bed, imperfections in its manufacturing, and a variety of other factors will conspire to produce an imperfect digital copy $\boldsymbol{z}$.

Thus, we can view the digitization process as an extremely complex deterministic function that takes $\boldsymbol{x}$ as an input and produces a corrupted output, $\boldsymbol{z}$. This view is illustrated in the following diagram:

$$\boldsymbol{x} \longrightarrow \boxed{\boldsymbol{z} = f(\boldsymbol{x})} \longrightarrow \boldsymbol{z}$$

where f is the corrupting processing. If we could precisely characterize f, it would be possible to reverse its effects (at least partially – f may not be invertible) and end up with an image $\boldsymbol{x}'$ that is close to our ideal image $\boldsymbol{x}$.

Of course, in practice it is impossible to measure every speck of dust, the exact degree of warping of the scanner's surface, etc. This is where probabilistic modeling comes to the rescue: instead of trying to exactly represent the impossibly complex corrupting process, we can approximate it with a much simpler *random* process. This view of the world is illustrated as follows:

$$\boldsymbol{x} \sim \Pr(\boldsymbol{X}) \longrightarrow \boxed{\boldsymbol{z} \sim \Pr(\boldsymbol{Z} \mid \boldsymbol{X} = \boldsymbol{x})} \longrightarrow \boldsymbol{z}$$

In this view, $\boldsymbol{X}$ and $\boldsymbol{Z}$ are random variables distributed according to a joint probability distribution $\Pr(\boldsymbol{X}, \boldsymbol{Z})$. Notably absent is the complicated function f – as long as we can characterize the relative probabilities of all possible $(\boldsymbol{x}, \boldsymbol{z})$ pairs, we need not concern ourselves with exactly *how* the corrupting process works.

Inference

Armed with a probabilistic model for $\boldsymbol{X}$ and $\boldsymbol{Z}$, we can attempt to recover the ideal image $\boldsymbol{x}$ by performing inference. One natural reconstruction is the *maximum a posteriori (MAP)* estimate of $\boldsymbol{X}$:

$$\boldsymbol{x}' = \arg\max_{\boldsymbol{x}} \Pr(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Z} = \boldsymbol{z}). \tag{1.1}$$

The MAP estimate is the most likely value of $\boldsymbol{X}$ conditioned on our corrupted image, $\boldsymbol{z}$. Alternatively, we may want to pick the most likely value for each pixel separately, without regard for compatibility with the other pixels in our reconstruction. This equates to

computing the *marginal posterior modes* (MPM) of our distribution:

$$\begin{aligned} x'_s &= \arg\max_{x_s} \Pr(X_s = x_s \mid \boldsymbol{Z} = \boldsymbol{z}) \\ &= \arg\max_{x_s} \sum_{\boldsymbol{X}: X_s = x_s} \Pr(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Z} = \boldsymbol{z}). \end{aligned} \tag{1.2}$$

where the notation $\boldsymbol{X} : X_s = x_s$ indicates summing over all images where pixel X_s takes on value x_s. Collectively, the processes of optimizing, marginalizing, and conditioning on some or all of the variables of a model are referred to as *probabilistic inference*.

Learning

Computation of the MAP and MPM estimates in the previous section relies on knowledge of the joint distribution $\Pr(\boldsymbol{X}, \boldsymbol{Z})$ – where would we get such a distribution in practice? Typically, the answer is to *learn* this distribution from data, using, e.g., maximum likelihood estimation. Assume we have access to a large collection of image pairs, $\{(\boldsymbol{x}^{(p)}, \boldsymbol{z}^{(p)})\}$, where the $\boldsymbol{x}^{(p)}$ were produced by an extremely accurate scanner and the $\boldsymbol{z}^{(p)}$ by our scanner of interest. Further, suppose our joint distribution is parameterized by a vector $\boldsymbol{\theta}$ – in the simplest case there could be an element of $\boldsymbol{\theta}$ corresponding to every possible $(\boldsymbol{x}, \boldsymbol{z})$ pair, but more parsimonious representations are possible. The maximum likelihood estimate for $\boldsymbol{\theta}$ is given by:

$$\widehat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} \prod_p \Pr(\boldsymbol{X} = \boldsymbol{x}^{(p)}, \boldsymbol{Z} = \boldsymbol{z}^{(p)}; \boldsymbol{\theta}) \tag{1.3}$$

If we did not have access to the extremely accurate scanner and our dataset were comprised only of the $\{\boldsymbol{z}^{(p)}\}$, we could still perform maximum likelihood estimation by marginalizing

over $\boldsymbol{X}$:

$$\widehat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{p} \sum_{\boldsymbol{X}} \Pr(\boldsymbol{X} = \boldsymbol{x}, \boldsymbol{Z} = \boldsymbol{z}^{(p)}; \boldsymbol{\theta}) \tag{1.4}$$

In this case, more restrictive modeling assumptions would be necessary to ensure identifiability [2]. Note that learning in this partially observed case uses inference (marginalization) as a subroutine.

1.2.2 Exponential family distributions

In the previous section we considered a probability distribution with arbitrary parameterization. We now turn our focus to distributions of a particular form: *exponential family distributions* [76]. An exponential family distribution on a random vector $\boldsymbol{X} = [X_1 \cdots X_n]$ with parameters $\boldsymbol{\theta}$ can be written as follows:

$$\Pr(\boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(\boldsymbol{x}) - A(\boldsymbol{\theta})\right), \tag{1.5}$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}) \cdots \phi_m(\boldsymbol{x})]$ is a vector of sufficient statistics and $A(\boldsymbol{\theta})$, called the *log partition function*, ensures that $\Pr(\boldsymbol{X})$ is properly normalized:

$$A(\boldsymbol{\theta}) = \log \sum_{\boldsymbol{X}} \exp\left(\boldsymbol{\theta} \cdot \boldsymbol{\phi}(\boldsymbol{x})\right) \tag{1.6}$$

For continuous-valued variables, the summation is replaced by an integral.

It is common for each sufficient statistic to be a function of only a subset of the variables, $\boldsymbol{X}_u \subset \boldsymbol{X}$. To make this explicit, we will use the notation $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}_1) \cdots \phi_m(\boldsymbol{x}_m)]$, where $\boldsymbol{x}_u$ is the subvector of $\boldsymbol{x}$ corresponding to the domain of ϕ_u. It is also common to see this

alternate form:

$$\Pr(\boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z} f(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z} \prod_{u=1}^{m} f_u(\boldsymbol{x}_u; \boldsymbol{\theta}), \tag{1.7}$$

where $f(\boldsymbol{x})$ is the unnormalized joint distribution, the $f_u(\boldsymbol{x}_u; \boldsymbol{\theta})$ are nonnegative functions, and $Z = \sum_{\boldsymbol{x}} f(\boldsymbol{x}; \boldsymbol{\theta})$. Note that we use a bold font to emphasize that $\boldsymbol{x}_u$ is a subvector, unlike the single value x_s. In this form, the f_u are called *factors* or *potential functions* and Z is the partition function. Notationally, the dependence of f on $\boldsymbol{\theta}$ is often suppressed and it is understood that the factors, themselves, define the distribution. We also frequently use the letters u and v to index factors, and the letters s and t to index variables.

The class of exponential family distributions includes a wide variety of common distributions, including Gaussian, "multinoulli", Poisson, exponential, and many more. Further, as we will see in Section 1.4.1, it possesses some convenient analytical properties which make it possible to define generic inference algorithms capable of operating on any exponential family distribution.

1.2.3 Conditional independence

In Section 1.2.1 we considered a joint probability distribution $\Pr(\boldsymbol{X}, \boldsymbol{Z})$, where each variable had a domain size of 256 and there were mn variables – one for each pixel in an image. For any reasonably sized image, the product mn will be in the thousands, if not millions. Thus, the joint probability mass function (PMF) $\Pr(\boldsymbol{X}, \boldsymbol{Z})$ – a table of 256^{mn} numbers – is far too large to represent without further modeling assumptions.

Conditional independence is one modeling tool we can use to simplify a joint distribution. Given a joint distribution over random variables $X_1 \ldots X_n$, variables X_A and X_B are said to

be *conditionally independent given* X_C *iff*

$$\Pr(X_A \mid X_B, X_C) = \Pr(X_A \mid X_C). \tag{1.8}$$

This relationship is often abbreviated with the notation $X_A \perp\!\!\!\perp X_B \mid X_C$, and the same statement holds when X_A, X_B, and X_C are sets rather than individual variables.

By asserting conditional independence relationships in our modeling assumptions we can reduce the amount of space and computation needed to represent and perform inference on a joint probability distribution. To see this, consider the chain rule decomposition of a joint distribution over $X_1 \ldots X_n$:

$$\Pr(X_1 \ldots X_n) = \Pr(X_1) \Pr(X_2 \mid X_1) \ldots \Pr(X_n \mid X_1 \ldots X_{n-1}) \tag{1.9}$$

Each term is a conditional probability table (CPT) with size exponential in its scope. If we assert a large collection of conditional independencies, we can apply Equation 1.8 to each of the CPTs and reduce the size of the conditioning sets. The result is that even high-dimensional joint distributions are amenable to efficient representation and inference if the right conditional independence relationships hold (or appropriate approximating assumptions are made).

1.2.4 Factor graphs

Factor graphs [44], a specific type of graphical model, provide a compact, visual representation of the conditional independencies present in a distribution. Suppose we have a joint distribution over $\boldsymbol{X} = [X_1 \ldots X_n]$ in the form of Equation 1.7. The factor graph representation of $\Pr(\boldsymbol{X})$ is a bipartite graph, $G = (\boldsymbol{X}, \boldsymbol{F})$, consisting of *variable nodes* and *factor nodes*. The graph G includes n variable nodes, one for each $X_s \in \boldsymbol{X}$, and m factor nodes,

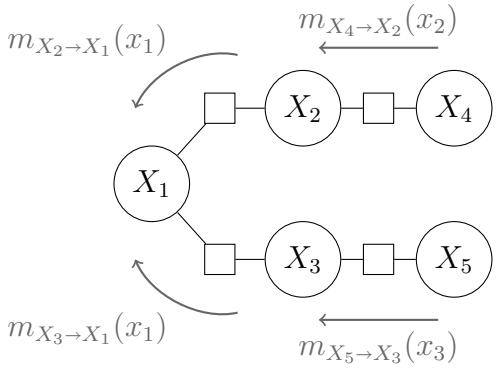


Figure 1.1: A sample factor graph. This graph corresponds to a distribution that factorizes as $\Pr(\boldsymbol{x}) \propto f(x_1, x_2)f(x_2, x_4)f(x_1, x_3)f(x_3, x_5)$. The annotated arrows (messages) surrounding the graph illustrate the variable elimination process used to compute the marginal $\Pr(x_1)$, as described in Section 1.3.

one for each factor f_u in Equation 1.7. Edges connect each factor node to the variables in its scope. In a common abuse of notation, we use the same symbols to represent each variable node and its associated variable X_s, and similarly for each factor node and its associated function f_u; it should be clear from the context what is intended. For convenience, let $\boldsymbol{X}_u$ denote the set of variables neighboring factor f_u, and $\boldsymbol{F}_s$ denote the set of factors neighboring variable X_s. An example of a factor graph is shown in Figure 1.1.

The connectivity of G captures the conditional independence structure of $\Pr(\boldsymbol{X})$ in the following sense: if all paths from X_A to X_B pass through X_C, then $X_A \perp\!\!\!\perp X_B \mid X_C$. As before, the above statement is also true when X_A, X_B, and X_C are sets of variables. For example, the graph in Figure 1.1 implies $X_3 \perp\!\!\!\perp X_2 \mid X_1$, but does *not* imply $X_5 \perp\!\!\!\perp X_1 \mid X_2$. This correspondence enables the development of efficient exact and approximate inference algorithms that leverage a distribution's conditional independencies through its factor graph structure [56, 44, 76].

1.3 Exact inference via variable elimination

As alluded to in the previous section, it is possible to leverage the structure of a factor graph to implement efficient algorithms for probabilistic inference. This section focuses on computing marginal probabilities and the partition function, but similar algorithms exist for

computing other quantities such as the mode.

1.3.1 Elimination as message-passing on a tree

Variable elimination [17, 18, 85] is a marginalization algorithm that proceeds by summing out variables one after another, taking advantage of conditional independencies whenever possible to reduce computational complexity. Consider computing the marginal probability of X_1 in the distribution described by Figure 1.1. The naïve approach simply sums over the joint PMF:

$$\Pr(X_1) \propto \sum_{X_2,X_3,X_4,X_5} f(X_1,X_2)f(X_2,X_4)f(X_1,X_3)f(X_3,X_5), \tag{1.10}$$

which has complexity $\mathcal{O}(d^5)$ when each variable has domain size d. The order in which the variables are summed out is called the *elimination order*. By choosing a good elimination order and rearranging the factor terms, we can significantly reduce the complexity:

$$\Pr(X_1) \propto \underbrace{\left(\sum_{X_2} f(X_1,X_2) \underbrace{\sum_{X_4} f(X_2,X_4)}_{m_{X_4\to X_2}}\right)}_{m_{X_2\to X_1}} \underbrace{\left(\sum_{X_3} f(X_1,X_3) \underbrace{\sum_{X_5} f(X_3,X_5)}_{m_{X_5\to X_3}}\right)}_{m_{X_3\to X_1}} \tag{1.11}$$

Note that by grouping the factor terms in this way, each summation ranges over a function of only two variables. Thus, this expression computes $\Pr(X_1)$ with computation only $\mathcal{O}(d^2)$.

The efficiency of Equation 1.11 hinges on the elimination order and the grouping of the factors. To simplify the bookkeeping involved, it is convenient to conceptualize the computation as *message-passing* between nodes of the distribution's factor graph. Each summation produces an intermediate result that can be viewed as a message being passed from the summed-out variable to the variable over which the resulting intermediate function is

defined. The braces underneath Equation 1.11 indicate the correspondence between sum operations and messages, and the same messages are illustrated alongside the factor graph in Figure 1.1.

In this way, the marginal distribution of a variable X_s in a tree-structured graph can be efficiently computed by rooting the graph at X_s and passing messages from the leaves to the root. A subsequent pass of messages from the root back out to the leaves is sufficient to compute the marginals of the remaining variables, and the partition function can be computed simply by summing over the final variable rather than normalizing. As we will see in Section 1.4.1, the message-passing view of the computation can be further decomposed into messages from factors to variables and from variables to factors; this more fine-grained view is convenient when some factors are defined over more than two variables.

1.3.2 Junction tree: variable elimination on a hyper-tree

Just as marginalization of a tree-structured distribution can be viewed as message-passing over its factor graph, marginalization of *non*-tree-structured distributions can be viewed as message-passing over a specific type of hyper-graph called a *junction tree*. A junction tree is a tree-structured graph in which each node is associated with a set of variables. The set of variables associated with a node is termed its *scope*, and the intersection between scopes of adjacent nodes is called a *separator set*. A junction tree must satisfy the following two conditions:

- For each factor f_u, the domain $\boldsymbol{X}_u$ is fully contained in the scope of at least one node.
- All separator sets must be non-empty.
- For each variable X_s, the set of nodes whose scope contains X_s forms a single connected component.

(a) (b)

Figure 1.2: (a) A "loopy" factor graph. (b) A junction tree consistent with the factor graph in (a). Variable elimination on this distribution can be viewed as message-passing over this junction tree structure.

Given a junction tree for a distribution, the marginal distribution of a variable or set of variables can be computed via message-passing analogously to the tree-structured case. As an example, consider the factor graph shown in Figure 1.2a, which results from adding a single factor to the graph in Figure 1.1 such that it is no longer a tree.

To compute the marginal $\Pr(x_1)$, using the same elimination order as before, we must perform the following computation:

$$\Pr(X_1) \propto \sum_{X_2} f(x_1, x_2) \sum_{X_3} f(x_1, x_3) \underbrace{\sum_{X_4} f(x_2, x_4) \underbrace{\sum_{X_5} f(x_3, x_5) f(x_4, x_5)}_{m_{345\to234}} \Bigg)}_{m_{234\to123}} \tag{1.12}$$

The summations result in intermediate functions that can be viewed as messages being passed between nodes of the junction tree, as indicated by the underbraces. Marginalization via message-passing on a junction tree has time complexity exponential in the size of the largest scope and space complexity exponential in the size of the largest separator set [17, 18, 38]. The size of the largest region minus one is known as the induced width of the graph, and the minimal induced width across all possible elimination orderings is known as the tree-width [17, 18]. Finding an elimination order that results in a graph of low induced width is an important step in efficient, exact inference [39, 25]. For some probability distributions, however, the tree-width is simply too large for exact inference to be tractable; in these cases

we can consider inexact alternatives, as described in the next section.

1.4 Approximate inference via variational message-passing

The calculus of variations provides a general framework for exact and approximate inference [76]. This section first introduces the framework, and then shows how several specific instantiations lead to well known approximate inference algorithms.

1.4.1 Variational inference: a birds-eye view

At the heart of the variational inference framework is the following formulation of the log partition function [76]:

$$A(\boldsymbol{\theta}) = \log Z = \max_{\boldsymbol{\mu} \in \mathcal{M}} \mathbb{E}_{\boldsymbol{\mu}} \left[\log f(\boldsymbol{X}) \right] + \mathbb{H}(\boldsymbol{\mu}), \tag{1.13}$$

where $\boldsymbol{\mu}$ is a vector of *mean parameters*, $\mathbb{H}(\boldsymbol{\mu})$ is the entropy of the exponential family distribution with mean parameters $\boldsymbol{\mu}$, and $\mathcal{M}$, known as the *marginal polytope*, is the set of all valid mean parameter vectors. This relationship, which arises as the conjugate dual of $A(\boldsymbol{\theta})$, offers an alternate way to compute the partition function via optimization rather than summation. Further, the optimum of Equation 1.13 is achieved when $\boldsymbol{\mu}$ corresponds exactly to the marginal probabilities of the sufficient statistics of $\Pr(\boldsymbol{X})$. Thus, by optimizing Equation 1.13 we can compute both marginals and the partition function.

Unfortunately, the optimization of Equation 1.13 is more difficult than it may seem at first glance. The marginal polytope is often extremely complex, requiring an intractably large

number of linear constraints to characterize its boundary, and the entropy function generally has no closed-form expression in terms of the mean parameters [76]. These difficulties lead one to approximate both $\mathcal{M}$ and $\mathbb{H}(\boldsymbol{\mu})$, and different choices of approximation lead to different approximate inference algorithms.

The following subsections introduce several of the most common variational inference algorithms. To keep the notation consistent throughout, assume that in each case we begin with a joint distribution $\Pr(\boldsymbol{X})$ over discrete variables $X_1 \cdots X_n$ that is consistent with the factor graph $G = (\{X_s\}_{s=1}^n, \{f_u\}_{u=1}^m)$, i.e.,

$$\Pr(\boldsymbol{X}) \propto \prod_{u=1}^{m} f_u(\boldsymbol{X}_u).$$

Naïve mean field

Naïve mean field inference [76, 37] replaces the constraint set $\mathcal{M}$ with an inner bound $\mathcal{M}^{MF} \subseteq \mathcal{M}$ corresponding to the subset of mean parameter vectors consistent with fully independent distributions. This approximation makes the constraint set tractable – it reduces to a set of decoupled normalization constraints on each variable's mean parameters – and also guarantees that the entropy term is computable in closed form:

$$\mathbb{H}(\boldsymbol{\mu}) = -\sum_{s=1}^{n} \sum_{X_s} \mu_{s;x_s} \log \mu_{s;x_s} \tag{1.14}$$

where $\mu_{s;x_s}$ is the mean parameter corresponding to the event $X_s = x_s$. Thus, naïve mean field solves the following optimization:

$$A^{MF}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}^{MF}} \mathbb{E}_{\boldsymbol{\mu}} \left[\log f(\boldsymbol{X})\right] + \mathbb{H}(\boldsymbol{\mu}), \tag{1.15}$$

Typically, this optimization is performed via coordinate ascent. Constructing the Lagrangian, setting the gradient with respect to $\mu_s(x_s)$ equal to zero, and solving yields the following update:

$$\mu_s(x_s) \propto \prod_{f_u \in \boldsymbol{F}_s} \exp\left(\sum_{\boldsymbol{X}_u \setminus X_s} \log f_u(\boldsymbol{x}_u) \prod_{x_j \in \boldsymbol{X}_u \setminus X_s} \mu_j(x_j) \right) \tag{1.16}$$

Naïve mean field is typically implemented by repeatedly iterating over the variables in sequence, updating each in turn according to Equation 1.16 until reaching convergence. Convergence is guaranteed, but because the constraint set $\mathcal{M}^{MF}$ is non-convex it will converge to a local optimum.

At convergence, the mean parameters $\boldsymbol{\mu}$ can be used as estimates of the marginals of $\Pr(\boldsymbol{X})$. Approximate marginals computed via variational inference are called *beliefs*, often written as:

$$b_s(x_s) = \mu_s(x_s) \tag{1.17}$$

$$b_u(\boldsymbol{x}_u) = \prod_{X_s \in \boldsymbol{X}_u} \mu_s(x_s) \tag{1.18}$$

For the sake of consistency with other message-passing algorithms on factor graphs, we can decompose the update equation into two steps. The resulting algorithm includes two types of message: *variable messages*, which pass information from a variable node to a factor node, and *factor messages*, which pass information from a factor to a variable. The corresponding

updates are as follows[1]:

$$m_{X_s \to f_u}(x_s) \propto \prod_{f_v \in \boldsymbol{F}_s} \exp\left(m_{f_v \to X_s}(x_s)\right) \tag{1.19}$$

$$m_{f_u \to X_s}(x_s) \propto \sum_{\boldsymbol{X}_u \setminus X_s} \log f_u(\boldsymbol{x}_u) \prod_{X_t \in \boldsymbol{X}_u \setminus X_s} m_{X_t \to f_u}(x_t) \tag{1.20}$$

$$b_s(x_s) \propto \prod_{f_u \in \boldsymbol{F}_s} \exp\left(m_{f_u \to X_s}(x_s)\right) \tag{1.21}$$

Note that the only difference between the mean field variational formulation of the log partition function (Equation 1.15) and the exact variational form (Equation 1.13) is that mean field optimizes over a tractable subset of the constraint set. As a result, mean field produces a lower bound on the log partition function: $A^{MF}(\boldsymbol{\theta}) \leq A(\boldsymbol{\theta})$. This lower bound can be computed from the current beliefs at any stage of the algorithm (even prior to convergence) as follows:

$$A^{MF}(\boldsymbol{\theta}) = \sum_{u=1}^{m} \sum_{\boldsymbol{X}_u} \log f_u(\boldsymbol{x}_u) \prod_{X_s \in \boldsymbol{X}_u} b_s(x_s) - \sum_{s=1}^{n} \sum_{X_s} b_s(x_s) \log b_s(x_s). \tag{1.22}$$

1.4.2 Loopy belief propagation

Whereas mean field optimizes over a tractable subset of $\mathcal{M}$, loopy belief propagation (BP) [76, 83] optimizes over a tractable superset, $\mathcal{M}^L$. The true marginal polytope contains only mean vectors that are *consistent* in the sense that there exists some joint exponential family distribution with the corresponding marginals. The set $\mathcal{M}^L$, called the *local polytope*, enforces a weaker notion of local consistency, requiring only that the mean vectors $\boldsymbol{\mu}$ satisfy the

[1]Note that message from a variable to a neighboring factor is just the variable's belief. We define them separately here to draw a parallel to later algorithms where they will not be the same.

following:

$$\sum_{\boldsymbol{X}_u \setminus X_s} \mu_u(\boldsymbol{x}_u) = \mu_s(x_s) \qquad \forall u \in \{1 \dots m\}, \forall X_s \in \boldsymbol{X}_u. \tag{1.23}$$

In other words, the beliefs of any two factors must be locally consistent with respect to their marginal beliefs on each shared variable. When the factor graph corresponding to $\Pr(\boldsymbol{X})$ is a tree, these constraints (along with normalization constraints) are sufficient to exactly characterize $\mathcal{M}$; when the factor graph is not a tree, $\mathcal{M}^L$ is an outer approximation to $\mathcal{M}$. Since $\mathcal{M}^L$ contains mean parameters that do not correspond to the marginals of any valid joint distribution, the beliefs produced by BP are sometimes called *pseudomarginals*.

This approximation of $\mathcal{M}$ does not guarantee a tractable form of the entropy term. In fact, for vectors $\boldsymbol{\mu} \in \mathcal{M}^L$ that do not correspond to a valid joint distribution, the concept of entropy is not even well defined. Thus, the entropy term must be approximated separately. Tree-structured distributions have the following convenient representation as a function of $\boldsymbol{\mu}$:

$$\Pr(\boldsymbol{X} = \boldsymbol{x}) = \prod_{X_s \in \boldsymbol{X}} \mu_s(x_s) \prod_{f_u \in \boldsymbol{F}} \frac{\mu_u(\boldsymbol{x}_u)}{\prod_{X_s \in \boldsymbol{X}_u} \mu_s(x_s)} \tag{1.24}$$

The entropy of a distribution with this form is easily computable. BP replaces the exact entropy $\mathbb{H}(\boldsymbol{\mu})$ with the Bethe entropy, $\mathbb{H}^{Bethe}(\boldsymbol{\mu})$, which simply assumes the factorization structure of Equation 1.24 even when the graph is not a tree:

$$\mathbb{H}^{Bethe}(\boldsymbol{\mu}) = -\sum_{u=1}^{m} \sum_{\boldsymbol{X}_u} b_u(\boldsymbol{x}_u) \log b_u(\boldsymbol{x}_u) + \sum_{s=1}^{n} (1 - |F_s|) \sum_{X_s} b_s(x_s) \log b_s(x_s). \tag{1.25}$$

The resulting variational form is as follows:

$$A^{Bethe}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}^L} \mathbb{E}_{\boldsymbol{\mu}} \left[\log f(\boldsymbol{X})\right] + \mathbb{H}^{Bethe}(\boldsymbol{\mu}). \tag{1.26}$$

BP optimizes the above objective using fixed-point iteration, where each of the updates can be viewed as a message passed between two adjacent nodes of the factor graph. The message updates for BP are as follows:

$$m_{f_u \to X_s}(x_s) \propto \sum_{\boldsymbol{X}_u \setminus X_s} \left[f_u(\boldsymbol{x}_u) \prod_{X_t \in \boldsymbol{X}_u \setminus X_s} m_{x_t \to f_u}(x_t) \right] \tag{1.27}$$

$$m_{X_s \to f_u}(x_s) \propto \prod_{f_v \in \boldsymbol{F}_s \setminus f_u} m_{f_v \to X_s}(x_s) \tag{1.28}$$

On tree-structured graphs, passing these messages inward from leaves to the root performs exact variable elimination as described in Section 1.3.

Beliefs can be computed as follows:

$$b_s(x_s) \propto \prod_{f_u \in \boldsymbol{F}_s} m_{f_u \to X_s}(x_s) \tag{1.29}$$

$$b_u(\boldsymbol{x}_u) \propto f_u(\boldsymbol{x}_u) \prod_{X_s \in \boldsymbol{X}_u} m_{X_s \to f_u}(x_s) \tag{1.30}$$

Unlike mean field, loopy BP does not provide a bound on the log partition function. However, one can evaluate Equation 1.26 at a fixed point of the algorithm to produce an estimate:

$$\begin{aligned} A^{BP}(\boldsymbol{\theta}) = &\sum_{u=1}^{m} \sum_{\boldsymbol{X}_u} \log f_u(\boldsymbol{x}_u) b_u(\boldsymbol{x}_u) - \sum_{u=1}^{m} \sum_{\boldsymbol{X}_u} b_u(\boldsymbol{x}_u) \log b_u(\boldsymbol{x}_u) \\ &+ \sum_{s=1}^{n} (1 - |\boldsymbol{F}_s|) \sum_{X_s} b_s(x_s) \log b_s(x_s) \end{aligned} \tag{1.31}$$

This estimate is exact for tree-structured graphs and quite accurate for many non-tree graphs, as well [54, 33].

In general, on non-tree graphs the fixed-point iteration of Equations 1.27-1.28 are not guaranteed to converge. Convergence is assessed either in terms of the log-partition function estimate or the message values. In either case, if the value(s) change by less than a prespec-

ified tolerance between iterations we consider the algorithm to have converged. To make convergence more likely, some form of *damped* updates are often used in place of Equations 1.27-1.28. For example [31]:

$$\log m^{t+1}_{f_u \to X_s}(x_s) = \log m^{t}_{f_u \to X_s}(x_s) + \epsilon \left[\log m^{Full}_{f_u \to X_s}(x_s) - \log m^{t}_{f_u \to X_s}(x_s) \right], \tag{1.32}$$

where m^t denotes the message value at iteration t, m^{Full} denotes the "full", undamped message update as defined by Equation 1.27, and ϵ is a step size parameter. The message-passing *schedule* – the order in which one cycles through message updates – can also affect the likelihood of convergence [70].

1.4.3 Tree-reweighted belief propagation

Tree-reweighted belief propagation (TRW) [76, 75] optimizes over the same constraint set as BP but uses a different approximation of the entropy. In particular, TRW approximates the entropy of a distribution with a convex combination of entropies of tree-structured subgraphs, which is an upper bound on the true entropy. The combination of a convex outer bound on $\mathcal{M}$ and a convex upper bound on $H(\boldsymbol{\mu})$ means that the TRW objective is convex and its optimum provides an upper bound on the true log-partition function.

Thus, TRW optimizes the following objective:

$$A^{TRW}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}^L} \mathbb{E}_{\boldsymbol{\mu}} \left[\log f(\boldsymbol{X}) \right] + \mathbb{H}^{TRW}(\boldsymbol{\mu}). \tag{1.33}$$

The TRW entropy can be written as follows:

$$\begin{aligned}\mathbb{H}^{TRW}(\boldsymbol{\mu}) = &\sum_{u=1}^{m} \rho_u \sum_{\boldsymbol{X}_u} b_u(\boldsymbol{x}_u) \log b_u(\boldsymbol{x}_u) \\ &+ \sum_{s=1}^{n} \left(1 - \sum_{f_u \in \boldsymbol{F}_s} \rho_u\right) \sum_{X_s} b_s(x_s) \log b_s(x_s),\end{aligned} \tag{1.34}$$

where the $\boldsymbol{\rho} = \{\rho_u\}_{u=1}^m$ are factor weights associated with the convex combination of subtrees. Each subtree has an associated tree weight, and a factor weight ρ_u is equal to the sum of the tree weights corresponding to trees that include factor f_u. Writing the entropy in terms of factor weights rather than tree weights makes it possible to efficiently perform computations over a very large, but implicit, set of trees.

Again constructing the Lagrangian and setting gradients equal to zero, we recover the following message-passing and belief update equations:

$$m_{f_u \to X_s}(x_s) \propto \sum_{\boldsymbol{X}_u \setminus X_s} f_u(\boldsymbol{x}_u)^{1/\rho_u} \prod_{X_t \in \boldsymbol{X}_u \setminus X_s} m_{X_t \to f_u}(x_t) \tag{1.35}$$

$$m_{X_s \to f_u}(x_s) \propto \frac{\prod_{f_v \in \boldsymbol{F}_s} m_{f_v \to X_s}(x_s)^{\rho_v}}{m_{f_u \to x_s}(x_s)} \tag{1.36}$$

$$b_s(x_s) \propto \prod_{f_u \in \boldsymbol{F}_s} m_{f_u \to X_s}(x_s)^{\rho_u} \tag{1.37}$$

$$b_u(\boldsymbol{x}_u) \propto f_u(\boldsymbol{x}_u)^{1/\rho_u} \prod_{X_s \in \boldsymbol{X}_u} m_{X_s \to f_u}(x_s). \tag{1.38}$$

Plugging the entropy approximation into Equation 1.33 gives the following form for the TRW upper bound on the log-partition function:

$$\begin{aligned}A^{TRW}(\boldsymbol{\theta}) = &\sum_{u=1}^{m} \sum_{\boldsymbol{X}_u} \log f_u(\boldsymbol{x}_u) b_u(\boldsymbol{x}_u) - \sum_{u=1}^{m} \rho_u \sum_{\boldsymbol{X}_u} b_u(\boldsymbol{x}_u) \log b_u(\boldsymbol{x}_u) \\ &+ \sum_{s=1}^{n} \left(1 - \sum_{f_u \in \boldsymbol{F}_s} \rho_u\right) \sum_{X_s} b_s(x_s) \log b_s(x_s).\end{aligned} \tag{1.39}$$

Note that this is only guaranteed to be an upper bound for beliefs corresponding to the optimum of Equation 1.33, i.e., after message-passing has reached convergence.

The TRW bound is a function of the factor weight vector, $\boldsymbol{\rho}$. Ideally, one would like to choose $\boldsymbol{\rho}$ to achieve the tightest possible upper bound. Since Equation 1.39 is convex in $\boldsymbol{\rho}$ and the space of valid $\boldsymbol{\rho}$ is convex, it is possible to optimize over $\boldsymbol{\rho}$ in an outer loop via a conditional gradient algorithm [75].

1.4.4 Generalized belief propagation

Generalized belief propagation (GBP) [76, 83, 82] refers to a family of algorithms similar to loopy BP but which use tighter outer bounds on $\mathcal{M}$ and more complex approximations to the entropy function. A GBP approximation is specified by a set of *regions* and *counting numbers*.

A region is simply a subset of variables, $\boldsymbol{X}_\alpha \subseteq \boldsymbol{X}$. The set of regions, $\mathcal{R}$, determines the constraint set $\mathcal{M}^{GBP}$: whereas BP requires that any two factor beliefs agree on the marginal beliefs of individual variables in their intersection, GBP requires that any two region beliefs agree on the belief of their entire intersection. These constraints can be summarized as follows:

$$\sum_{\boldsymbol{X}_\alpha \setminus \boldsymbol{X}_\beta} b_\alpha(\boldsymbol{x}_\alpha) = \sum_{\boldsymbol{X}_\beta \setminus \boldsymbol{X}_\alpha} b_\beta(\boldsymbol{x}_\beta) \qquad \forall \alpha \in \mathcal{R}, \beta \in \mathcal{R}. \tag{1.40}$$

There are many ways to select regions for GBP. One simple method is to construct a junction graph (which satisfies all the same properties as a junction tree but need not be tree-structured) and create one region for each node and separator set [82]. The set of all regions forms a directed *region graph* in which ancestry is determined by set inclusion: for a given region α, its ancestors $an(\alpha) = \{\gamma \in \mathcal{R} : \boldsymbol{X}_\alpha \subset \boldsymbol{X}_\gamma\}$ are the regions whose scopes are

supersets of α's, and its descendants $de(\alpha) = \{\beta \in \mathcal{R} : \boldsymbol{X}_\alpha \supset \boldsymbol{X}_\beta\}$ are the regions whose scopes are subsets.

The counting numbers, one for each region, specify the entropy approximation:

$$\mathbb{H}^{GBP}(\boldsymbol{\mu}) = \sum_{\alpha \in \mathcal{R}} c_\alpha \mathbb{H}(\boldsymbol{\mu}_\alpha), \tag{1.41}$$

where c_α is the counting number for region α. Different sets of counting numbers result in different entropy approximations and, correspondingly, different GBP algorithms. In principle one can set them arbitrarily, but it is generally recommended that they satisfy the following criteria [82]:

$$\begin{aligned} \sum_{\alpha \in R(f_u)} c_\alpha &= 1 \qquad u = 1, \ldots, m \\ \sum_{\alpha \in R(X_s)} c_\alpha &= 1 \qquad s = 1, \ldots, n \end{aligned} \tag{1.42}$$

where $R(f_u)$ is the set of regions containing factor f_u and $R(X_s)$ is the set of regions whose scope contains variable X_s.

Together, the two approximations result in the GBP form of the log-partition function:

$$A^{GBP}(\boldsymbol{\theta}) = \max_{\boldsymbol{\mu} \in \mathcal{M}^{GBP}} \mathbb{E}_{\boldsymbol{\mu}} \left[\log f(\boldsymbol{X})\right] + \mathbb{H}^{GBP}(\boldsymbol{\mu}), \tag{1.43}$$

On a junction tree, Equation 1.43 is equivalent to Equation 1.13 and GBP performs exact inference. Also note that BP can be viewed as a special case of GBP, where the region set is $\mathcal{R} = \boldsymbol{F} \cup \boldsymbol{X}$ and the counting numbers are $c_u = 1$ for all factors f_u and $c_s = 1 - |\boldsymbol{F}_s|$ for all variables X_s. Intuitively, choosing larger regions can be thought of as interpolating between Equation 1.26 and Equation 1.13.

As with the previous variational formulations, we can solve the constrained optimization in

Equation 1.43 via the method of Lagrange multipliers. This results in fixed-point updates that can be viewed as messages passing from parent to child regions. When regions are chosen based on a junction graph as described above, the message and belief update equations are as follows [78]:

$$m_{\alpha\to\beta}(x_\beta) \propto \frac{\sum_{\boldsymbol{X}_\alpha \setminus \boldsymbol{X}_\beta} b_\alpha(\boldsymbol{x}_\alpha)}{b_\beta(\boldsymbol{x}_\beta)} m^{old}_{\alpha\to\beta}(\boldsymbol{x}_\beta) \tag{1.44}$$

$$b_\alpha(x_\alpha) \propto \prod_{u\in\alpha} f_u(\boldsymbol{x}_u) \prod_{\gamma\in an(\Delta_\alpha)\setminus\Delta_\alpha,\ \beta\in\Delta_\alpha} m_{\gamma\to\beta}(\boldsymbol{x}_\beta) \tag{1.45}$$

where $\Delta_\alpha = \alpha \cup de(\alpha)$.

1.4.5 Mini-bucket and weighted mini-bucket

Mini-bucket (MB) [19] is an approximate marginalization algorithm based on direct approximation of the variable elimination procedure in its primal form, rather than the dual formulation used by the algorithms presented in earlier subsections. Recall the loopy factor graph of Figure 1.2. Exact variable elimination has complexity $\mathcal{O}(d^3)$, due to computations like the following:

$$\sum_{X_5} f(x_3, x_5) f(x_4, x_5) \tag{1.46}$$

which sums over a function of three variables. MB instead computes a bound on this quantity using the following inequalities:

$$\begin{aligned} \sum_{X_5^1} f(x_3, x_5^1) \min_{X_5^2} f(x_4, x_5^2) &\le \sum_{X_5} f(x_3, x_5) f(x_4, x_5) \\ &\le \sum_{X_5^1} f(x_3, x_5^1) \max_{X_5^2} f(x_4, x_5^2), \end{aligned} \tag{1.47}$$

where the variable X_5 has been "split" into two distinct replicates, X_5^1 and X_5^2. Note that the upper and lower bounds can be computed in $\mathcal{O}(d^2)$, as opposed to $\mathcal{O}(d^3)$ for the exact computation. More generally, MB operates by splitting variables whenever necessary to avoid elimination operations over functions with scopes above a prespecified limit called the *ibound*. As a result, the each elimination in complexity of MB is exponential in the *ibound* rather than exponential in the induced width.

The inequalities of Equation 1.47 hold whenever all but one replicate of a variable is eliminated using max or min and the final copy is eliminated via summation. This non-iterative, primal form of this bound has some advantages over the dual formulations of the previous subsections. Since the bound can be computed deterministically in a single pass, convergence is not an issue. Further, since it does not depend on the exact solution of a fixed-point iteration algorithm, sample-based approximations of the bound are more amenable to analysis than, e.g., TRW.

Weighted mini-bucket (WMB) [48] is a recent generalization of MB that replaces the inequalities in Equation 1.47 with Hölder's inequality [30]. The upper bound in Equation 1.47 becomes:

$$\sum_{X_5} f(x_3, x_5) f(x_4, x_5) \leq \sum_{X_5^1}^{w_1} f(x_3, x_5^1) \sum_{X_5^2}^{w_2} f(x_4, x_5^2), \tag{1.48}$$

where $w_1 + w_2 = 1, w_1 > 0, w_2 > 0$, and

$$\sum_{X}^{w} f(x) \equiv \left(\sum_{X} f(x)^{1/w} \right)^{w} \tag{1.49}$$

is a weighted summation operator.

More generally, Hölder's inequality states that

$$\sum_X \prod_i f_i(x) \leq \prod_i \sum_X^{w_i} f_i(x), \tag{1.50}$$

where $\sum_i w_i = 1$ and $\forall i \ w_i > 0$. A related bound exists with the reverse inequality sign, but we do not focus on it here. The resulting upper bound on the log-partition function, assuming elimination order $o = [X_1 \dots X_n]$, can be written as follows:

$$A^{WMB}(\boldsymbol{\theta}) = \log \left(\sum_{\bar{X}_n}^{\bar{w}_{\bar{n}}} \cdots \sum_{\bar{X}_1}^{\bar{w}_1} \prod_{u=1}^{m} \bar{f}_u(\bar{\boldsymbol{x}}_u) \right), \tag{1.51}$$

where $\bar{\boldsymbol{X}}$ is an expanded variable set including replicates $X_s^1 \dots X_s^{R_s}$ of each original variable X_s, $\bar{\boldsymbol{f}}$ is the corresponding set of factors defined over $\bar{\boldsymbol{X}}$, and $\bar{\boldsymbol{w}}$ is a vector of positive weights such that, for each variable X_s, the weights corresponding to its replicates sum to one. As in TRW, the weights $\bar{\boldsymbol{w}}$ can be optimized to tighten the upper bound. The extended variable set $\bar{\boldsymbol{X}}$ is created as in mini-bucket, first choosing an elimination order, o, and then splitting a variable into replicates whenever the function to be summed over exceeds the *ibound*.

As noted in [19, 48], MB and WMB elimination can be viewed as message-passing on a junction tree. Let $\bar{o}$ be the trivial extension of o to the extended variable set $\bar{\boldsymbol{X}}$, where all replicates of each variable are sequentially eliminated in the same order as in o. Let k be a linear index into $\bar{o}$. Then the eliminations in Equation 1.51 correspond to message computations on a junction tree over $\bar{\boldsymbol{X}}$ with cliques $\{c_k\}_{k=1}^{\bar{n}}$, where c_k includes $\bar{X}_k$ and all of its neighbors that follow it in order $\bar{o}$. The messages are computed as follows, in a single pass from the leaves to the root:

$$m_{k \to l}(\bar{\boldsymbol{x}}_{c_l}) = \left(\sum_{\bar{x}_k} \left(\bar{f}_{c_k}(\boldsymbol{x}_{c_k}) \prod_{j : k \in \bar{pa}(j)} m_{j \to k}(\boldsymbol{x}_{s_{jk}}) \right)^{1/\bar{w}_k} \right)^{\bar{w}_k}, \tag{1.52}$$

where $s_{kl} = c_k \cap c_l$ is the *separator set* between cliques k and l.

MB can be recovered as a limiting case of WMB, where the max and min operators are replaced by weighted summations with weights set to $0+$ and $0-$, respectively. WMB, and thus MB, can also be formulated in the same variational framework as the previous algorithms, but in this work we focus only on their convenient primal form.

Chapter 2

Computing Track Marginals in the Track-Oriented MHT

Multi-target tracking is a core component of many real-world sensing systems in applications including air defense, autonomous navigation, video surveillance, and robotics [47, 22, 62, 66]. Due to imperfect or low-information sensors, many such systems must handle an uncertain correspondence between sensor observations and real-world targets. Resolving this uncertainty is known as the data association problem, and it is the fundamental reason why optimal estimators for multi-target state estimation are generally intractable [53].

Despite its intractability, the problem's practical importance has motivated numerous algorithms based on simplifying assumptions, approximations, and computational shortcuts. The track-oriented multiple hypothesis tracker (TOMHT) [45], originally proposed in 1990, is one such algorithm. The TOMHT is generally considered to be among the most effective approaches for tracking in cluttered environments given medium to high computational resources [10, 60, 71]. Broadly speaking, the TOMHT works by implicitly representing all

possible joint data associations within a temporal sliding window, postponing hard decisions until observations fall behind the window's trailing edge. At each step the TOMHT reports the most likely data association for all observations processed up to that point, placing it in the category of MAP-based algorithms for data association.

Although the TOMHT makes hard data association decisions outside the sliding window, its internal data structures define a full posterior distribution over the space of possible associations within the window. The MAP estimate typically computed by the TOMHT is only one of several possible summaries of this posterior. The distribution's entropy, for instance, conveys a measure of uncertainty that could be used to dynamically adjust the length of the sliding window. Marginal probabilities of individual tracks could be useful as part of a real-time display for the tracker operator or, as we explore in Chapter 3, as a component of an expectation maximization (EM) algorithm for estimating system parameters.

In this chapter we provide a novel formulation of the TOMHT in the language of graphical models. This formulation enables the use of efficient, general purpose algorithms for approximating intractable probabilistic queries. In particular, we focus on the computation of track marginal probabilities. Even in the TOMHT's pruned data association space, marginalization is intractable. To this end we first identify two families of approximate estimators for the track marginals. The first technique follows a well-known approach based on exact computation of the k-best data association hypotheses. The second uses a novel application of variational message-passing algorithms for marginalization. We conduct an empirical comparison of these estimators in terms of their accuracy on simulated sensor data, and show that while the k-best approach can be very accurate in small scenarios, its accuracy degrades quickly as the problem size increases. The message-passing algorithms, on the other hand, may be less accurate on small problems but scale more robustly.

2.1 Introduction to multi-target tracking

We begin by introducing the standard probabilistic model used in multi-target tracking. We also review some of the most common algorithms for multi-target tracking and discuss the traditional development of the TOMHT.

2.1.1 Generative probabilistic model

Multitarget tracking aims to recover trajectories of targets of interest from sets of observations. For example, a radar sensing system returns a set of observations, each corresponding to a possible airplane detection, every time it scans a portion of the sky. Given several sets of observations (each set is sometimes called a *scan*), our goal would be to determine the number, current locations, and past trajectories of any planes in the region. This task is often approached from a probabilistic perspective, in which we condition on a set of scans to produce a posterior distribution on our quantities of interest. This subsection introduces the standard modeling assumptions made in this probabilistic approach [11].

Targets are modeled as points x in a bounded state space $\mathcal{X} \subset \mathbb{R}^{d_x}$, where d_x denotes the dimensionality of the target state space. The semantics of $\mathcal{X}$ will vary depending on the particular tracking application, but it typically includes components corresponding to the target's kinematic state such as position, velocity, and acceleration. Target states are assumed to evolve in discrete time according to a first-order Markov dynamics model. That is, we specify a conditional probability density $f_d(x^{k+1} \mid x^k)$ on a target's state at time $k+1$ given its state at time k. The density f_d is variously referred to as the dynamics model, motion model, or state transition function.

In a typical multitarget tracking application, a sensor scans the entire surveillance region (e.g.

a portion of the sky or a frame of a video) at each time and then returns a collection of real-valued detections. Each detection can be considered a potential target – in practice only some unknown fraction of the detections actually correspond to targets and the remainder are false alarms, also known as *clutter*. Thus, at a given time step k each target *may* be detected by the sensor, yielding an observation z in the bounded observation space $\mathcal{Z} \subset \mathbb{R}^{d_z}$, where d_z is the dimensionality of the observation z. Note that the observation domain $\mathcal{Z}$ need not be the same as the target state domain. For example, when the state vectors represent both position and velocity the sensor may only observe its position. Each observation corresponding to an actual target is distributed according to the conditional density $f_o(z^k \mid x^k)$, often called the observation model. Observations not corresponding to an actual target (clutter) are distributed uniformly throughout $\mathcal{Z}$.

The choice of appropriate dynamics and observation models is application-dependent, but due to its analytic tractability the linear Gaussian model is a common choice:

$$f_d(x^{k+1} \mid x^k) = \mathcal{N}(Ax^k, Q) \tag{2.1}$$

$$f_o(z^k \mid x^k) = \mathcal{N}(Hx^k, R), \tag{2.2}$$

where A is a $d_x \times d_x$ transition matrix, H is the $d_z \times d_x$ observation matrix, Q is the $d_x \times d_x$ process noise covariance matrix, and R is the $d_z \times d_z$ observation noise covariance matrix. Use of this model results in the well-known Kalman filtering and smoothing recursions for single-target tracking, which is an important subroutine in the multi-target case [67].

We also make the standard assumptions regarding missed detections and false alarms (clutter), as follows. At each time step k, each existing target is detected with probability p_D. The sensor generates n_ϕ clutter observations not corresponding to any target, where n_ϕ is assumed to be Poisson distributed with positive rate parameter λ_ϕ.

The number of targets in the surveillance region can change over time. At each time step k,

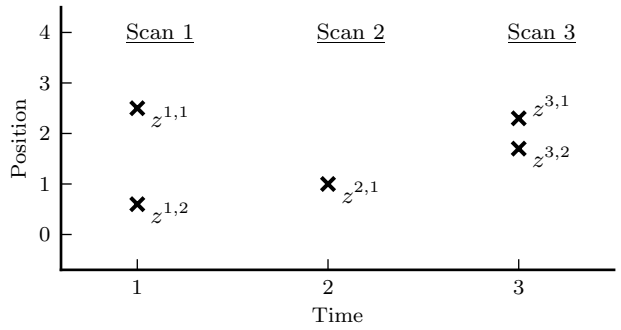


Figure 2.1: A small example scenario with three scans of data. Observations $z^{1,1}$ and $z^{3,2}$ are false alarms, but when they are processed by the tracker there are no labels identifying them as such.

new targets may enter the surveillance region and some or all of the preexisting targets may leave; we call these events *track births* and *track deaths*, respectively. At a given time, we model the number of track births as a Poisson random variable with rate parameter λ_ν and assume that each preexisting track has probability p_γ of dying. For simplicity we assume that track births and deaths are distributed uniformly over the target state space $\mathcal{X}$.

2.1.2 Data association and the multiple hypothesis tracker

The difficulty of multi-target tracking stems from the lack of an observed correspondence between observations and targets. A common strategy is to take a data augmentation approach, introducing auxiliary variables that represent the (unobserved) sources of the observations [53, 61, 45]. In this subsection we introduce these auxiliary variables and formulate the MAP data association problem as an optimization over their posterior.

At each time step, the tracker receives as input all newly generated observations – both actual detections and clutter – grouped together into a single set called a *scan*. Let m_k denote the number of observations in the k^{th} scan. The scene in Figure 2.1, for example, shows three scans: $\{z^{1,1}, z^{1,2}\}$, $\{z^{2,1}\}$, and $\{z^{3,1}, z^{3,2}\}$, where $m_1 = 2, m_2 = 1, m_3 = 2$, and we use the notation $z^{k,j}$ to represent the j^{th} observation in scan k. In this example the observation space $\mathcal{Z}$ is 1-dimensional, which allows for clear 2-dimensional plotting as a function of time. There are real applications with 1-dimensional observation spaces (e.g.

bearings-only tracking), but in general $\mathcal{Z}$ can be higher dimensional. Note that while the first index (scan number) conveys a meaningful temporal ordering of the observations, the second (within-scan) index is arbitrary – it does not contain any information regarding the identity of the target that generated that observation. We use the notation $\boldsymbol{z}^k$ to represent the scan at time k, and $\boldsymbol{z}$ for the union of all scans up to the present time.

A *data association hypothesis* is a partitioning of $\boldsymbol{z}$ into a single set of false alarms and zero or more *tracks*, where a track is the complete set of observations corresponding to an individual target. A track is represented by a set of index pairs (k, j) from consecutive scans, each corresponding to an observation $z^{k,j}$. For example, $\{(1,2),(2,1),(3,2)\}$, $\{(3,1)\}$, and $\{(2,1),(3,1)\}$ are three possible tracks in Figure 2.1. The second index of each pair is restricted to the range $0 \leq j \leq m_k$, where the resulting $(k, 0)$ pairs refer to pseudo-observations we introduce to represent missed detections. In the same figure, $\{\{(1,2),(2,1)\},\{(1,1),(2,0),(3,1)\}\}$ is one possible data association hypothesis (or simply *hypothesis*). For clarity we represent a hypothesis as a set of tracks, and it is understood that any observation not included in one of these tracks must be clutter. Thus, this particular hypothesis asserts that there were two targets, one of which was missed at time 2, and a single clutter observation at time 3. Note that a hypothesis provides a complete explanation for how the observations were generated, effectively decomposing the multi-target state estimation task into a set of independent single-target problems.

Let $\mathcal{T}$ denote the set of all possible tracks, $\mathcal{T}_u$ the u^{th} track, and $\mathcal{T}_{u,v}$ the v^{th} index pair within track u. Note that the set $\mathcal{T}$ will grow exponentially in time. For example, suppose we have k scans with $m_k = m$ observations per scan. Requiring only that each track begin with an actual observation (not a missed detection), the total number of possible tracks is:

$$\sum_{\ell=1}^{k}(k+1-\ell)m(m+1)^{\ell-1}, \tag{2.3}$$

where ℓ ranges over possible track lengths. Thus, the space complexity of any algorithm that explicitly represents $\mathcal{T}$ must be at least $\mathcal{O}(km^k)$. For now we will ignore the obvious intractability of this representation; later we will see how the TOMHT uses pruning to represent only a tractable subset of $\mathcal{T}$.

We model the space of possible data associations with a track indicator vector, $\boldsymbol{T}$: a binary random vector with one element for each possible track. In this representation, an instantiation $\boldsymbol{T} = \boldsymbol{\tau}$ identifies the subset of tracks included in a particular hypothesis. Denote by τ_u the element of $\boldsymbol{\tau}$ corresponding to track u. Observations not included in one of the selected tracks are assumed to be false alarms.

Recall our assumption that each observation is generated by at most one target. This assumption corresponds to a constraint on the elements of $\boldsymbol{T}$: a hypothesis must not contain two tracks that both contain the same observation. More formally, let $\boldsymbol{z}^{*,0}$ be the set of pseudo-observations representing missed detections, i.e., $\boldsymbol{z}^{*,0} = \{z^{1,0}, z^{2,0}, \ldots, z^{k,0}\}$. Then we can represent the constraint with the following function:

$$h(\boldsymbol{T} = \boldsymbol{\tau}) = \begin{cases} 1 & \forall_{u \neq v} \; (\tau_u = 1 \wedge \tau_v = 1) \implies (\mathcal{T}_u \setminus \boldsymbol{z}^{*,0}) \cap (\mathcal{T}_v \setminus \boldsymbol{z}^{*,0}) = \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Note that $h(\boldsymbol{\tau})$ evaluates to zero for hypotheses that violate our assumption; such hypotheses are said to be *invalid*. We include $h(\boldsymbol{\tau})$ in the prior over hypotheses, $\Pr(\boldsymbol{T} = \boldsymbol{\tau})$, so that all invalid hypotheses are assigned zero probability.

Under this model the observations induce a joint posterior distribution over the binary track indicator vector and real-valued target state vectors: $p(\boldsymbol{x}, \boldsymbol{\tau} \mid \boldsymbol{z})$. The goal of multi-target tracking is to estimate, at each scan, the number, identities, and states of all targets currently in the surveillance region. MAP-based tracking algorithms like the TOMHT first attempt to identify the most likely hypothesis and then compute the posterior over target states

conditioned on that hypothesis:

$$\Pr(\boldsymbol{x} \mid \boldsymbol{\tau}^*, \boldsymbol{z}), \tag{2.5}$$

where

$$\boldsymbol{\tau}^* = \arg\max_{\boldsymbol{\tau}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{z}). \tag{2.6}$$

Solving for $\boldsymbol{\tau}^*$ in Equation 2.6 is the core computational challenge of this approach, and is commonly known as the data association problem.

The modeling assumptions made thus far result in the following track posterior distribution [53, 45]:

$$\begin{aligned}\Pr(\boldsymbol{\tau} \mid \boldsymbol{z}) &\propto \Pr(\boldsymbol{z} \mid \boldsymbol{\tau}) \Pr(\boldsymbol{\tau}) \\ &\propto \prod_{u=1}^{|\mathcal{T}|} \left[\frac{\lambda_\nu}{\lambda_\phi} f_o(z^{\mathcal{T}_{u,1}}) \prod_{v=2}^{|\mathcal{T}_u|} \left(\frac{p_D \Pr(z^{\mathcal{T}_{u,v}} \mid z^{\mathcal{T}_{i,1}}, \ldots, z^{\mathcal{T}_{u,v-1}}, \boldsymbol{\tau})}{\lambda_\phi} \right)^{\mathbb{1}_{[\mathcal{T}_{u,v} \neq (*,0)]}} \right. \\ &\qquad \left. (1 - p_D)^{\mathbb{1}_{[\mathcal{T}_{u,v} = (*,0)]}} \right]^{\tau_u} h(\boldsymbol{\tau}).\end{aligned} \tag{2.7}$$

A full derivation of this posterior is provided in the Appendix. The terms inside the square brackets combine the likelihood of a single track's observations under the dynamics and observation models with weighting factors to account for the target birth, death, and detection models. The logarithm of these terms is often called the *track score*. It can be thought of as a myopic measure of how likely the track is to correspond to a target, i.e., it captures the degree to which we would like to group these observations into a track *without consideration of the other observations*. In practice, many possible tracks share observations; the constraint function $h(\boldsymbol{\tau})$ encodes mutual exclusivity constraints between overlapping tracks, ensuring that Equation 2.7 normalizes to the correct posterior distribution.

Let s_u denote the score of the u^{th} track. Taking the log and rewriting (2.7) in terms of track scores yields this simple form:

$$\log \Pr(\boldsymbol{\tau} \mid \boldsymbol{z}) = \sum_{u=1}^{|\mathcal{T}|} \tau_u s_u + \log h(\boldsymbol{\tau}) + C, \tag{2.8}$$

where

$$\begin{aligned} s_u = \log \Bigg[\frac{\lambda_\nu}{\lambda_\phi} f_o(z^{\mathcal{T}_{u,1}}) \prod_{v=2}^{|\mathcal{T}_u|} \Bigg(\frac{p_D \Pr(z^{\mathcal{T}_{u,v}} \mid z^{\mathcal{T}_{i,1}}, \ldots, z^{\mathcal{T}_{u,v-1}}, \boldsymbol{\tau})}{\lambda_\phi} \Bigg)^{\mathbb{1}_{[\mathcal{T}_{u,v} \neq (*,0)]}} \\ (1 - p_D)^{\mathbb{1}_{[\mathcal{T}_{u,v} = (*,0)]}} \Bigg]. \end{aligned} \tag{2.9}$$

As first shown in [53], the constraint function $h(\boldsymbol{\tau})$ can be encoded as a set of linear constraints. Thus, the optimization problem in Equation 2.6 can be formulated as an integer linear program:

$$\begin{aligned} \boldsymbol{\tau}^* = \underset{\boldsymbol{\tau}}{\text{argmax}} \quad & \boldsymbol{s} \cdot \boldsymbol{\tau} \\ \text{subject to} \quad & \Omega \boldsymbol{\tau} \leq \mathbf{1}, \end{aligned} \tag{2.10}$$

where Ω is a sparse matrix in which rows and columns correspond to observations and tracks, respectively, and $\Omega_{ij} = 1$ if observation i is included in track j [53, 45].

We have now presented the well known track posterior distribution (Equation 2.8) and Morefield's integer program formulation of the MAP data association problem (Equation 2.10). As discussed earlier in this section, the large number of possible tracks – $\mathcal{O}(km^k)$ – makes the exact solution of Equation 2.10 intractable. In the next subsection we introduce the TOMHT, a popular algorithm for *approximating* Equation 2.10 by considering only a subset of possible tracks, thereby reducing the length of $\boldsymbol{\tau}$.

2.1.3 Track-oriented multiple hypothesis tracker

The TOMHT [45] is an *online* multi-target tracking algorithm, i.e., it attempts to solve Equation 2.10 after receiving each new scan of observations. Unfortunately, integer linear programs are NP-complete and the cost of solving the IP grows exponentially with the number of possible tracks. Equation 2.10 may be tractable for the first few scans, but because the number of possible tracks grows exponentially in time the IP quickly becomes intractable.

To address this difficulty the TOMHT constructs and maintains a small set of candidate tracks. At each time step it (1) extends the tracks in its candidate set with the observations of the new scan, and (2) discards some of the new tracks to keep the candidate set from growing too large. Thus, in the TOMHT each instance of Equation 2.10 is an IP with a number of variables equal to the number of candidate tracks.

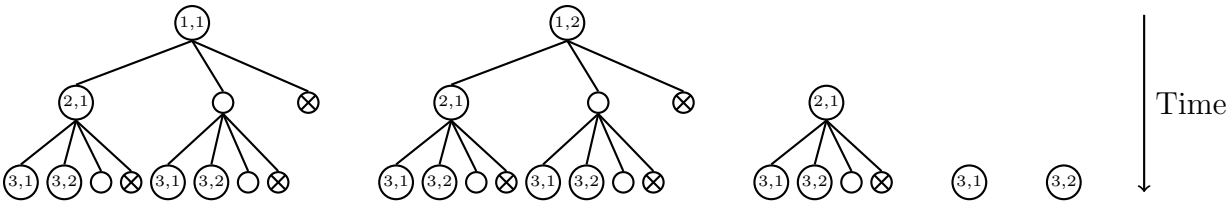
We provide pseudocode for the TOMHT in Algorithm 1. The TOMHT algorithm is comprised of two main components: a *data structure* used to enumerate the candidate track set, and a *pruning strategy* for keeping the candidate set sufficiently small. We now discuss each of these components in turn.

Track trees

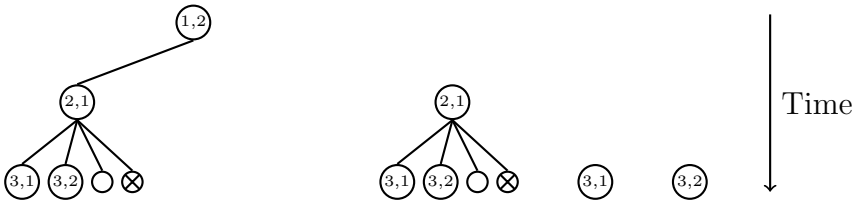
A track tree is a rooted, tree-structured graph in which nodes correspond to observations and every path from the root to a leaf corresponds to a possible track. The TOMHT maintains a collection of track trees, which together represent all candidate tracks. Each time a new scan is received, the leaves of these trees are extended with children corresponding to the new observations. Each observation of the new scan also serves as the root of a new track tree.

Algorithm 1 Track-oriented MHT (TOMHT) [45]

procedure TOMHT($\boldsymbol{z}^{1:T}, n$) // Sequentially process T scans of observations.
  $trackTrees \leftarrow \{\}$
  for $k = 1 \rightarrow T$ **do**
    ExtendTrees($trackTrees$, $\boldsymbol{z}^k$)
    Compute $\boldsymbol{\tau}^*$ // Solve IP, Equation 2.10
    nscanPrune($trackTrees$, $\boldsymbol{\tau}^*$, n)
    OutputStates($trackTrees$, $\boldsymbol{\tau}^*$)
procedure ExtendTrees($trackTrees$, $\boldsymbol{z}^k$) // Incorporate a new scan.
  for each $tree \in trackTrees$ **do**
    for each $leaf \in$ Leaves($tree$) **do**
      if $leaf \neq Death$ **then**
        for each $z^{k,i} \in \boldsymbol{z}^k$ **do**
          if WithinGate($leaf$, $z^{k,i}$) **then**
            AddChild($leaf$, $z^{k,i}$)
          AddChild($leaf$, $Miss$)
          AddChild($leaf$, $Death$)
  for each $z^{k,i} \in \boldsymbol{z}^k$ **do** AddTree($trackTrees$, CreateNode($z^{k,i}$))
procedure NscanPrune($trackTrees$, $\boldsymbol{\tau}^*$, n) // n-scan pruning on track trees.
  for each $tree \in trackTrees$ **do**
    if Depth($tree$) $> n$ **then**
      for each $leaf \in$ Leaves($tree$) **do**
        $node \leftarrow$ nthParent($leaf$, $n-1$)
        if $node \notin$ Ancestors(LeafNodes($\boldsymbol{\tau}^*$)) **then**
          PruneTrack($leaf$)
procedure AddChild($leaf$, $z^{k,i}$) // Add a new leaf node to a track tree.
  $node \leftarrow$ CreateNode($z^{k,i}$)
  $leaf.child \leftarrow node$
  $node.parent \leftarrow leaf$
  $node.state \leftarrow$ KalmanFilter($leaf.state, z^{k,i}$)
  $node.score \leftarrow$ UpdateScore($leaf.score, z^{k,i}$)
  $node.observation \leftarrow z^{k,i}$



(a) The complete set of track trees resulting from the example scenario in Figure 2.1. Empty and crossed circles represent missed detections and track deaths, respectively.



(b) If the most likely hypothesis at time step 3, $\boldsymbol{\tau}^*$, contains just the single track $\{(1,2),(2,1),(3,2)\}$, 2-scan pruning results in this reduced set of track trees. Since the leftmost tree had height greater than 2 and does not contain any tracks in $\boldsymbol{\tau}^*$, the entire tree was pruned. The rightmost three trees all had height less than or equal to 2, so they were preserved entirely. The remaining tree had depth 3, so it is vulnerable to pruning; the unpruned tracks are those that share a common ancestor with the track in $\boldsymbol{\tau}^*$ in scan 2.

Figure 2.2: Construction of track trees and illustration of n-scan pruning.

Figure 2.2a shows the collection of track trees resulting from the three scans of data in Figure 2.1. Note that while track deaths do not correspond to observations, it is convenient to represent them as nodes in the trees so that there is a one-to-one correspondence between leaves and tracks. The track trees in Figure 2.2a represent 24 possible tracks: 12 of length 3, 7 of length 2, and 5 of length 1.

If one were to add an additional node connected to the root of every track tree and perform no pruning, the result would be the search space of possible tracks. The IP in Equation 2.10 then corresponds to the multi-node search query for the set of non-conflicting leaf nodes with maximum combined score. Of course, the complete search space of possible tracks cannot be represented explicitly. Under this perspective, the pruning strategy presented next can be viewed as constructing an explicit search tree as part of a heuristic search.

Pruning

As described, the number of leaves in a track tree grows exponentially with the number of scans. Specifically, if there are m observations per scan, the number of leaves will grow at a rate of $\mathcal{O}(m^k)$ where k is the number of scans. The TOMHT limits this growth using a strategy called *n-scan pruning* [11, 10]. After incorporating the k^{th} scan, the TOMHT solves Equation 2.10 (considering only those tracks in its current candidate set) to find the most likely set of tracks, $\boldsymbol{\tau}^*$. It then prunes all tracks that do not satisfy at least one of the following two conditions:

1. Share a common ancestor with one of the tracks selected by $\boldsymbol{\tau}^*$ within the n most recent scans (where n is a parameter of n-scan pruning).
2. Belong to a tree with height at most n.

In Figure 2.2, for example, 2-scan pruning results in the track trees shown in Figure 2.2b, reducing the number of candidate tracks from 24 to 10. Notice that n-scan pruning effectively eliminates all branching of the track trees above the most recent n scans. Pseudocode for n-scan pruning is provided in Algorithm 1.

The complexity of n-scan pruning is dominated by the computation of $\boldsymbol{\tau}^*$, but in the context of the TOMHT we can assume that $\boldsymbol{\tau}^*$ has already been computed for the purpose of reporting tracker output (Equations 2.5-2.6). Thus, the additional complexity due to n-scan pruning arises only from finding and deleting nodes of pruned tracks. This can be accomplished by first "marking" all nodes corresponding to tracks in $\boldsymbol{\tau}^*$ and then traversing the trees upward from each leaf, either for $n-1$ scans or until a root node or marked node is encountered. The complexity of this operation is $\mathcal{O}(n|\widetilde{\mathcal{T}}|)$, where $\widetilde{\mathcal{T}}$ is the current set of candidate tracks. Since pruning is performed after incorporating each new scan, the pre-pruning size of $\widetilde{\mathcal{T}}$ is $\mathcal{O}(nm^{n+1})$ rather than $\mathcal{O}(km^k)$. From this point on we will simply use

$\mathcal{T}$ to refer to the set of candidate tracks; it should be understood that, in the context of the TOMHT, we never work with the unpruned set of all possible tracks.

Most implementations of the TOMHT apply additional pruning techniques such as gating, track score thresholding, and track merging [11]. For simplicity we do not focus on these techniques, but they are trivially compatible with the marginalization and parameter estimation techniques we develop in this and the next chapter. In our implementation we use only n-scan pruning and gating. Gating refers to a class of simple, local pruning heuristics: rather than extending each node of a track tree with a child for *every* observation in the current scan, we create a child node only for track continuations that look locally plausible. If each existing track has only $p < m$ plausible continuations at each time step, the size of the pre-pruning candidate track set is reduced from $\mathcal{O}(nm^{n+1})$ to $\mathcal{O}(nmp^n)$. By reducing the base of the exponential term, gating can enable one to choose a larger n for n-scan pruning while still maintaining tractability. Thinking back to the search perspective discussed at the end of the track tree subsection, gating can be viewed as a modification of the heuristic using partial expansion to reduce search tree size.

Gating strategies differ in how they formalize this notion of plausibility. We implement elliptical gating, which takes advantage of the fact that the normalized residuals in a Kalman filter – the squared Mahalanobis distance between the predicted state and observation at a particular time – are distributed according to a $\mathcal{X}^2_{d_x}$ distribution [5]. By limiting track extensions to observations with normalized residuals below a prespecified threshold, we can prune unlikely observations with a bounded error rate corresponding to the tail probability of the $\mathcal{X}^2_{d_x}$ distribution. In our implementation we set this threshold such that the correct track extension is pruned only 0.1% of the time (assuming our observation and dynamics models are correct).

PROPOSITION **2.1.** *The TOMHT algorithm has complexity* $\mathcal{O}((n+d_x^2+d_z^2)nmp^n+2^{nmp^n}+mk)$ *to process the* k^{th} *scan of data, assuming* n*-scan pruning and gating that results in a branching*

factor of p.

Proof. In a single time step, the TOMHT grows the track trees, solves for $\boldsymbol{\tau}^*$, performs pruning, and returns state trajectories; we will analyze each of these operations in turn.

Adding a single node to a track tree requires a Kalman filtering step, which has complexity $\mathcal{O}(d_x^2 + d_z^3)$. With n-scan pruning and gating that results in a branching factor of p, this stage results in a total of $\mathcal{O}(nmp^n)$ tracks for a total complexity of $\mathcal{O}(nmp^n(d_x^2 + d_z^2))$.

Solving an integer linear program is exponential in the number of variables. There is one binary variable per track and $\mathcal{O}(nmp^n)$ total tracks, so the complexity is $\mathcal{O}(2^{nmp^n})$.

As discussed above, the added complexity of n-scan pruning is linear in n for each track, i.e., $\mathcal{O}(n^2mp^n)$.

Finally, to output the filtered state trajectories one must simply traverse the tracks trees from each leaf node corresponding to a track in $\boldsymbol{\tau}^*$ up to its root node, gathering state estimates along the way. Since no two tracks in $\boldsymbol{\tau}^*$ can contain the same observation, there is a maximum of $\mathcal{O}(mk)$ nodes to traverse.

Putting all of these together, the overall asymptotic complexity of the TOMHT algorithm is $\mathcal{O}((n + d_x^2 + d_z^2)nmp^n + 2^{nmp^n} + mk)$. □

2.1.4 Other popular algorithms for multi-target tracking

In this section we briefly mention some popular alternative methods for multi-target tracking. Most closely related is the original multiple hypothesis tracker [61], sometimes called the hypothesis-oriented MHT (HOMHT) to distinguish it from the TOMHT. Like the TOMHT, the HOMHT is a pruning-based algorithm. It differs from the TOMHT in that it enumerates full data association hypotheses rather than tracks; i.e., it constructs a tree of *hypotheses*,

where each node corresponds to an entire vector $\boldsymbol{\tau}$. As a result, there is no need to solve an integer program to find the most likely candidate hypothesis. The downside of this approach is a much greater space complexity – the TOMHT can implicitly represent many more hypotheses than a HOMHT would ever be able to represent explicitly.

The joint probabilistic data association filter (JPDAF) [23, 4] attempts to solve a slightly different problem: rather than estimating $\Pr(\boldsymbol{x} \mid \boldsymbol{\tau}^*, \boldsymbol{z})$ as in the TOMHT, the JPDAF aims to compute $\Pr(\boldsymbol{x} \mid \boldsymbol{z})$, i.e., it attempts to marginalize over possible data association hypotheses rather than conditioning on the most likely one. To achieve tractability the JPDAF assumes that the marginal target states are Gaussian distributed and that target states are independent given past observations. This results in a fairly efficient algorithm, but the relatively strong assumptions make it less robust than the TOMHT [4].

2.2 Estimating track marginals

Having introduced the popular TOMHT algorithm for multi-target tracking, we are now ready to present our novel contributions. We consider the task of computing the marginal probability of a single track, restricted to the candidate track set produced by the TOMHT:

$$\begin{aligned} b_u &= \Pr(T_u = 1 \mid \boldsymbol{z}) \\ &= \sum_{\boldsymbol{\tau} \in \{0,1\}^{|\mathcal{T}|}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{z}) \mathbb{1}_{[\tau_u = 1]} \end{aligned} \tag{2.11}$$

The sum in Equation 2.11 ranges over all subsets of the candidate track set. Since explicit summation is generally intractable, we consider two approaches to approximate marginalization: one based on the k-best hypotheses, and one based on variational inference algorithms operating on a factor graph representation of the track posterior.

2.2.1 Marginalization via the k-best hypotheses

As mentioned previously, even with n-scan pruning it is infeasible to sum over the entire space of hypotheses in (2.11). However, it is often the case in practice that the posterior probability mass is concentrated on a relatively small fraction of the hypotheses. The k-best estimator takes advantage of this tendency by restricting the sum to a tractable subset of hypotheses with high mass. Intuitively, if the combined mass of discarded hypotheses is small enough, the estimated track marginals will be close to the exact values. Formally, the k-best estimator is defined as follows:

$$b_u^{kbest} = \frac{\sum_{\boldsymbol{\tau} \in \{\boldsymbol{\tau}^{(1)}, \ldots, \boldsymbol{\tau}^{(k)}\}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{z}) \mathbb{1}_{[\tau_u = 1]}}{\sum_{\boldsymbol{\tau} \in \{\boldsymbol{\tau}^{(1)}, \ldots, \boldsymbol{\tau}^{(k)}\}} \Pr(\boldsymbol{\tau} \mid \boldsymbol{z})} \tag{2.12}$$

where $\boldsymbol{\tau}^{(i)}$ is the hypotheses with the i^{th} highest posterior probability mass. Note that the hypothesis probabilities, $\Pr(\boldsymbol{\tau} \mid \boldsymbol{z})$, are themselves intractable since they must be normalized over the entire hypothesis space. Fortunately, in Equation 2.12 the normalizing constants cancel and b_u^{kbest} is easily computable for moderate k. Pseudocode for computing Equation 2.12 is presented in Algorithm 2.

Algorithm 2 k-best estimator for track marginals (KBEST-MARG)

```
1: procedure KBEST-MARG(trackTrees, k)
2:     ip ← BUILDIP(trackTrees)                          // Equation 2.10
3:     τ^(1) ← SOLVEIP(ip)
4:     for i = 2 → k do
5:         ADDCONSTRAINT(ip)                             // Exclude τ^(i−1), Equation 2.13
6:         τ^(i) ← SOLVEIP(ip)
       return b_u^kbest                                  // Equation 2.12
```

The most expensive piece of this algorithm is the computation of the k-best hypotheses. Recall the integer program used to compute the most likely hypothesis (Equation 2.10). The same approach can be used to compute the next best hypothesis by adding a single linear

constraint excluding $\boldsymbol{\tau}^{(1)}$ from the solution space:

$$\sum_{u:\tau_u^{(1)}=0} \tau_u + \sum_{u:\tau_u^{(1)}=1} (1-\tau_u) > 0 \tag{2.13}$$

PROPOSITION **2.2.** *The constraint in Equation 2.13 excludes $\boldsymbol{\tau}^{(1)}$, and only $\boldsymbol{\tau}^{(1)}$, from the solution space.*

Proof. First we show that $\boldsymbol{\tau}^{(1)}$ violates the constraint. The first sum ranges over tracks not included in $\boldsymbol{\tau}^{(1)}$, so each term in the sum is zero and thus

$$\sum_{u:\tau_u^{(1)}=0} \tau_u^{(1)} = 0$$

The second sum ranges over tracks that *are* included in $\boldsymbol{\tau}^{(1)}$, so again each term is zero and we have

$$\sum_{u:\tau_u^{(1)}=1} (1-\tau_u^{(1)}) = 0$$

We are left with $0 > 0$, so our constraint is violated and $\boldsymbol{\tau}^{(1)}$ is excluded from the solution space.

Now consider a different hypothesis, $\boldsymbol{\tau} \neq \boldsymbol{\tau}^{(1)}$. The two vectors $\boldsymbol{\tau}$ and $\boldsymbol{\tau}^{(1)}$ differ in at least one track; let u index one such track.

Case 1. $\tau_u^{(1)} = 0$

Since $\tau_u^{(1)} = 0$, this track contributes a term to the first sum in (2.13). We have assumed that $\tau_u \neq \tau_u^{(1)}$, so $\tau_u = 1$. Thus, τ_u contributes 1 to the first sum. Since all terms are non-negative, the overall sum must be positive and the constraint is satisfied.

Case 2. $\tau_u^{(1)} = 1$

Since $\tau_u^{(1)} = 1$, this track contributes a term to the second sum in (2.13). We have assumed that $\tau_u \neq \tau_u^{(1)}$, so $\tau_u = 0$. Thus, τ_u contributes $(1 - 0) = 1$ to the second sum. Since all terms are non-negative, the overall sum must be positive and the constraint is satisfied.

Thus, Equation 2.13 does not exclude any hypotheses other than $\boldsymbol{\tau}^{(1)}$. □

Repeating this process, alternately solving the integer program and adding a constraint to exclude the previous solution, yields the top k hypotheses.

Each successive integer program is more complex than the previous – note that Equation 2.13 is a global constraint, defined over all variables – so in practice this approach scales worse than linearly with k. Faster algorithms exist to approximate the k-best hypotheses [59] [24] [6], but are not explored here.

PROPOSITION **2.3.** *KBEST-MARG has complexity $\mathcal{O}(k2^{n^2mp^{n-1}})$, where k is the number of hypotheses to compute, n is the parameter of n-scan pruning, p is the number of observations per gate, and m is the total number of observations per scan.*

Proof. Constructing the initial IP in Equation 2.10 requires population of the constraint matrix Ω. Assuming n-scan pruning, this matrix has at most $\mathcal{O}(n^2mp^{n-1})$ nonzero elements (n per track).

Solving the IP is exponential in the number of variables, requiring $\mathcal{O}(2^{n^2mp^{n-1}})$ time to solve once and thus dominating the overall complexity. KBEST-MARG must solve the IP k times (adding one new constraint before each solution) for a total cost of $\mathcal{O}(k2^{n^2mp^{n-1}})$. □

The number of hypotheses, k, serves as a tuning parameter to trade off speed and accuracy. Setting k to the total number of valid hypotheses results in an exact – but intractable – algorithm. Setting k to 1, on the other hand, corresponds to the MAP estimator, assigning

probability 1 to every track in $\boldsymbol{\tau}^*$ and 0 to the rest. The accuracy of the estimator at small values of k depends the fraction of probability mass concentrated in the top k hypotheses.

The k-best estimator can be viewed as a partial translation from the track-oriented representation of the TOMHT to the hypothesis-oriented representation of its predecessor, the HOMHT [61]. The TOMHT's advantage over the HOMHT stems from its ability to represent a number of hypotheses that is exponential in the number of candidate tracks. Since the k-best approach discards this advantage by explicitly converting to the hypothesis-oriented representation, it makes sense to consider marginalization methods that operate directly in the track-oriented data association space.

2.2.2 Marginalization via variational message-passing

Variational message passing algorithms for graphical models, described in Section 1.4, offer an alternate approach to approximating the marginalization in Equation 2.11. By taking advantage of conditional independencies present in the model, these algorithms can often obtain quick marginal estimates even when exact marginalization is very costly.

To use these algorithms for track marginal estimation, we first formulate the track posterior distribution (Equation 2.8) as a factor graph over a set of random variables, $\boldsymbol{y}$. Figure 2.3 illustrates the factor graph corresponding to the example data of Figure 2.1. The factor graph contains one variable corresponding to each track tree node. Each variable is binary, i.e. $y_i \in \{0, 1\}$, and serves as an indicator for the *partial* track terminating in its corresponding node in the track tree. For example, the variable in the middle-left of Figure 2.3 labeled $\textcircled{2,1}$ is an indicator variable for the partial track $\{z^{1,1}, z^{2,1}\}$. If a variable corresponds to a track tree leaf node, we call it a track indicator variable. Since there is a one-to-one correspondence between track indicator variables and candidate tracks, we will refer to the set of these variables as $\mathcal{T}$. These track indicator variables correspond exactly to the variables of the

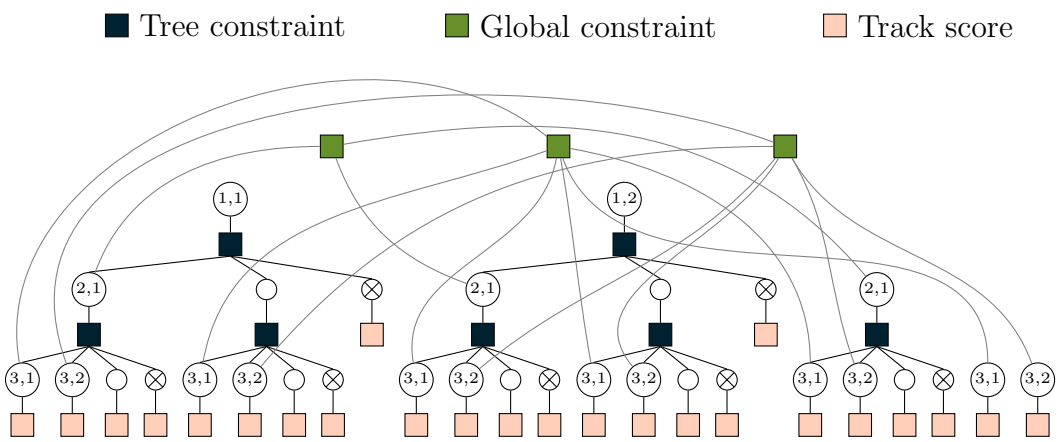


Figure 2.3: The factor graph corresponding to the track trees in Figure 2.2 (a). Round and square nodes represent variables and factors, respectively. Variable nodes are annotated as in Figure 2.2 to emphasize the correspondence between track tree nodes and factor graph variables. The corresponding joint distribution is the track posterior in Equation 2.8: the constraint factors correspond to $h(\boldsymbol{\tau})$ and the score factors contribute the remainder. For clarity, global constraint factors with only one neighbor are omitted; they have no effect on the distribution. Also note that this model corresponds to the *unpruned* set of track trees. If 2-scan pruning were in use, the model would mirror the structure of Figure 2.2 (b) instead.

integer program (Equation 2.10). The additional, non-leaf variables exist only to enable a structured encoding of the constraint $h(\boldsymbol{\tau})$, which we discuss next. In total, since there is one variable for each track tree node, the graphical model has $\mathcal{O}(nmp^{n-1})$ variables, assuming n-scan pruning, m variables per scan, and a branching factor limited to p by gating.

The factors are grouped into three classes: tree constraints, global constraints, and track score factors. Together, the two classes of constraint factors encode $h(\boldsymbol{\tau})$, our constraint that each observation belongs to at most one track. Before defining these factors we must introduce some notation. We will use a single subscript, e.g. y_i or y_k, to refer to an arbitrary variable in our factor graph. Further, let $ch(y_i)$ denote the variables that are children of y_i (borrowing parent-child relationships from the corresponding track tree), and $\boldsymbol{y}^{i,j}$ the set of variables corresponding to the observation $z^{i,j}$. Then we define tree constraints f_i^t and global

constraints f_i^g as follows:

$$f_i^t(y_i, ch(y_i)) = \begin{cases} 1 \ : \ (y_i = 0 \wedge \sum_{y_k \in \boldsymbol{y}_{ch(i)}} y_k = 0) \\ \quad \vee (y_i = 1 \wedge \sum_{y_k \in \boldsymbol{y}_{ch(i)}} y_k = 1) \\ 0 \ : \ \text{otherwise} \end{cases} \tag{2.14}$$

$$f_{i,j}^g(\boldsymbol{y}^{i,j}) = \begin{cases} 1 \ : \sum_{y_k \in \boldsymbol{y}^{i,j}} y_k \leq 1 \\ 0 \ : \text{otherwise.} \end{cases} \tag{2.15}$$

Every instantiation of the leaf variables corresponds to a hypothesis, and the constraint factors assign zero probability to all invalid hypotheses. Score factors, f_i^s, weight hypotheses according to the scores of their constituent tracks:

$$f_i^s(y_i) = \begin{cases} \exp(s_i) & : \ y_i = 1 \\ 1 & : \ y_i = 0. \end{cases}$$

There is one score factor for each track indicator variable.

The probability mass function represented by the factor graph may be written as:

$$\Pr(\boldsymbol{y}) \propto \prod_{y_i \notin \mathcal{T}} f_i^t(y_i, ch(y_i)) \prod_{z^{i,j} \in \boldsymbol{z}} f_{i,j}^g(\boldsymbol{y}^{i,j}) \prod_{y_i \in \mathcal{T}} f_i^s(y_i).$$

An instantiation of the variables will evaluate to the exponentiated sum of the selected track scores if it corresponds to a valid hypothesis, and zero otherwise. Thus, the marginal distribution of the track indicator variables is identical to the track posterior in Equation 2.8. Having formulated the track posterior as a factor graph, we can now run message-passing algorithms like BP and GBP (see Section 1.4) to compute approximate track marginals. Pseudocode for this approach to marginalization is provided in Algorithm 3.

PROPOSITION **2.4.** *The BP-MARG algorithm has complexity* $\mathcal{O}(T(nmp^n + n^2m^2p^{n-2}))$,

Algorithm 3 BP estimator for track marginals (BP-MARG)

```
1: procedure BP-MARG(trackTrees)
2:     fg ← BuildFactorGraph(trackTrees)
3:     runBP(fg)                                          // (or GBP)
4:     trackMarginals ← []
5:     for each var ∈ Variables(fg) do
6:         if IsLeaf(var) then
7:             Append(trackMarginals, belief(var))
       return trackMarginals
```

where T is the number of BP iterations, n is the parameter of n-scan pruning, p is the number of observations per gate, and m is the total number of observations per scan.

Proof. The factor graph has one variable for each track tree node, for a total of $\mathcal{O}(nmp^n)$ variable nodes. Each variable has one child factor, zero or one parent factor, and one global constraint factor; thus, the number of edges over which to pass messages is also $\mathcal{O}(nmp^n)$. As all variables are binary and each variable has at most three neighbors, computing a variable message requires $\mathcal{O}(1)$ time. The total complexity to compute variable messages along every edge is $\mathcal{O}(nmp^n)$.

As discussed further in Section 2.2.2, the sparsity structure of both tree constraints and global constraints are such that computing all messages out of a single factor requires time linear in the number of adjacent variables. Again assuming gating that limits the branching factor of the track trees, a tree constraint factor has a scope of at most $p+1$ variables. There are $\mathcal{O}(nmp^{n-1})$ tree constraint factors – one for each non-leaf variable. In total, computing all outgoing messages from tree constraint factors requires $\mathcal{O}(nmp^n)$.

Finally, there are nm global constraint factors – one per observation. The largest factors correspond to observations at the most recent time step that fell inside the gate of every leaf node at the previous time step. This results in an global constraint over $\mathcal{O}(nmp^{n-2})$ variables, so the cost to compute all outgoing messages from the global constraint factors is $\mathcal{O}(n^2m^2p^{n-2})$.

If BP is run for a total of T iterations, the overall complexity is $\mathcal{O}(T(nmp^n + n^2m^2p^{n-2}))$. $\square$

This approach is potentially attractive due to the computational efficiency of BP and GBP. However, their accuracy depends heavily on model-specific characteristics [33, 82]. To assess their accuracy on models generated by the TOMHT, we conduct an empirical evaluation relative to known ground truth marginals in Section 2.2.3.

Graphical models and approximate inference methods have also been applied to other formulations of the data association problem. The most closely related work is Williams et al. [80], which considers a graphical model formulation of the data association problem and uses BP to estimate marginal associations. Their treatment of multi-scan association probabilities is similar in spirit to ours, but instead of the popular track-oriented MHT representation it uses a hybrid "target-oriented" representation. The target-oriented representation assumes that the number of targets is known (though this can be relaxed by using a track initialization framework, e.g. [32]), and it is unclear how to perform pruning in this representation. Chen et al. [15] uses BP to approximate association probabilities in a sensor network, but only considers associations within a single scan. To our knowledge, this is the first work to compute marginal multi-scan association probabilities using the representation of the TOMHT. Working within the TOMHT representation is particularly attractive because it naturally handles track birth and death, supports pruning to trade off speed and accuracy, and has been successfully deployed in real-world systems.

Sparse message updates for BP inference

The constraint factors (Equations 2.14-2.15) may be defined over a large number of variables, making direct computation of the factor message in Equation 1.27 intractable. Fortunately, the sparsity structure of these factors admits an exact reformulation that reduces the complexity from exponential to linear in the number of variables. The key idea is to modify the

sum over the factor domain to explicitly iterate only over its non-zero elements. See [73] for a general treatment of optimizations of this sort. The exact forms of the efficient message updates for this model are provided below:

Tree constraint factor f_i^t to parent variable Y_i:

$$m_{f_i^t \to Y_i}(0) \propto \prod_{Y_j \in ch(Y_i)} m_{Y_j \to f_i^t}(0) \tag{2.16}$$

$$m_{f_i^t \to Y_i}(1) \propto \sum_{Y_j \in ch(Y_i)} \left[m_{Y_j \to f_i^t}(1) \prod_{Y_k \in ch(Y_i) \setminus Y_j} m_{Y_k \to f_i^t}(0) \right] \tag{2.17}$$

Tree constraint factor f_i^t to child variable Y_j:

$$\begin{aligned} m_{f_i^t \to Y_j}(0) \propto\ & m_{Y_i \to f_i^t}(0) \prod_{Y_k \in ch(Y_i) \setminus Y_j} m_{Y_k \to f_i^t}(0) \\ & + \sum_{Y_k \in ch(Y_i) \setminus Y_j} \left[m_{Y_i \to f_i^t}(1) m_{Y_k \to f_i^t}(1) \prod_{Y_l \in ch(Y_i) \setminus \{Y_j, Y_k\}} m_{Y_l \to f_i^t}(0) \right] \end{aligned} \tag{2.18}$$

$$m_{f_i^t \to Y_j}(1) \propto m_{Y_i \to f_i^t}(1) \prod_{Y_k \in ch(Y_i) \setminus Y_j} m_{Y_k \to f_i^t}(0) \tag{2.19}$$

Global constraint factor $f_{i,j}^g$ to neighbor variable Y_k:

$$m_{f_{i,j}^g \to Y_k}(0) \propto \prod_{Y_l \in \boldsymbol{y}^{i,j} \setminus Y_k} m_{Y_l \to f_{i,j}^g}(0) + \sum_{Y_l \in \boldsymbol{y}^{i,j} \setminus Y_k} \left[m_{Y_l \to f_{i,j}^g}(1) \prod_{y_m \in \boldsymbol{y}^{i,j} \setminus \{Y_k, Y_l\}} m_{y_m \to f_{i,j}^g}(0) \right] \tag{2.20}$$

$$m_{f_{i,j}^g \to Y_k}(1) \propto \prod_{Y_l \in \boldsymbol{y}^{i,j} \setminus Y_k} m_{Y_l \to f_{i,j}^g}(0) \tag{2.21}$$

The above updates enable efficient loopy BP despite the high-cardinality factors. Similar techniques can be used for algorithms that operate on region graphs, such as GBP, but the

Figure 2.4: Decomposition of large constraint factors. (a) A global constraint factor f, defined over four variables for the sake of illustration. (b) By introducing two auxiliary variables, V_1 and V_2, we can decompose the constraint into three factors defined over two or three variables each.

equations would be more complex since each region may have a different sparsity structure.

Constraint decomposition for GBP inference

Our experiments use a generic implementation of GBP that does not take advantage of factor sparsity. Rather than writing custom GBP code that exploits the sparsity, we opt to perform a preprocessing step on the model. Specifically, by introducing auxiliary variables we can replace a single constraint defined over m variables with at most $m - 1$ constraints defined over three variables each.

Recall that a global constraint is satisfied when at most one adjacent variable is "active." Equivalently, we can split a constraint's variables into two sets, enforce an *at-most-one* constraint on each set separately, and add a final *at-most-one* constraint over auxiliary variables defined to be the logical-or of the variables in each set. Figure 2.4 illustrates this transformation for a global constraint factor f defined over four variables. Factors labeled f are simple *at-most-one* constraints. The factors labeled g simultaneously enforce an *at-most-one* constraint on their children and constrain the parent variable to be equal to the logical-or of their children. Applying this transformation recursively to a global constraint over m variables replaces a single factor of cardinality 2^m with $\mathcal{O}(m)$ factors of cardinality $\mathcal{O}(1)$. The resulting model has no intractably large factors and we can apply off-the-shelf

inference algorithms (though they will be less efficient than if they were to directly take advantage of the constraint structure as in the previous section).

2.2.3 Experimental results

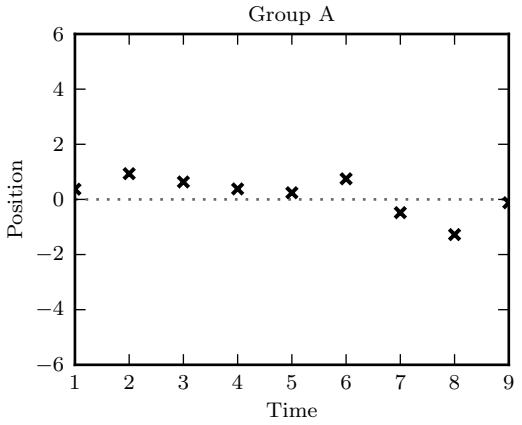
Both the k-best estimator and the variational approximations (BP and GBP) sacrifice correctness to attain tractability, and it is not immediately clear how the respective approximations impact the resulting marginal estimates. We conducted an empirical study using simulated data, enabling direct comparison of the estimated marginals to their exact values.

The input to each algorithm is a collection of track trees representing the state of a TOMHT (its internal track forest) after processing a sequence of scans. In the context of these experiments, we will refer to such track forests as *models*. Model generation is a three step process:

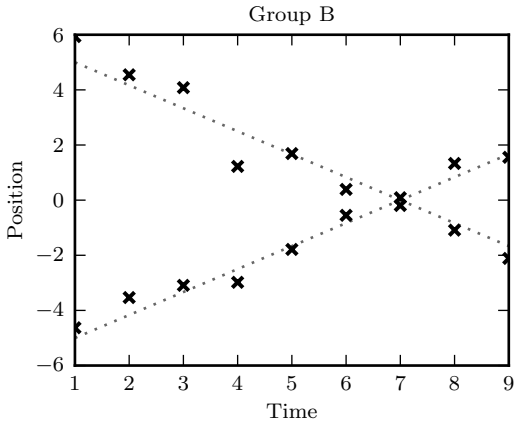
1. Simulate a set of target state trajectories.
2. Sample observations at each time step according to a specified sensor model.
3. Process the simulated scans with a TOMHT.

At the end of this process, the track trees stored by the TOMHT are saved to a model file. We generated three groups of models, each corresponding to a different set of underlying target state trajectories as shown in Figure 2.5. The three groups are designed to explore a range of problem characteristics, subject to the constraint that exact inference must be feasible so that the approximate marginals can be evaluated against exact values. The median number of variables in the models is 63 for Group A, 197 for Group B, and 264 for Group C.

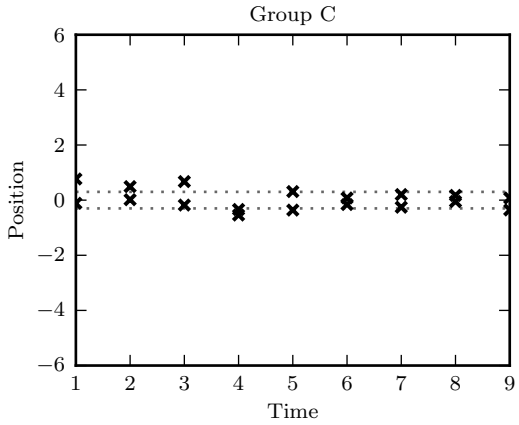
In step 2, the data was generated using a linear Gaussian observation model with the fol-



(a)



(b)



(c)

Figure 2.5: One example scene from each of the three groups used to evaluate marginalization accuracy. The groups are designed to explore a range of local structures that arise in real-world scenarios.

lowing parameters:

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} .5^2 \end{bmatrix} \qquad \lambda_\phi = 0 \qquad \lambda_\nu = 0 \qquad p_D = .95 \qquad p_\gamma = 0.$$

In step 3, tracking was performed using a TOMHT with H, R, and p_D set as above and the remaining parameters set as follows:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} .5^2 & 0 \\ 0 & .2^2 \end{bmatrix} \qquad \lambda_\phi = 1 \qquad \lambda_\nu = 1 \qquad p_\gamma = .1$$

The tracker used 4-scan pruning; models generated with 3-scan pruning were too small to be interesting, and those generated with 5-scan pruning were too large to solve exactly. We generated 50 models in each group, and for each model computed approximate marginals using k-best, BP, and GBP, as well as the exact marginals via variable elimination. With BP and GBP, all messages were initialized with ones. BP was run until all messages converged with a tolerance of 1E-5 or for a maximum of 100 iterations. GBP was run until its estimate of $\log Z$ converged with a tolerance of 1E-6 or for a maximum of 200 iterations. Experiments were parallelized across a cluster using GNU Parallel [72].

Figure 2.6 plots the running time and marginalization error for several k-best and messaging-passing estimators. As expected, increasing k for the k-best estimator and cluster size for GBP both decrease marginalization error. The 1-best estimator universally has the highest error and GBP-20 the lowest. BP typically produces marginals with error between 1-best and 10-best.

BP is, by far, the fastest algorithm. While the computational complexity of GBP is tunable, the implementation used in our experiments is from a general-purpose inference package. It is written in C++ and reasonably efficient, but does not take advantage of the sparsity structure of the factors as our BP implementation does, resulting in an overhead that dominates the

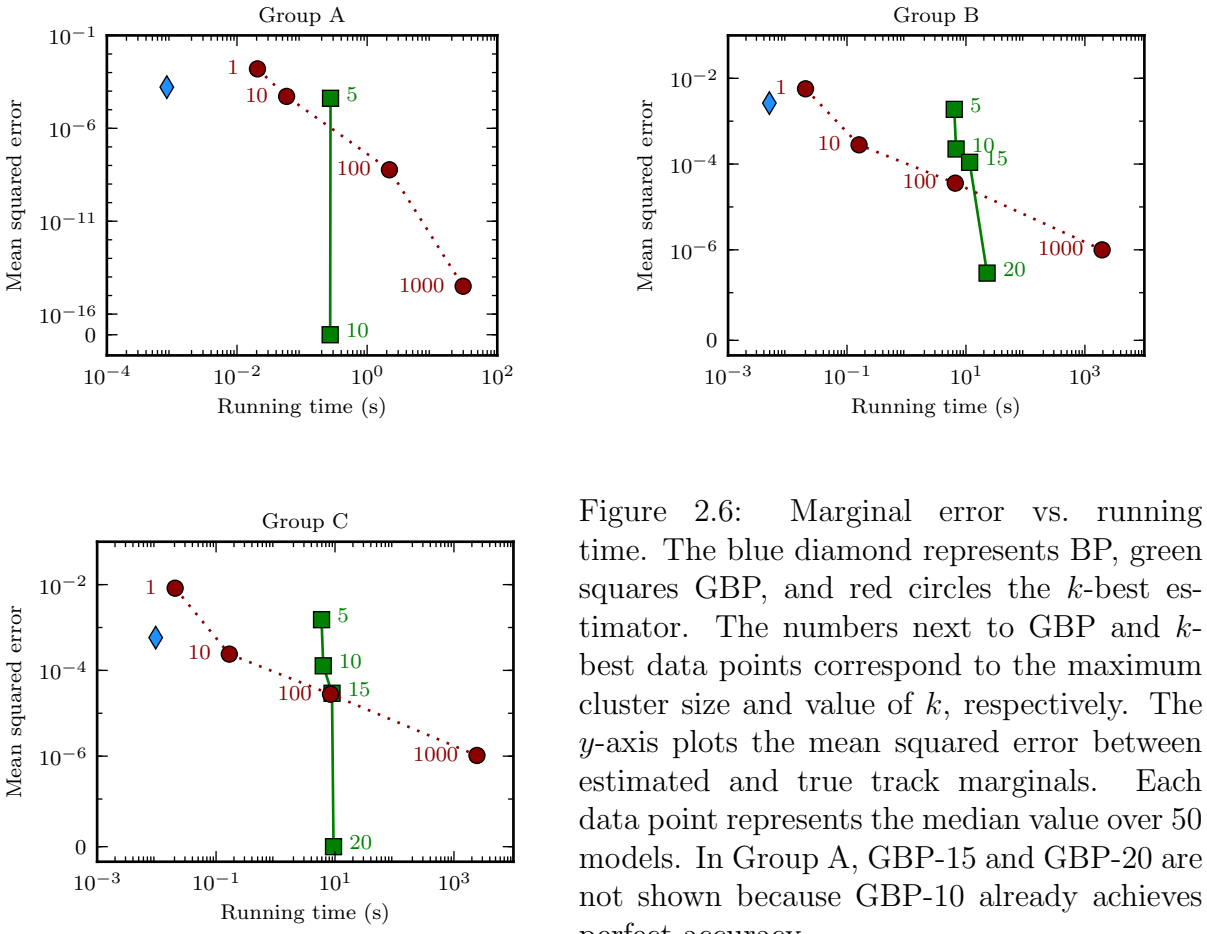


Figure 2.6: Marginal error vs. running time. The blue diamond represents BP, green squares GBP, and red circles the k-best estimator. The numbers next to GBP and k-best data points correspond to the maximum cluster size and value of k, respectively. The y-axis plots the mean squared error between estimated and true track marginals. Each data point represents the median value over 50 models. In Group A, GBP-15 and GBP-20 are not shown because GBP-10 already achieves perfect accuracy.

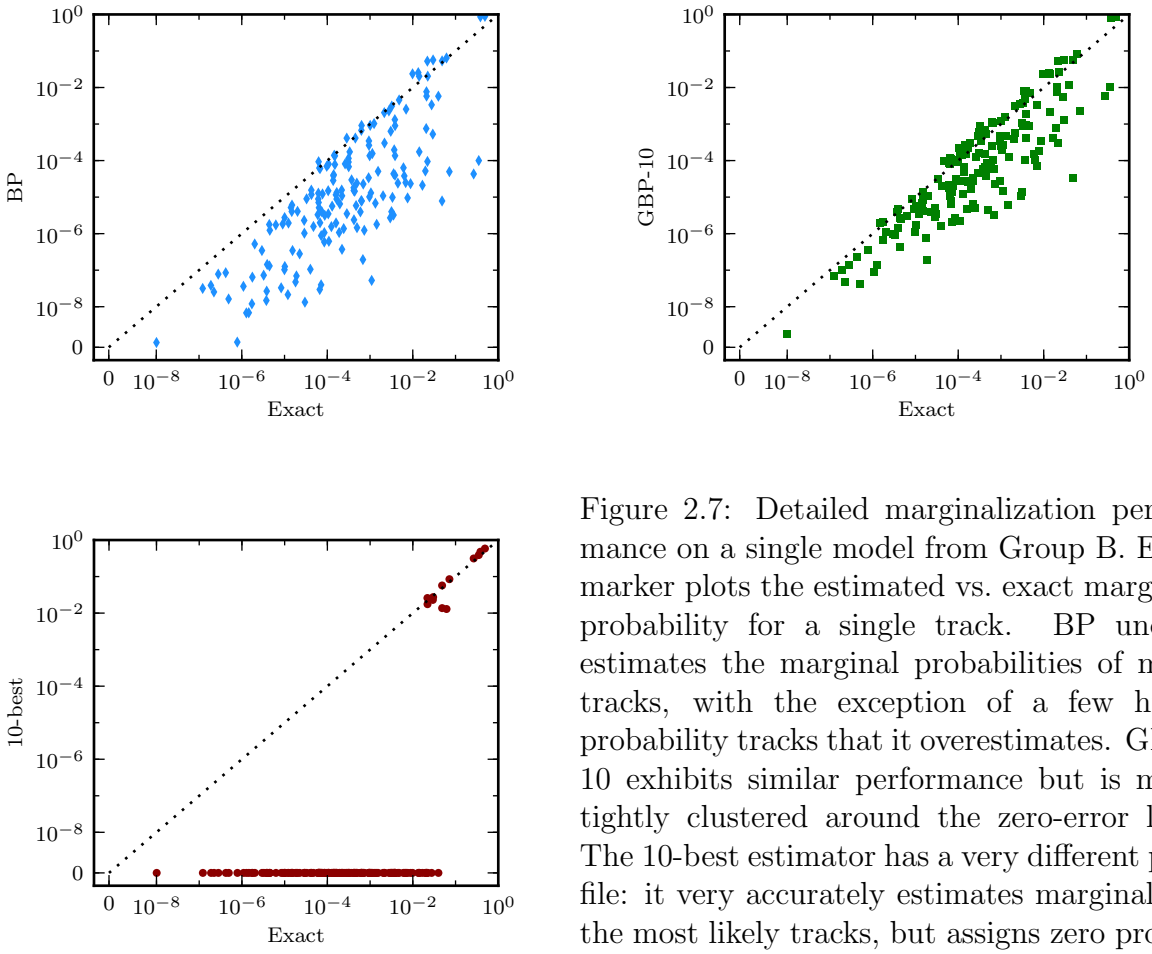


Figure 2.7: Detailed marginalization performance on a single model from Group B. Each marker plots the estimated vs. exact marginal probability for a single track. BP underestimates the marginal probabilities of most tracks, with the exception of a few high-probability tracks that it overestimates. GBP-10 exhibits similar performance but is more tightly clustered around the zero-error line. The 10-best estimator has a very different profile: it very accurately estimates marginals of the most likely tracks, but assigns zero probability to all of the less likely tracks.

computation time for small cluster sizes. On the other hand, the k-best estimator scales poorly with k; with $k = 100$ it already takes 10 seconds to run in groups B and C. Values up to $k = 10$ may be usable in practice, but even the 1-best estimator is slower than BP. The appropriate algorithm for a particular application will depend both on its running time requirements and its sensitivity to error in the marginals. If the application can tolerate moderate error, BP is attractive due to its high efficiency. If higher accuracy is required, then a more expensive GBP or k-best estimator may be preferable.

To better illustrate the accuracy profiles of the various estimators, Figure 2.7 plots estimated vs. exact marginals for the individual tracks of a single model from Group B. Note that

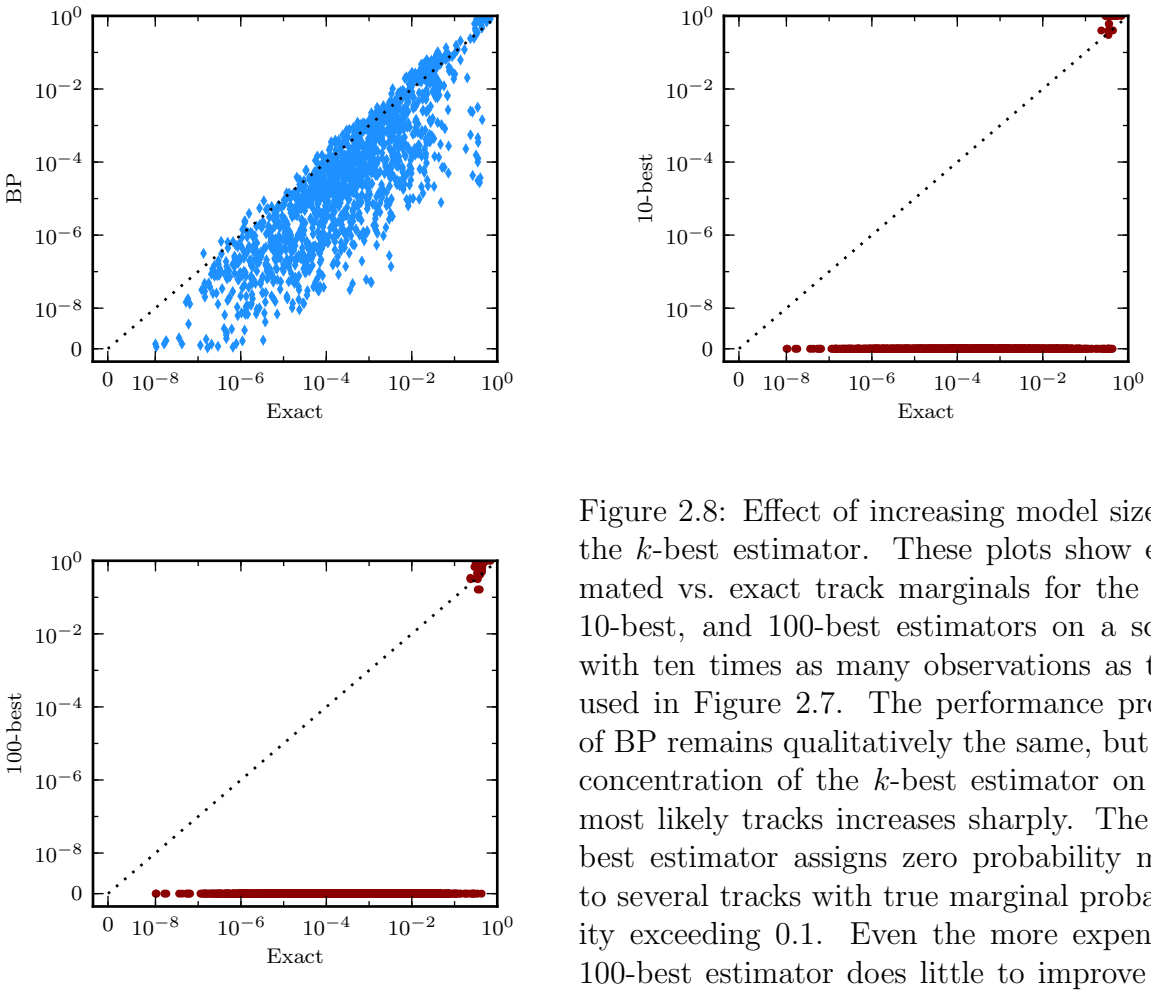


Figure 2.8: Effect of increasing model size on the k-best estimator. These plots show estimated vs. exact track marginals for the BP, 10-best, and 100-best estimators on a scene with ten times as many observations as that used in Figure 2.7. The performance profile of BP remains qualitatively the same, but the concentration of the k-best estimator on the most likely tracks increases sharply. The 10-best estimator assigns zero probability mass to several tracks with true marginal probability exceeding 0.1. Even the more expensive 100-best estimator does little to improve the estimates.

the 10-best estimator is very accurate for the highest probability tracks but assigns zero probability to all tracks with probability less than 0.06. BP and GBP, on the other hand, are less accurate on the high probability tracks but do much better on the majority of less likely tracks.

As model size increases, the k-best estimator's concentration of mass in the few most likely tracks becomes even more exaggerated. In the extreme case, the top k hypotheses may actually be very minor perturbations of $\boldsymbol{\tau}^{(1)}$, in which case the k-best estimator will be little better than the 1-best estimator. To illustrate this effect while still retaining the ability to compute exact marginals, we constructed a larger scene by "stacking" 10 different scenes from Group B, overlapping them in time but shifting them to be well separated in space. The spatial separation, combined with elliptical gating, results in a model that is 10 times larger than any individual model from Group B but still computationally tractable.

Figure 2.8 plots estimated vs. exact track marginals for this larger model. The BP results are qualitatively unaffected, but the 10-best estimator is notably worse: it does not support any tracks with marginal probability less than 0.3. The 100-best estimator performs only slightly better. This phenomenon is likely to arise in any large tracking scenario – not just those with completely separated subproblems – and highlights a significant weakness of the k-best approach to marginalization.

2.3 Additional probabilistic queries

In addition to marginalization, graphical models support efficient approximate inference algorithms for the tasks of MAP, m-best, diverse m-best, and mixed maximization/marginalization. This section briefly explores how these algorithms could be useful in the multi-target tracking context.

2.3.1 MAP estimation

In our experiments we computed the MAP data association, $\boldsymbol{\tau}^*$, using Morefield's integer program formulation [53] as described in Section 2.1. To solve the integer program we used CPLEX [36], a mathematical programming software suite by IBM ILOG. CPLEX is very fast and robust, implementing a wide array of optimization methods and heuristics. Correspondingly, the price for a non-academic license is quite high – a deployment license can cost upwards of $70,000. Formulating the track posterior as a factor graph opens up the alternative of solving the integer program using generic algorithms for MAP estimation in graphical models, many of which are quite simple to implement.

In a preliminary study, we implemented an algorithm based on the dual-decomposition framework [42], using a combined subgradient and coordinate ascent strategy [81] to optimize the resulting objective. On a set of test models generated by a TOMHT, the resulting algorithm was frequently able to identify the true MAP estimate (and provide a certificate of optimality) in time comparable to or less than CPLEX. The custom solver is around 500 lines of C++ and has not been heavily optimized.

2.3.2 m-best and diverse m-best

Beyond MAP estimation, efficient algorithms exist for approximating the m-best configurations of a graphical model [24, 6]. Recently, this approach was extended to allow computation of a *diverse* set of highly probable configurations [7]. A diverse set of likely hypotheses could be very useful for summarizing the uncertainty present in the track posterior distribution. In Section 2.2, we saw that the exact m-best hypotheses did a poor job of characterizing uncertainty; even for $m = 1000$, there is a high degree of similarity between top hypotheses, often differing only by one or two observations. By encouraging an appropriate measure

of diversity, we could instead recover a set of qualitatively different hypotheses that well represent the full space of possible data associations.

2.3.3 Marginal-MAP inference

Recently, the variational message-passing framework described in Section 1.4.1 was extended to support efficient approximate inference for marginal-MAP queries [49]. Marginal-MAP inference combines marginalization and maximization in a single probabilistic query. Given a random vector $\boldsymbol{X}$, split the variables into two sets $\boldsymbol{X}_A$ and $\boldsymbol{X}_B$. The marginal-MAP task is defined as follows:

$$\boldsymbol{x}_B^* = \arg\max_{\boldsymbol{X}_B} \sum_{\boldsymbol{X}_A} \Pr(\boldsymbol{X}_A, \boldsymbol{X}_B), \tag{2.22}$$

first marginalizing over the variables in $\boldsymbol{X}_A$ and then maximizing over the resulting distribution on $\boldsymbol{X}_B$.

Marginal-MAP inference could be useful for more effective pruning in the TOMHT. Recall the n-scan pruning procedure, described in Section 2.1.3. At each time step k, n-scan pruning picks a subset of nodes from time $k-n+1$ and prunes all tracks that are not ancestors of this protected set. Standard n-scan pruning chooses this protected set by finding the most likely hypotheses given all the data up to time k and then selecting its ancestor nodes at time $k-n+1$. A better method would be to directly optimize over the nodes at time $k-n+1$ while marginalizing out all subsequent nodes – a classic marginal-MAP query.

2.4 Summary of contributions

In this chapter we provide a new perspective on one of the most popular multi-target tracking algorithms. Our main contributions are:

- A factor graph representation of the TOMHT's probabilistic model, which enables the application of many approximate inference algorithms to the track posterior distribution.
- An empirical comparison of two approaches to estimating track marginal probabilities – one based on the k-best hypotheses, and one based on variational message-passing.

Chapter 3

Online Approximate EM for Parameter Estimation in the TOMHT

In the previous chapter we presented two approaches to estimating track marginals in the TOMHT. In this chapter, we make use of these marginals in an approximate EM algorithm to estimate parameters of the dynamics and observation models.

The probabilistic model for multi-target tracking, as described in Section 2.1, contains a number of tuning parameters. The observation model can often be tuned separately via offline analysis, but target dynamics and environmental parameters may require hand-labeled data and may change over time, making offline analysis impractical. As a result, it is desirable to estimate these parameters from *unlabeled* data and in an *online* fashion.

With labeled data – observations that have been grouped into ground truth tracks – parameter estimation is often carried out using the EM algorithm, treating the target states $\boldsymbol{X}$ as "missing data" [67]. We show that the same strategy can be used to estimate parameters from the unlabeled data typical of multi-target tracking. The E-step decomposes naturally

into two stages: computing marginal probabilities of the track indicator variables and computing smoothed state estimates for the candidate tracks. The M-step then proceeds as in the labeled case, with tracks weighted by their marginal probabilities.

3.1 Background: parameter estimation with known data associations

This section provides a brief introduction to parameter estimation in the case of known data associations. We first introduce the EM algorithm and then describe its popular application to learning in linear Gaussian state space models.

3.1.1 The Expectation-maximization algorithm

Expectation-maximization (EM) is a general algorithm for maximum likelihood estimation in the presence of missing data [20]. It can be viewed as a coordinate ascent algorithm, alternately updating a concave lower bound on the log-likelihood and choosing model parameters to maximize that bound. Here we provide the basic outline of the method – a more complete derivation can be found in [9, 50] (for example).

Suppose we have a joint probabilistic model over some observed variables $\boldsymbol{Z}$ and hidden variables $\boldsymbol{X}$, parameterized by a vector $\boldsymbol{\theta}$ (conventionally, $\boldsymbol{Z}$ is often used to represent hidden data and $\boldsymbol{X}$ observed data – we deviate from this convention to maintain consistency with the tracking application, where $\boldsymbol{Z}$ represents observations and $\boldsymbol{X}$ the latent target states). The goal is to compute the maximum likelihood estimate of $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \log \Pr(\boldsymbol{z} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log \sum_{\boldsymbol{x}} \Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}) \tag{3.1}$$

The term $\Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta})$ is called the complete-data log-likelihood (CDLL). Since it does not involve a high-dimensional sum, it is generally a much more tractable function than the log-likelihood itself.

To construct a lower bound, we recast Equation 3.1 as an expectation with respect to a variational distribution $q(\boldsymbol{x})$ on the latent variables. Applying Jensen's inequality to the result yields a lower bound, as follows:

$$\log \sum_{\boldsymbol{x}} \Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}) = \log \mathbb{E}_q \left[\frac{\Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta})}{q(\boldsymbol{x})} \right] \geq \mathbb{E}_q \left[\log \left(\frac{\Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta})}{q(\boldsymbol{x})} \right) \right] \tag{3.2}$$

Note that the variational distribution, $q(\boldsymbol{x})$, is a free parameter. Thus, each choice of $q(\boldsymbol{x})$ defines a different lower bound on the log likelihood. EM performs a joint optimization over both $q(\boldsymbol{x})$ and $\boldsymbol{\theta}$ via coordinate ascent. The "E-step" corresponds to optimization of the variational distribution. It is straightforward to show that, for a fixed $\boldsymbol{\theta}$, the optimal choice of q is given by

$$q(\boldsymbol{x}) = \Pr(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta}) \tag{3.3}$$

Holding q fixed, the "M-step" computes the maximum of the lower bound, which occurs at

$$\arg\max_{\boldsymbol{\theta}} \mathbb{E}_q \left[\log \Pr(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta} \right]. \tag{3.4}$$

Note that the M-step depends on q only through the expectation of the CDLL. In fact, when the CDLL is itself a linear or quadratic function of $\boldsymbol{x}$, the M-step depends on q only through its low-order moments (by the linearity of expectation). Because of this, the E-step is often defined to be the computation of the low-order moments of $\Pr(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\theta})$, rather than the full conditional distribution. The M-step then proceeds by plugging in the moments computed in the E-step and optimizing with respect to $\boldsymbol{\theta}$.

EM is guaranteed to converge to a local maximum of the likelihood function (global maximum if the likelihood is concave). In the non-concave setting, performance can depend heavily on the initial value of $\boldsymbol{\theta}$.

3.1.2 EM for linear Gaussian state-space models

With known data associations, the observations in a multi-target tracking scenario can be grouped into individual tracks and a set of false alarms. Given this grouping, it is trivial to compute maximum likelihood estimates of the parameters governing target detection, false alarms, and track birth and death. A more interesting task is optimizing the parameters of the observation and dynamics models. Since the likelihood of these parameters involves marginalization over the targets' latent states, the standard optimization approach uses EM, treating the target states as missing data. We now present this standard approach as described in [21]; in Section 3.2 we will show how to generalize the approach to the case of unknown data assocation.

With linear Gaussian observation and dynamics models, closed-form expressions exist for both the E-step and M-step. The probabilistic model for an individual track is a linear Gaussian state-space model, resulting in the following CDLL:

$$\begin{aligned}\log \Pr(\boldsymbol{z}, \boldsymbol{x}) = &-\frac{1}{2}\log|R| - \frac{1}{2}(z_1 - Hx_1)^\top R^{-1}(z_1 - Hx_1) \\ &+ \sum_{v=2}^{n}\Bigg[-\frac{1}{2}\log|R| - \frac{1}{2}(z_v - Hx_v)^\top R^{-1}(z_v - Hx_v) \\ &\qquad -\frac{1}{2}\log|Q| - \frac{1}{2}(x_v - Ax_{v-1})^\top Q^{-1}(x_v - Hx_{v-1})\Bigg]\end{aligned} \tag{3.5}$$

Due to the Markov dependence structure of the hidden variables, the E-step only requires

computation of the following moments:

$$\mathbb{E}\left[X_v\right],\ v = 1 \dots n, \tag{3.6}$$

$$\mathbb{E}\left[X_v X_v^\top\right],\ v = 1 \dots n, \tag{3.7}$$

$$\mathbb{E}\left[X_v X_{v-1}^\top\right],\ v = 2 \dots n, \tag{3.8}$$

where all expectations are taken with respect to $\Pr(\boldsymbol{x} \mid \boldsymbol{z})$. All of these can be computed efficiently via the Kalman smoothing algorithm, which is essentially belief propagation on a chain-structured Gaussian graphical model [67]. The M-step is then carried out with the following closed-form updates:

$$A^{new} = \frac{1}{n-1} \sum_{v=2}^{n} \mathbb{E}\left[X_v X_{v-1}^\top\right] \left(\frac{1}{n-1} \sum_{v=2}^{n} \mathbb{E}\left[X_{v-1} X_{v-1}^\top\right] \right)^{-1} \tag{3.9}$$

$$H^{new} = \frac{1}{n} \sum_{v=1}^{n} z_v \mathbb{E}\left[X_v^\top\right] \left(\frac{1}{n} \sum_{v=1}^{n} \mathbb{E}\left[X_v X_v^\top\right] \right)^{-1} \tag{3.10}$$

$$Q^{new} = \frac{1}{n-1} \sum_{v=2}^{n} \mathbb{E}\left[X_v X_v^\top\right] - A^{new} \left(\frac{1}{n-1} \sum_{v=2}^{n} \mathbb{E}\left[X_v X_{v-1}^\top\right] \right)^\top \tag{3.11}$$

$$R^{new} = \frac{1}{n} \sum_{v=1}^{n} z_v z_v^\top - H^{new} \left(\frac{1}{n} \sum_{v=1}^{n} z_v \mathbb{E}\left[X_v^\top\right] \right)^\top \tag{3.12}$$

The above updates assume no missed detections. Missed detections can be accommodated with only slight modifications, taking expectations with respect to the missing observations as well as the latent states. The algorithm also extends trivially to the case of multiple tracks, where each track shares the same observation and dynamics models.

3.2 Parameter estimation in the TOMHT

Section 3.1 described how to estimate dynamics and observation model parameters from known tracks. We now turn our attention to the case where tracks are not known. The only difference between this case and the previous is that now the likelihood involves marginalization over both hidden states and track indicator variables. This suggests using EM again, treating the track indicator variables as additional missing data. Pseudocode for this approach is provided in Algorithm 4. We now discuss the main steps of the algorithm in detail.

Recall the probabilistic model of Section 2.1. The corresponding CDLL is as follows:

$$\begin{aligned}\log \Pr(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\tau}) = \sum_{u=1}^{|\mathcal{T}|} \tau_u \Bigg[& \log \frac{\lambda_\nu}{\lambda_\phi} \\ & -\frac{1}{2}\log|R| - \frac{1}{2}(z_{\mathcal{T}_{u,1}} - Hx_{u,1})^\top R^{-1}(z_{\mathcal{T}_{u,1}} - Hx_{u,1}) \\ & + \sum_{v=2}^{|\mathcal{T}_u|} \Bigg(-\frac{1}{2}\log|Q| - \frac{1}{2}(x_{u,v} - Ax_{u,v-1})^\top Q^{-1}(x_{u,v} - Hx_{u,v-1}) \\ & + \mathbb{1}_{[\mathcal{T}_{u,v} \neq (*,0)]} \Big(-\frac{1}{2}\log|R| - \frac{1}{2}(z_{\mathcal{T}_{u,v}} - Hx_{u,v})^\top R^{-1}(z_{\mathcal{T}_{u,v}} - Hx_{u,v}) \\ & - \log\lambda_\phi + \log p_D \Big) + \mathbb{1}_{[\mathcal{T}_{u,v} = (*,0)]} \log(1 - p_D) \Bigg) \Bigg]\end{aligned} \tag{3.13}$$

where $x_{u,v}$ is the state of track u's target during its v^{th} scan of existence, conditioned on $T_u = 1$.

3.2.1 E-Step

Note the similarity between Equation 3.13 and Equation 3.5: aside from the fact that we are now dealing with a collection of tracks, the main substantive difference is the introduction

of the T_u variables. Generalizing from the single-track case, it is easy to see that the multi-target E-step requires the following expectations:

$$\mathbb{E}\left[T_u\right],\ u = 1 \ldots |\mathcal{T}|, \tag{3.14}$$

$$\mathbb{E}\left[T_u X_{u,v}\right],\ u = 1 \ldots |\mathcal{T}|,\ v = 1 \ldots |\mathcal{T}_u|, \tag{3.15}$$

$$\mathbb{E}\left[T_u X_{u,v} X_v^\top\right],\ u = 1 \ldots |\mathcal{T}|,\ v = 1 \ldots |\mathcal{T}_u|, \tag{3.16}$$

$$\mathbb{E}\left[T_u X_{u,v} X_{v-1}^\top\right],\ u = 1 \ldots |\mathcal{T}|,\ v = 2 \ldots |\mathcal{T}_u|, \tag{3.17}$$

where now expectations are taken with respect to $\Pr(\boldsymbol{x}, \boldsymbol{\tau} \mid \boldsymbol{z})$. Note that since T_u is an indicator variable, its expectation is simply its marginal probability, i.e., $\mathbb{E}\left[T_u\right] = \Pr(T_u = 1 \mid \boldsymbol{z})$. We will show that the latter three expectations can be decomposed into the product of the track marginal probability and a track-conditioned state moment.

Consider the more general expectation $\mathbb{E}\left[T_u g(\boldsymbol{X}_u)\right]$, where g is an arbitrary function of the states corresponding to track u. Note that this encompasses Equations 3.15-3.17 as special cases. We can decompose this expectation as follows:

$$\mathbb{E}\left[T_u g(\boldsymbol{X}_u)\right] = \int_{\boldsymbol{X}} \sum_{\tau_u \in \{0,1\}} \tau_u g(\boldsymbol{x}_u) \Pr(\boldsymbol{x}, \tau_u \mid \boldsymbol{z}) \tag{3.18}$$

$$= \int_{\boldsymbol{X}_u} g(\boldsymbol{x}_u) \Pr(\boldsymbol{x}_u, T_u = 1 \mid \boldsymbol{z}) \tag{3.19}$$

$$= \underbrace{\Pr(T_u = 1 \mid \boldsymbol{z})}_{(a)} \underbrace{\int_{\boldsymbol{X}_u} g(\boldsymbol{x}_u) \Pr(\boldsymbol{x}_u \mid T_u = 1, \boldsymbol{z})}_{(b)}. \tag{3.20}$$

The term marked (a) is the marginal probability of a track indicator, a quantity for which we have several tractable estimators (Section 2.2). The second term, (b), is an expectation of a target's state variables *conditioned on its particular track*. Computation of such moments amounts to single-target smoothing – a tractable, well understood problem. Since computation of the track-conditioned state moments is isolated from the data association

uncertainty, track marginals can be viewed as the key ingredient in a reduction from multi-target to single-target parameter estimation.

3.2.2 M-Step

After computing the expectations in Equations 3.14-3.17, the M-step proceeds analogously to the labeled data case, with the expectations weighted by the probabilities of their corresponding track:

$$A^{new} = \frac{1}{n''} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=2}^{|\mathcal{T}_u|} \mathbb{E}\left[X_{u,v} X_{u,v-1}^\top\right] \left(\frac{1}{n''} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=2}^{|\mathcal{T}_u|} \mathbb{E}\left[X_{u,v-1} X_{u,v-1}^\top\right] \right)^{-1} \tag{3.21}$$

$$H^{new} = \frac{1}{n'} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=1}^{|\mathcal{T}_u|} z_v \mathbb{E}\left[X_{u,v}^\top\right] \left(\frac{1}{n'} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=1}^{|\mathcal{T}_u|} \mathbb{E}\left[X_{u,v} X_{u,v}^\top\right] \right)^{-1} \tag{3.22}$$

$$Q^{new} = \frac{1}{n''} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=2}^{|\mathcal{T}_u|} \mathbb{E}\left[X_{u,v} X_{u,v}^\top\right] - A^{new} \left(\frac{1}{n''} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=2}^{|\mathcal{T}_u|} \mathbb{E}\left[X_{u,v} X_{u,v-1}^\top\right] \right)^\top \tag{3.23}$$

$$R^{new} = \frac{1}{n'} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=1}^{|\mathcal{T}_u|} z_v z_v^\top - H^{new} \left(\frac{1}{n'} \sum_{u=1}^{|\mathcal{T}|} b_u \sum_{v=1}^{|\mathcal{T}_u|} z_v \mathbb{E}\left[X_{u,v}^\top\right] \right)^\top, \tag{3.24}$$

where b_u is the marginal probability of track u, and

$$n' = \sum_{u=1}^{|\mathcal{T}|} b_u |\mathcal{T}_u| \tag{3.25}$$

$$n'' = \sum_{u=1}^{|\mathcal{T}|} b_u \left(|\mathcal{T}_u| - 1\right). \tag{3.26}$$

As before, the updates above assume no missed detections for simplicity but can easily be adapted to allow them.

Algorithm 4 EM for the Track-oriented MHT (TOMHT-EM)

procedure TOMHT-EM($\boldsymbol{z}^{1:T}$, *params*)
  $trackTrees \leftarrow \{\}$
  for $k = 1 \rightarrow T$ **do**
    ExtendTrees($trackTrees$, $\boldsymbol{z}^k$)
    Compute $\boldsymbol{\tau}^*$
    nscanPrune($trackTrees$, $\boldsymbol{\tau}^*$, n)
    $params \leftarrow$ UpdateParameters($trackTrees, params$)
    OutputStates($trackTrees$, $\boldsymbol{\tau}^*$)
procedure UpdateParameters($trackTrees$, *params*) // Perform online EM.
  for $i = 1 \rightarrow maxIter$ **do**
    $trackMarg \leftarrow$ EstimateMarginals($trackTrees$) // Section 2.2.
    $\mathcal{T}^{EM} \leftarrow$ SelectEMTracks($trackMarg$) // Section 3.2.4.
    $moments \leftarrow []$
    for $u = 1 \rightarrow |\mathcal{T}^{EM}|$ **do**
      $moments[u] \leftarrow$ KalmanSmoother($track$)
    $params \leftarrow$ mStep($trackMarg, moments$)
    UpdateTrackTrees($trackTrees, params$)
  return *params*
procedure UpdateTrackTrees($trackTrees$, *params*)
  for each $tree \in trackTrees$ **do**
    for each $node \in$ DFSTraverse($tree$) **do**
      $node.state \leftarrow$ KalmanFilter($node.parent.state, node.observation$)
      $node.score \leftarrow$ UpdateScore($node.parent.score, node.observation$)

3.2.3 Online updates

The TOMHT makes hard data association decisions while processing scans in sequence, so it is critical to update the system parameters online instead of waiting until all of the data has been processed. Since online EM is not the focus of this work, we implement a simple incremental batch approach in our experiments; in principle, one could also use a more sophisticated EM algorithm designed for online processing of dependent observations [14]. In our approach we perform 10 iterations of batch EM at each time step. When estimating a static parameter we use all scans processed up to the current time; when estimating a time-varying parameter (Section 3.3.4) we use only the most recent w scans, for some window size w. Due to the approximate track marginals used in the E-step, some iterations may decrease the likelihood. The hope is that the marginals are sufficiently accurate to produce good parameter estimates in spite of this approximation.

3.2.4 Truncated E-step

In our implementation, Kalman filtering and smoothing consume the bulk of the time spent running EM. Note that all of the track scores must be recomputed after each update of the parameters. Additionally, the E-step requires the computation of smoothed state estimates of every track. Many of these tracks have very low marginal probability, and as a result their contributions to Equations 3.21-3.24 are minimal.

We use a simple truncation strategy to reduce the computational requirements. Specifically, during the EM procedure we consider only those tracks whose marginal probability exceeds some threshold, ϵ. This selection criterion results in a new set of tracks, $\mathcal{T}^{EM}$:

$$\mathcal{T}^{EM} = \{\mathcal{T}_u \in \mathcal{T} : b_u > \epsilon\} \tag{3.27}$$

where, usually, $|\mathcal{T}^{EM}| \ll |\mathcal{T}|$.

The threshold ϵ can be set to a fixed value, e.g. $\epsilon = .001$, or allowed to vary as a function of the current set of tracks. One realization of a time-varying threshold strategy is to include in $\mathcal{T}^{EM}$ a fixed proportion p of the expected target detections:

$$\mathcal{T}^{EM} = \{\mathcal{T}_u \in \mathcal{T} : b_u > b^{(k)}\}, \tag{3.28}$$

where

$$k = \min \left\{ j : \sum_{i=j}^{|\mathcal{T}|} |\mathcal{T}_i| b^{(i)} > p \sum_{i=1}^{|\mathcal{T}|} |\mathcal{T}_i| b^{(i)} \right\} \tag{3.29}$$

and $\{b^{(i)}\}$ are the order statistics of the track marginal probabilities $\{b_u\}$. In our experiments we use a fixed threshold of $\epsilon = 0.01$.

PROPOSITION **3.1.** *The TOMHT-EM algorithm, using BP for track marginal estimation, has complexity* $\mathcal{O}((n + d_x^2 + d_z^2)nmp^n + 2^{nmp^n} + r(T(nmp^n + n^2m^2p^{n-2}) + knmp^{n-1}(d_x^3 + d_z^3)))$ *to process each scan, where* k *is current time step,* r *is the number of EM iterations at each time step,* T *is the maximum number of BP iterations,* n *is the parameter of* n*-scan pruning,* p *is the number of observations per gate, and* m *is the total number of observations per scan.*

Proof. It is shown in Proposition 2.1 that the TOMHT algorithm has complexity $\mathcal{O}((n+d_x^2+d_z^2)nmp^n+2^{nmp^n}+mk)$. TOMHT-EM differs only in the addition of the $UpdateParameters$ call, which we now analyze.

The E-step of EM requires estimating track marginals and running a Kalman smoother on a subset of tracks. The complexity of estimating track marginals depends on the particular algorithm used; with BP-MARG the complexity is $\mathcal{O}(T(nmp^n + n^2m^2p^{n-2}))$ (Proposition 2.4). Running a Kalman smoother on a single track of length k has complexity $\mathcal{O}(k(d_x^3 + d_z^3))$. Assuming BP-MARG uses a fixed fraction of all tracks, this leads to a com-

plexity of $\mathcal{O}(nmp^{n-1}k(d_x^3+d_z^3))$ for the smoothing. Thus, the total complexity for the E-step is $\mathcal{O}(T(nmp^n + n^2m^2p^{n-2}) + nmp^{n-1}k(d_x^3 + d_z^3))$.

The M-step (Equations 3.21-3.24) aggregates the moments computed in the E-step across all selected tracks. Again assuming we use a constant fraction of all candidate tracks, this step has a complexity of $\mathcal{O}(knmp^{n-1}(d_x^2 + d_z^2) + d_x^3)$.

EM is run for r iterations, and between each iteration the filtered state estimates and scores for all nodes in the track trees must be recomputed. There are $\mathcal{O}(nmp^{n-1})$ nodes, and each node update takes $\mathcal{O}(d_x^2+d_z^3)$ time, so the total cost of this operation is $\mathcal{O}(nmp^{n-1}(d_x^2+d_z^3))$.

Putting it all together, the complexity of TOMHT-EM is $\mathcal{O}((n + d_x^2 + d_z^2)nmp^n + 2^{nmp^n} + r(T(nmp^n + n^2m^2p^{n-2}) + knmp^{n-1}(d_x^3 + d_z^3)))$. □

3.3 Experimental results

In this section we evaluate the effect of online EM on tracker output. Unlike Chapter 2, we are not directly interested in the accuracy of our learning and inference algorithms. Instead, we aim to study their effectiveness at improving the quality of tracker output relative to the case where parameters are tuned offline and then fixed and treated as constant. To conduct this evaluation, we generate a dataset of simulated tracking scenarios and compare the output of our tracker under various configurations with the (simulated) ground truth tracks.

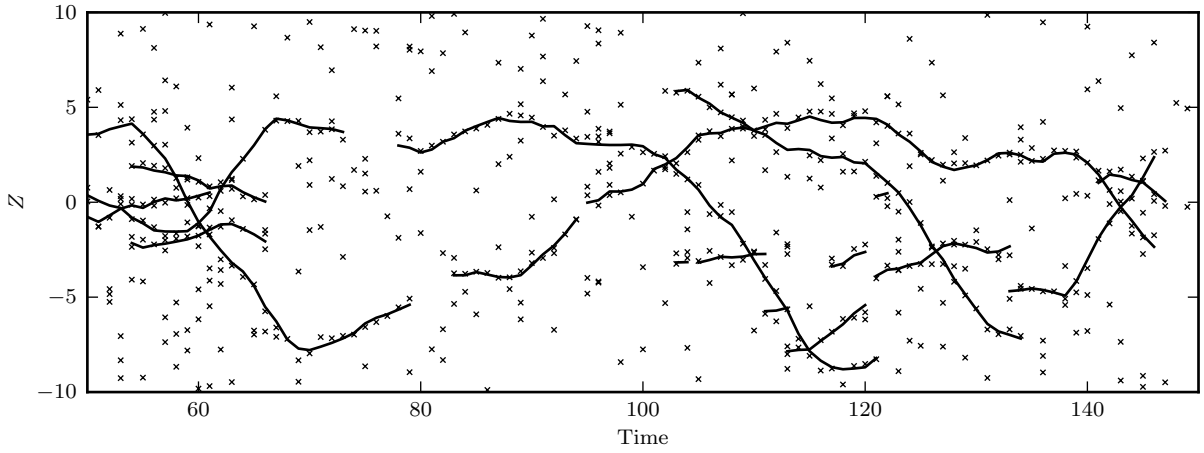


Figure 3.1: Part of a simulated scenario used to evaluate the impact of parameter estimation. The 'x's represent observations, and true target state trajectories are shown as solid lines.

3.3.1 Description of simulated data

For each simulated scenario we generate data using linear Gaussian dynamics and observation models, parameterized as follows:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} .1^2 & 0 \\ 0 & .2^2 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} .25^2 \end{bmatrix} \tag{3.30}$$

$$\lambda_\nu = .1 \qquad \lambda_\phi = 3 \qquad p_D = .95 \qquad p_\gamma = .05$$

This is an instance of the common "nearly-constant velocity" dynamics model, where the 2-dimensional state space represents the target's position and velocity and the observation space corresponds to target position. To encourage frequent track crossings, we slightly bias targets' velocity state components toward the origin. To achieve this we add additional state-dependent Gaussian random noise, with positive mean when velocity is negative and negative mean when the velocity is positive. We also cap the speed at 1 unit per time step. We generated 50 such scenarios, each 200 scans in duration. A portion of one simulated scenario is shown in Figure 3.1.

3.3.2 Evaluation of multi-target tracking output

Evaluating the output of a multi-target tracking system is a complex task in itself. There are many classes of potential error, including missed tracks, spurious tracks, merged or split tracks, and state estimation error. We use a metric called *optimal subpattern assignment for tracks* (TOSPA) to weight and combine all such errors into a single number summarizing the deviation of the tracker's filtered state trajectories from ground truth [63].

The TOSPA metric represents a track as a labeled sequence of states: $((s, x_1), (s, x_2), \ldots, (s, x_k))$. Here, s is an integer label for the track and does not change over time; x_i is the state of the track at time k. Given a set of tracks (either output from the tracker or the ground truth tracks), the set of all label/state pairs in existence at time t is called a snapshot. Let $\mathfrak{X}_t$ be the snapshot of the ground truth tracks at time t, and $\mathfrak{N}_t$ be the corresponding snapshot of the tracker output. Let the cardinality of these sets be m and n, respectively.

TOSPA defines a distance between these snapshots:

$$D_{p,c}(\mathfrak{X}_t, \mathfrak{N}_t) = \left[\frac{1}{n} \left(\min_{\pi \in \Pi_n} \sum_{i=1}^{m} \left(d_c \left(\mathfrak{X}_{t,i}, \mathfrak{N}_{t,\pi(i)} \right) \right)^p + (n-m)c^p \right) \right]^{\frac{1}{p}}, \tag{3.31}$$

where p and c are parameters of the metric, and Π_n is the set of all permutations of the integers from 1 to n. Here, d_c is called the truncated base distance between label/state pairs, and is defined as follows:

$$d_c\left((s,x),(t,y)\right) = \min\left(c, \left(||x-y||_{p'}^{p'} + \left(\alpha(1 - \mathbb{1}_{[s=t]})\right)^{p'} \right)^{\frac{1}{p'}} \right) \tag{3.32}$$

Note that d_c is itself parameterized by p' and α.

TOSPA error depends, in part, on the agreement of the labels between tracker output and ground truth tracks. These labels are assigned in a pre-processing step in a way that min-

imizes the average error across time steps. In the subsequent experiments, all reported TOSPA error values correspond to this time-averaged quantity rather than the error for a particular snapshot.

Essentially, TOSPA error can come from one of three sources: cardinality error, influenced by the parameter c; localization error, influenced by p' and α; and label-switching error, influenced by α. Equation 3.31 combines these sources of error into a single number, weighting the different types of error according its parameterization. In the following experiments, we use $c = 0.5$, $\alpha = 0.25$, and $p = p' = 2$. This results in a metric that is lower bounded by 0, upper bounded by 0.5, and penalizes label-switching errors half as severely as a cardinality error.

3.3.3 Recovery from poor initial model specification

It will often be the case that the parameters with which a tracker is run are not actually optimal with respect to the data it is processing. Although real data is not truly generated from a linear dynamical system, we can still consider some parameter setting *optimal* in the sense that it results in the highest quality tracker output with respect to an error metric of interest (in our case, TOSPA). The difference between a tracker's initial parameter setting and the parameter's optimal value is termed parameter misspecification.

Minor misspecification will not affect the data association decisions made by the TOMHT, so the only impact will be slightly degraded target state estimates. More significant misspecification has the potential to affect the optimal solution to Equation 2.10, leading to false tracks, split tracks, and missed tracks in addition to state estimation error. By reducing the level of misspecification as the tracker progresses, EM has the potential to mitigate errors of both types.

To test this hypothesis, we processed each of the 50 scenarios with seven different initial settings of Q. All initial settings of Q were diagonal, corresponding to independent noise on the position and velocity state components. We began with the covariance matrix used in the simulation process – call this matrix Q^*. Since states were perturbed to encourage track crossings these should not be thought of as "true" parameter values, but they are likely close to optimal since the perturbations were not large. From there we generated three covariance matrices with overestimates of the noise variance, doubling the standard deviations (SDs) each time. Finally, we generated three corresponding matrices underestimating the noise variance, halving the SDs rather than doubling them. The result is a set of covariance matrices indexed by the scaling factors shown along the x-axis of Figure 3.2.

For each scenario and initial value of Q we ran a TOMHT with four different EM configurations: (1) no EM; (2) EM using BP for marginalization; (3) EM using the 1-best estimator for marginalization; and (4) EM with 10-best. The runs were distributed across a cluster using GNU Parallel [72]. Comparing tracker performance under these four configurations allows us to determine whether online EM is useful and which marginalization algorithms work best in that context.

Figure 3.2 plots tracker error as a function of the initial parameter values for the four different EM configurations. Looking at the "No EM" line, we see that Q^* performs best, as expected. Without EM, initializing the noise SDs to either larger or smaller values leads to worse tracker performance. With online EM, however, the initial setting of Q has very little impact on tracker error, indicating that the parameters quickly converge to near-optimal values. All marginalization methods seem to perform about equally well, with the exception of the 1-best estimator which causes catastrophic failure with the largest initial noise SDs.

Figure 3.3 shows an example of how parameter misspecification and EM affect tracker output. The left-most and center ellipses highlight successes of EM. On the left, we see that inflated variance parameters cause the tracker to miss a track entirely, but with EM it is

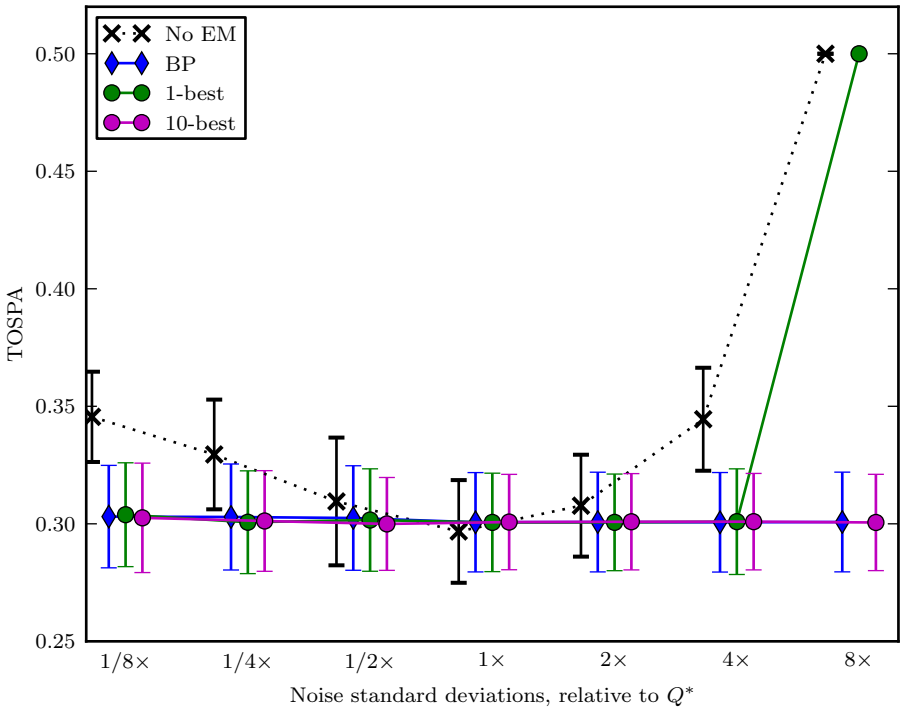


Figure 3.2: Online EM makes the tracker robust to misspecification of Q, preventing the increase in error observed in the static (No EM) case.

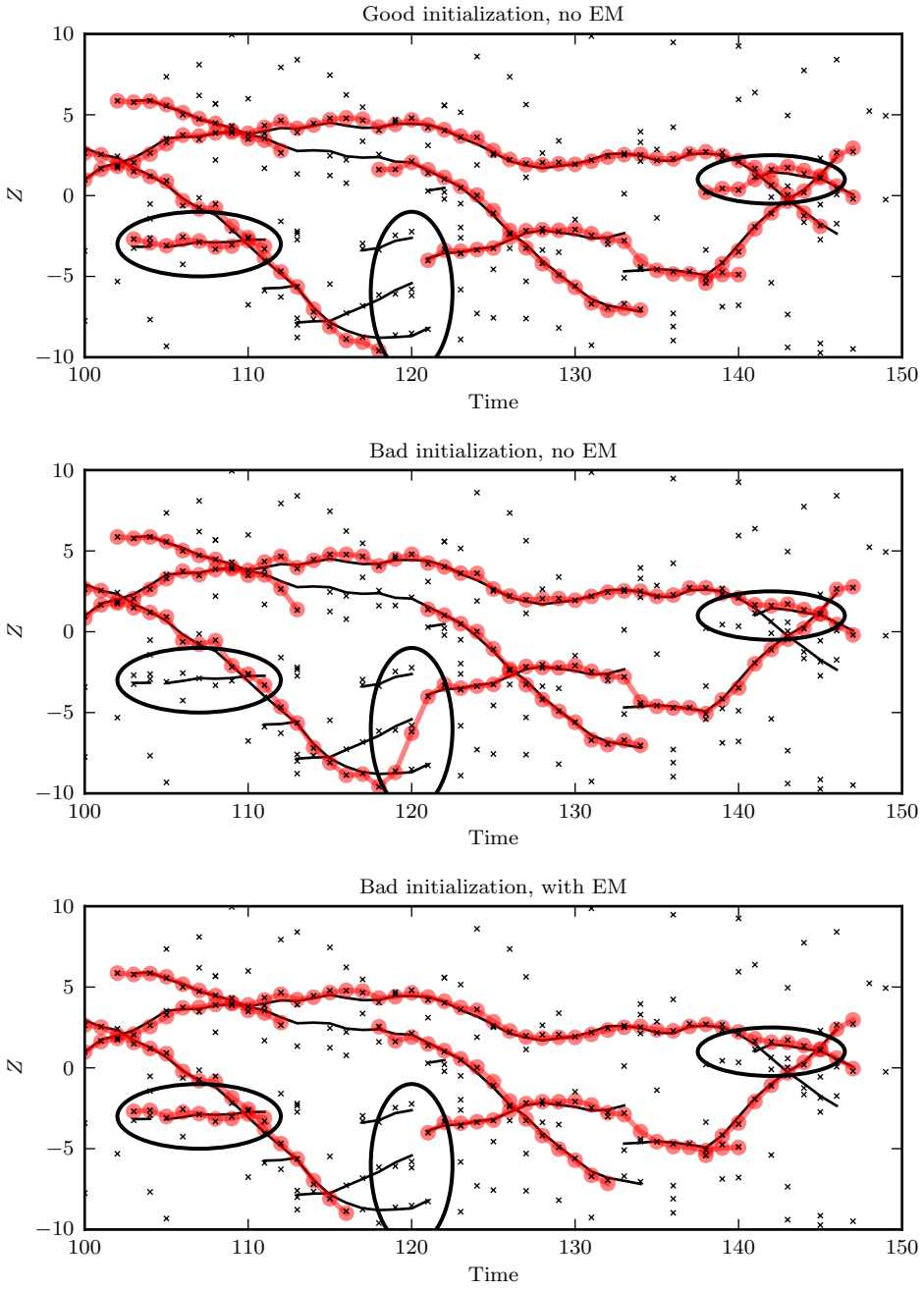


Figure 3.3: A detailed view of how tracker output is affected by parameter misspecification with and without online EM. (top) Tracker output on a portion of one scene when run with near optimal parameter settings and no online EM. Observations are plotted as 'x's, actual state trajectories with solid black lines, and the tracker's filtered state estimates with red lines marked by circles. (middle) Tracker output on the same scene resulting from a poor initializing (corresponding to the scaling of the noise SD) and no online EM. (bottom) Tracker output given the same initialization as in the middle figure, but with online EM.

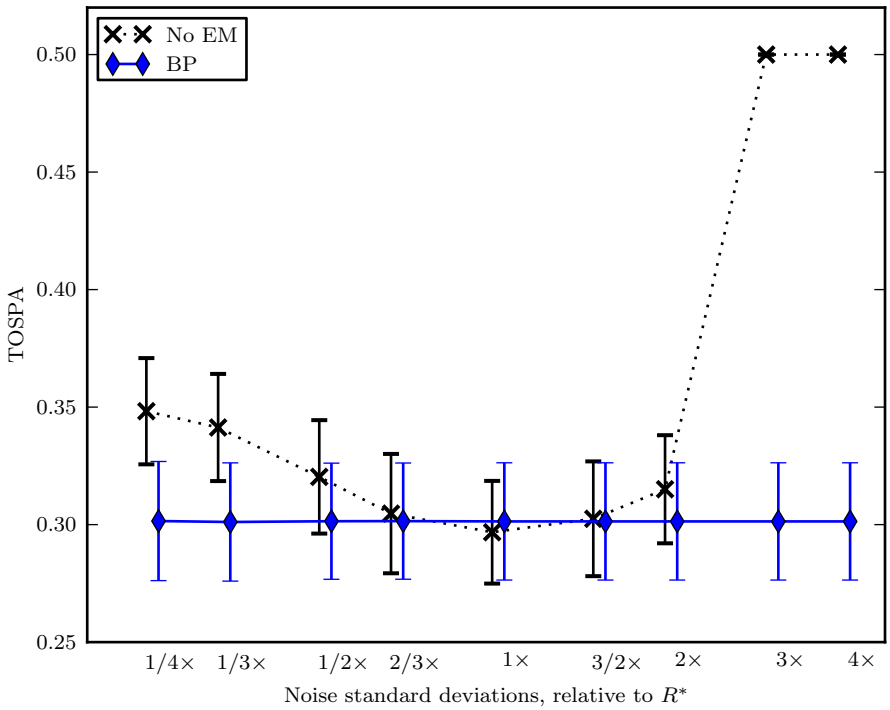


Figure 3.4: Online EM makes the tracker robust to initial misspecification of R, preventing the increase in error observed in the static (No EM) case.

able to recover and identify the track. The middle shows an instance where inflated variance parameters cause the tracker to join two separate tracks, whereas EM is able to reduce the variances and avoid this mistake. EM is not a panacea, however. The rightmost ellipse highlights a case where the static tracker with a good initialization makes the correct data association, parameter misspecification introduces an error, and EM fails to recover.

The EM algorithm developed in Section 3.1.1 allows us to estimate other parameters, as well. In Figure 3.4 we explore a range of misspecification values for the observation noise variance just as in the process noise case. The value used in the simulation was $R^* = [0.25^2]$, and we again consider several larger and smaller values by scaling the standard deviation. The results are qualitatively very similar to the case of misspecified Q. Varying the initial noise standard deviation from $1/4\times$ to $4\times$ the near-optimal value R^* causes severe performance degradation in the absence of online parameter estimation, but incorporating the online EM

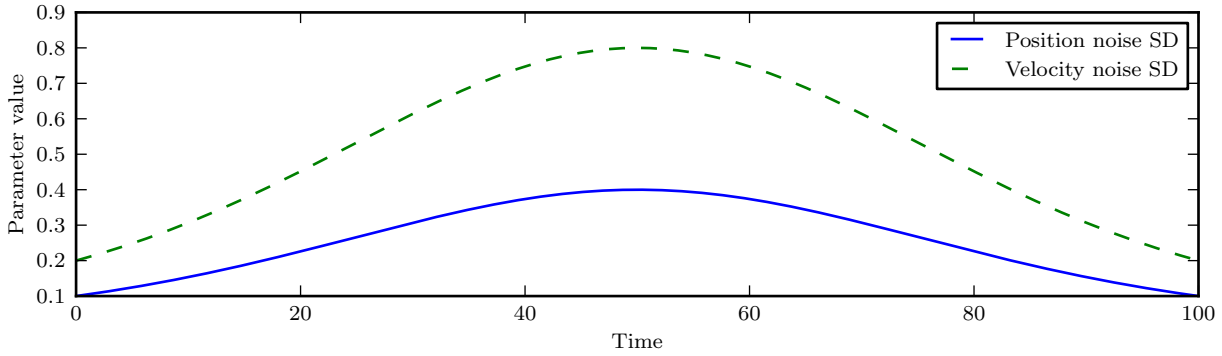


Figure 3.5: Position and velocity noise standard deviations over time. The parameters range from 1/4× to 4× the values used in the static parameter simulation of Section 3.3.3.

updates makes the tracker robust against this misspecification.

3.3.4 Tracking targets with time-varying dynamics

The previous section assumes that parameters remain constant over the course of a tracking scenario. As a result, a tracker with online EM can only ever do as well as a static-parameter tracker with optimal parameter settings. In practice, however, parameters may vary over time. In such a setting it is intuitive that a tracker with fixed parameters would perform worse than a tracker that dynamically adjusts its parameters to follow their optimal values.

To test this assumption we generated another set of simulated tracking scenarios. The generative procedure is identical to that described in Section 3.3.1, except now the process noise covariance matrix changes over time. The noise standard deviations were initialized with 0.1 and 0.2, as before, and then evolved in time according to an exponentiated sinusoid ranging from the 1/4× to 4× values from Section 3.3.3 with a period of 200 time steps. Figure 3.5 illustrates the evolution of the two standard deviations over time. When running the tracker on these scenarios with online parameter estimation, we restrict EM to consider only the most recent 10 scans as described in Section 3.2.3. Note that our simple online EM algorithm treats parameters as being constant within this window, which contradicts our

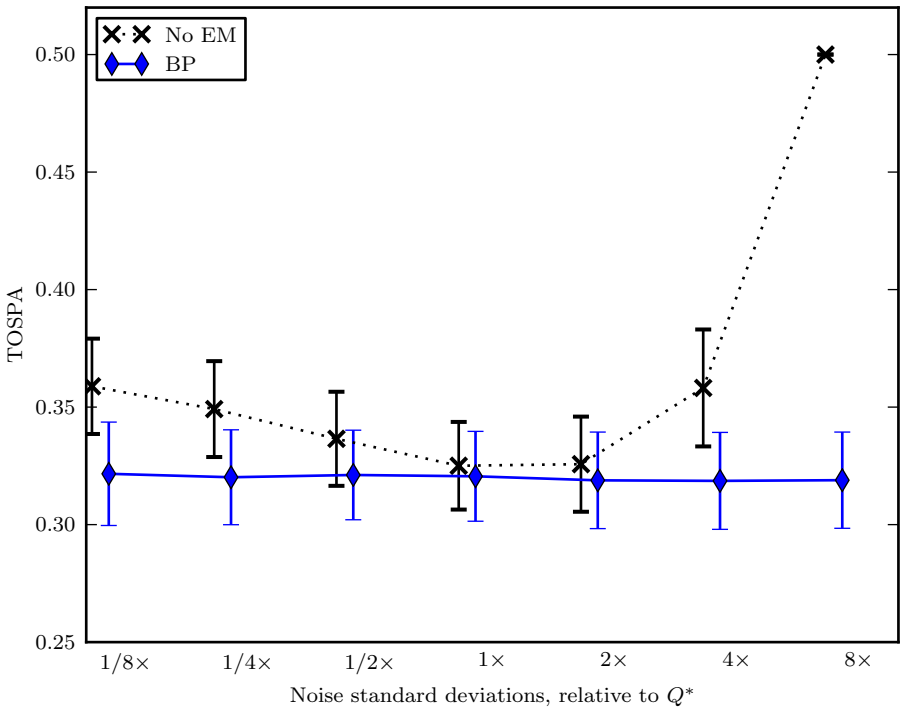


Figure 3.6: Tracking targets with time-varying dynamics.

assumption that parameters can vary from scan to scan. However, if the window is small and parameters change slowly over time, a parameter's value within the window will be well approximated by a constant.

Figure 3.6 plots TOSPA error as a function of initial parameter settings, both with and without online EM. Relative to Figure 3.2, both curves have been shifted "up" – the tracker performs worse than in the static parameter setting. As expected, tracker performance with online EM is slightly better than the best static parameter setting. However, the difference is quite small. This is due to the simplicity of our windowed batch EM algorithm, which introduces a lag in the estimated parameter values relative to their current optima. Explicit modeling the parameter values as time-varying quantities would almost certainly improve the online EM results. Even as it stands, though, Figure 3.6 demonstrates the potential for real-time parameter estimation based on approximate marginalization to improve tracker performance beyond what is possible with a standard, fixed-parameter tracker.

3.4 Summary of contributions

We have introduced a framework for online parameter estimation in the TOMHT based on the approximate track marginal estimators developed in the previous chapter. The resulting EM-enabled tracker is robust against parameter misspecification, and can actually perform better than the corresponding perfectly initialized static-parameter tracker if the true parameters are changing over time.

Our main contributions in this chapter are:

- We develop an EM algorithm for estimating parameters of the TOMHT model.
- We conduct experiments on simulated data, demonstrating the potential of this method to make the TOMHT robust to parameter misspecification.
- We compare performance of the EM-enabled TOMHT based on three of the track marginal estimators introduced in Chapter 2. We show that BP results in better performance than the 1-best estimator and comparable performance to the 10-best estimator.
- We conduct an evaluation of tracker performance on data simulated with time-varying parameters, and reveal that online EM in such cases can result in better performance than that of the best possible corresponding static-parameter tracker.

Chapter 4

Variational message-passing for continuous graphical models

Graphical models have proven to be an effective tool for representing the underlying structure of probability distributions and organizing the computations required for exact and approximate inference. Early examples of the use of graph structure for inference include join or junction trees [56] for exact inference, Markov chain Monte Carlo (MCMC) methods [26], and variational methods such as mean field and structured mean field approaches [37]. Belief propagation (BP), originally proposed by Pearl [56], has gained in popularity as a method of approximate inference, and in the last decade has led to a number of more sophisticated algorithms based on conjugate dual formulations and free energy approximations [82, 75, 68].

However, the progress on approximate inference in systems with continuous random variables has not kept pace with that for discrete random variables. Some methods, such as MCMC techniques, are directly applicable to continuous domains, while others such as belief propagation have approximate continuous formulations [69, 51]. Sample-based representations,

such as are used in particle filtering [3], are particularly appealing as they are relatively easy to implement, have few numerical issues, and have no inherent distributional assumptions. This chapter extends particle methods to take advantage of recent advances in approximate inference algorithms for discrete-valued systems.

Several recent algorithms provide significant advantages over loopy belief propagation. Double-loop algorithms such as CCCP [84] and UPS [74] use the same approximations as BP but guarantee convergence. More general approximations can be used to provide theoretical bounds on the partition function [75, 37] or are guaranteed to improve the quality of approximation [68], allowing an informed trade-off between computation and accuracy. Like belief propagation, they can be formulated as local message-passing algorithms on the graph, making them amenable to parallel computation [29] or inference in distributed systems [34, 65].

In short, the algorithmic characteristics of these recently-developed algorithms are often better, or at least more flexible, than those of BP. However, these methods have not been applied to continuous random variables, and in fact this subject was one of the open questions posed at a recent NIPS workshop [27].

In order to develop particle-based approximations for these algorithms, we focus on one particular technique for concreteness: tree-reweighted belief propagation (TRW) [75]. TRW represents one of the earliest of a recent class of inference algorithms for discrete systems, but as we discuss in Section 4.2 the extensions of TRW can be incorporated into the same framework if desired.

The basic idea of our approach is simple and extends previous particle formulations of exact inference [41] and loopy belief propagation [35]. We use collections of samples drawn from the continuous state space of each variable to define a discrete problem, "lifting" the inference task from the original space to a restricted, discrete domain in which TRW and other variational inference algorithms can be performed. At any point, the current results of the

discrete inference can be used to re-select the sample points from a variable's continuous domain. This iterative interaction between the sample locations and the discrete messages produces a dynamic discretization that adapts itself to the inference results.

We demonstrate that TRW and similar methods can be naturally incorporated into the lifted, discrete phase of particle belief propagation and that they confer similar benefits on the continuous problem as hold in truly discrete systems. To this end we measure the performance of the algorithm on an Ising grid, an analogous continuous model, and the sensor localization problem. In each case, we show that tree-reweighted particle BP exhibits behavior similar to TRW and produces significantly more robust marginal estimates than ordinary particle BP.

4.1 A review of inference methods for continuous graphical models

We begin with a brief overview of some of the most popular methods for inference in continuous graphical models. The methods are grouped according to the strategy they use for dealing with the intractability of exact operations on arbitrary continuous functions.

4.1.1 Special case: jointly Gaussian models

Gaussian MRFs [77] are an important special case of continuous graphical models. The analytical form of the Gaussian PDF enables efficient, exact computation of the standard sum-product messages updates. Furthermore, when BP converges on a Gaussian graphical model it is guaranteed to identify the exact mean of each variable [77]. Unfortunately, many models of interest involve continuous random variables that are not normally distributed.

These models require additional techniques to deal with the continuous space of values.

4.1.2 Discretization

The simplest approach to dealing with a continuous-valued graphical model is to convert it to a discrete graphical model via binning. Specifically, suppose we have a factor graph defined over bounded-domain real-valued variables $\boldsymbol{X} = [X_1 \cdots X_n]$. Assume for simplicity that all variables have the same domain, $X_i \in [a, b]$. Then we can define a new set of discrete variables, $\widehat{\boldsymbol{X}} = [\widehat{X}_1 \cdots \widehat{X}_n]$, such that the event $\widehat{X}_i = j$ approximates the event

$$a + \frac{b-a}{d}(j-1) < X_i < a + \frac{b-a}{d}(j) \tag{4.1}$$

for all j from 1 to d. Thus, $\widehat{X}_i$ represents a uniform discretization of X_i into d equally sized bins. The corresponding discrete factor graph also replaces the factors with discretized versions: $\widehat{f}_u(\widehat{\boldsymbol{x}}_{\boldsymbol{u}}) = f_u(\widetilde{\boldsymbol{x}_u})$, where $\widetilde{\boldsymbol{x}_u}$ is a real-valued vector with elements equal to the centers of the bins identified by $\widehat{\boldsymbol{x}}_{\boldsymbol{u}}$.

Using the simple approach described above, a continuous factor defined over variables $\boldsymbol{X}_u$ will contain $d^{|\boldsymbol{X}_u|}$ values once discretized. This causes the storage and computation complexity of most exact and approximate inference algorithms to have complexity polynomial in d. To mitigate the effect of quantization error, d must generally be fairly large. As a result, uniform discretization is generally practical only for pairwise graphical models.

More intelligent discretization schemes are also possible [43, 1]. For example, *continuously-adaptive discretization for message passing* (CAD-MP), performs a greedy, non-uniform discretization of each variable to minimize the K-L divergence between its piecewise-constant beliefs and the unquantized continuous beliefs [1]. This can greatly reduce the number of bins needed to accurately approximate continuous beliefs in higher dimensions, but it does not

offer any accuracy guarantees and it requires that all continuous factors be exactly integrable over arbitrary hyperrectangles.

4.1.3 Parametric approximation

Expectation propagation (EP) [51] deals with continuity by restricting the messages in a BP-like algorithm to have a tractable, exponential family form. By including an explicit projection step to ensure that no messages or beliefs grow too complex to compute, EP can perform fast, approximate inference in models where even standard sum-product would be intractable (high-dimensional mixed continuous/discrete models, for example). One downside of this approach is that it can be difficult to understand the impact of the various approximations that must be made to attain tractability. Also, while EP is a very general approach (encompassing algorithms like discrete tree-reweighted BP as special cases), its flexibility on any particular model is limited by the analytical forms of the factors present. In general, EP cannot tractably emulate an algorithm like tree-reweighted BP on a model with arbitrary continuous factors.

4.1.4 Kernel density estimation

Nonparametric belief propagation (NBP) [69] uses kernel density estimation to approximate continuous messages and beliefs. BP messages are represented as Gaussian mixtures and message products are approximated by drawing samples, which are then smoothed to form new Gaussian mixture distributions. A key aspect of this approach is the fact that the product of several mixtures of Gaussians is also a mixture of Gaussians, and thus can be sampled from with relative ease. However, it is difficult to see how to extend this algorithm to more general message-passing algorithms. For example, the TRW fixed point updates (Equation (1.38)) involve ratios and powers of messages, which do not have a simple form

for Gaussian mixtures and may not even form finitely integrable functions.

4.1.5 Importance sampling

Particle belief propagation (PBP) [35], the algorithm on which this chapter is founded, is a recent approach based on importance sampling. Whereas NBP represents each message with a mixture of Gaussians, PBP represents a message as a weighted mixture of delta functions. This parsimonious representation turns out to be quite flexible. The weights of the delta functions are chosen to enable unbiased importance sampling estimates of arbitrary expectations, including the integral in the BP factor message computation. The remainder of this section introduces the PBP algorithm.

First, we briefly review the concept of importance sampling. Suppose we have a random vector $\boldsymbol{X} \sim p(\boldsymbol{x})$ and some function of interest $f(\boldsymbol{x})$ for which we would like to compute the following expectation:

$$\mu = \mathbb{E}_p f(\boldsymbol{x}) = \int_{\boldsymbol{X}} f(\boldsymbol{x}) p(\boldsymbol{x}). \tag{4.2}$$

For arbitrary densities p and functions f, this expectation may be intractable. A standard approach in such cases is Monte Carlo integration [64], which approximates the integral with the average of N function evaluations corresponding to samples from p:

$$\widehat{\mu} = \frac{1}{N} \sum_{i=1}^{N} f\left(\boldsymbol{x}^{(i)}\right), \tag{4.3}$$

where $\boldsymbol{x}^{(i)} \sim p(\boldsymbol{x})$ is the i^{th} sample drawn from p. It is easy to see that $\widehat{\mu}$ is an unbiased estimator of μ. Further, the law of large numbers states that $\widehat{\mu} \to \mu$ as $N \to \infty$, and the central limit theorem reveals that, asymptotically, the standard deviation of $\widehat{\mu}$ decreases at a rate of $1/\sqrt{N}$ [64].

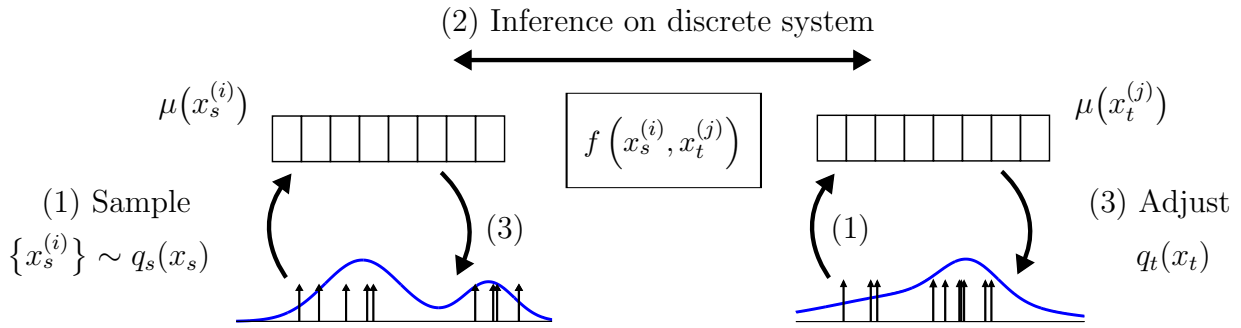


Figure 4.1: Schematic view of particle-based inference. (1) Samples for each variable provide a dynamic discretization of the continuous space; (2) inference proceeds by optimization or message-passing in the discrete space; (3) the resulting local functions can be used to change the proposals $q_s(x_s)$ and choose new sample locations for each variable.

Unfortunately, in many cases of practical interest it is infeasible to draw samples from $p(\boldsymbol{x})$, rendering direct Monte Carlo integration inapplicable. In still more cases, it is possible to draw samples from p but the variance of the resulting estimator may be very large, requiring an unacceptably large number of samples to produce good estimate. Importance sampling is a technique that can circumvent, or at least mitigate, these difficulties. Importance sampling reformulates the quantity μ as an expectation with respect to a convenient proposal distribution, q [64]. Specifically, let

$$\widehat{\mu}^{IS} = \frac{1}{N} \sum_{i=1}^{N} \frac{f\left(x^{(i)}\right) p\left(x^{(i)}\right)}{q\left(x^{(i)}\right)}, \tag{4.4}$$

where now the $\boldsymbol{x}^{(i)}$ are drawn from a proposal distribution $q(\boldsymbol{x})$. Note that $\widehat{\mu}^{IS}$ is the Monte Carlo approximation of the reweighted expectation $\mathbb{E}_q[f(\boldsymbol{x})p(\boldsymbol{x})/q(\boldsymbol{x})]$, which is equal to our original target quantity $\mathbb{E}_p[f(\boldsymbol{x})]$.

Particle BP uses importance sampling to approximate BP updates on continuous graphical models. At a high level, the procedure iterates between sampling particles from each variable's domain, performing inference over the resulting discrete problem, and adaptively updating the sampling distributions. This process is illustrated in Figure 4.1. Formally, we define an independent proposal distribution $q_s(x_s)$ for each variable X_s such that $q_s(x_s)$ is

non-zero over the domain of X_s. Let us index the variables that are neighbors of factor f_u as $\boldsymbol{X}_u = \{X_{u_1}, \ldots, X_{u_b}\}$. Then we can rewrite the standard BP factor message computation (Equation 1.27) as an importance reweighted expectation:

$$m_{f_u \to X_s}(x_s) \propto \mathop{\mathbb{E}}_{\boldsymbol{X}_u \setminus X_s} \left[f_u(\boldsymbol{x}_u) \prod_{X_t \in \boldsymbol{X}_u \setminus X_s} \frac{m_{X_t \to f_u}(x_t)}{q_t(x_t)} \right]. \tag{4.5}$$

After sampling N particles $\{x_s^{(1)}, \cdots, x_s^{(N)}\}$ from $q_s(x_s)$, we can index a particular assignment of particle values to the variables in $\boldsymbol{X}_u$ with $\boldsymbol{X}_u^{(\boldsymbol{j})} = [x_{u_1}^{(j_1)}, \ldots, x_{u_b}^{(j_b)}]$. We then obtain a finite-sample approximation of the factor message in the form

$$m_{f_u \to X_{u_k}}\left(x_{u_k}^{(j)}\right) \propto \frac{1}{N^{b-1}} \sum_{\boldsymbol{i}: i_k = j} \left[f_u\left(\boldsymbol{x}_u^{(\boldsymbol{i})}\right) \prod_{l \neq k} \frac{m_{X_{u_l} \to f_u}\left(x_{u_l}^{(i_l)}\right)}{q_{u_l}\left(x_{u_l}^{(i_l)}\right)} \right]. \tag{4.6}$$

In other words, PBP computes a Monte Carlo approximation to the integral using importance weighted samples from the proposal. Each of the values in the message then represents an estimate of the continuous function (4.5) evaluated at a single particle. Observe that the sum is over N^{b-1} elements, and hence the complexity of computing an entire factor message is $\mathcal{O}(N^b)$; this could be made more efficient at the price of increased stochasticity by summing over a random subsample of the vectors $\boldsymbol{i}$.

Variable messages and beliefs are computed as simple point-wise products, as in standard BP:

$$m_{X_s \to f_u}\left(x_s^{(j)}\right) \propto \prod_{f_v \in \boldsymbol{f}_s \setminus f_u} m_{f_v \to X_s}(x_s^{(j)}) \tag{4.7}$$

$$b_u\left(\boldsymbol{x}_u^{(\boldsymbol{j})}\right) \propto f_u(\boldsymbol{x}_u^{(\boldsymbol{j})}) \prod_{X_s \in \boldsymbol{X}_u} m_{X_s \to f_u}(x_s^{(j)}). \tag{4.8}$$

Algorithm 5 provides pseudocode for PBP.

Algorithm 5 Particle BP [35]

procedure PBP($\{f_u\}_{u=1}^m, q_s(x_s), N$)
  Draw particle sets: $\{x_s^{(i)}\}_{i=1}^N \sim q_s(x_s)$, s=1...n
  Initialize messages: $m_{f_u \to X_s}(x_s^{(j)}) = 1$, $m_{X_s \to f_u}(x_s^{(j)}) = 1 \; \forall f_u, X_s \in \boldsymbol{X}_u$
  $iter \leftarrow 1$
  while not converged and $iter < maxIter$ **do**
    $s \leftarrow$ NEXTVARIABLE() // Message schedule
    for each $f_u \in \boldsymbol{F}_s$ **do**
      $k \leftarrow$ INDEXOF($X_s, \boldsymbol{X}_u$) // X_{u_k} is X_s
      $m_{f_u \to X_{u_k}}(x_{u_k}^{(j)}) \propto 1/N^{b-1} \sum_{\boldsymbol{i}: i_k = j} f_u(\boldsymbol{x}_u^{(\boldsymbol{i})}) \prod_{l \neq k} m_{X_{u_l} \to f_u}(x_{u_l}^{(i_l)}) / q_{u_l}(x_{u_l}^{(i_l)}) \; \forall j$
    $q_s, \{x_s^{(i)}\}_{i=1}^N \leftarrow$ RESAMPLEPARTICLES(s) // Optional
    for each $f_u \in \boldsymbol{F}_s$ **do**
      $m_{X_s \to f_u}(x_s^{(j)}) \propto \prod_{f_v \in \boldsymbol{f}_s \setminus f_u} m_{f_v \to X_s}(x_s^{(j)}) \; \forall j$
    $iter \leftarrow iter + 1$
  end while
  for each $f_u \in \boldsymbol{F}$ **do**
    $b_u(x_u^{(\boldsymbol{j})}) \propto f_u(\boldsymbol{x}_u^{(\boldsymbol{j})}) \prod_{X_s \in \boldsymbol{X}_u} m_{X_s \to f_u}(x_s^{(j)}) \; \forall \boldsymbol{j}$

PROPOSITION **4.1.** *The PBP algorithm has time complexity $\mathcal{O}(TmbN^b)$, where T is the number of iterations, m is the number of factors, b is the maximal scope of any factor, and N is the number of particles per variable.*

Proof. The complexity of BP on a factor graph is dominated by the computation of factor messages. Computing a single factor message takes time $\mathcal{O}(d^b)$, where d is the maximum variable domain size and b is the maximum factor scope size. In PBP, the variable domain size d is replaced by a particle domain size, N, so the complexity becomes $\mathcal{O}(N^b)$. Each factor must send at least one message to each of its neighbors, which amounts to $\mathcal{O}(mb)$ messages. Usually a message must be re-computed multiple times prior to convergence, so the running time scales linearly with the number of iterations, T. Thus, the total time for the discrete message-passing phase of PBP is $\mathcal{O}(TmbN^b)$.

Aside from the discrete message-passing, PBP also includes sampling and resampling steps. The initial sampling takes time $\mathcal{O}(nN)$. Subsequent resampling complexity varies depending on how often it is performed and by which method. Here we consider the Metropolis-Hastings

strategy. To resample a single variable, we perform a constant number of M-H steps, each of which requires recomputing the incoming factor messages. One round of resampling over all variables thus takes time $\mathcal{O}(mbN^b)$. If resampling is performed after each full round of message updates, the total cost of resampling is $\mathcal{O}(TmbN^b)$.

Combining the complexities of the discrete message-passing and sampling phases gives an overall complexity of $\mathcal{O}(TmbN^b)$. □

Rao-Blackwellization

Quantities about X_s such as expected values under the pseudomarginal can be computed using the samples $\{x_s^{(i)}\}$. Note that for any given variable node X_s, the incoming messages to X_s given in (4.6) are defined in terms of the importance weights and sampled values of the neighboring variables. Thus, we can compute an estimate of the messages and beliefs defined in (4.6)–(4.8) at arbitrary values of X_s, simply by evaluating (4.6) at that point. This can be viewed as an instance of *Rao-Blackwellization* [12], conditioning on the samples at the neighbors of X_s and integrating over possible particle sets $\{X_s^{(i)}\}$.

Using this trick we can often get much higher quality estimates from the inference for small N. In particular, if the variable state spaces are sufficiently small that they can be discretized but the resulting factor domain size, d^b, is intractably large, we can evaluate (4.6) on the discretized grid for only $\mathcal{O}(dN^{b-1})$. More generally, we can substitute a larger number of samples $N' \gg N$ with cost that grows only linearly in N'.

For a more general application of Rao-Blackwellization in the context of sampling-based inference in graphical models, see [8].

Resampling and Proposal Distributions

Another critical point is that the efficiency of this procedure hinges on the quality of the proposal distributions q_s. Unfortunately, this forms a circular problem – q must be chosen to perform inference, but the quality of q depends on the distribution and its pseudomarginals. This interdependence motivates an attempt to learn the sampling distributions in an online fashion, adaptively updating them based on the results of the partially completed inference procedure. Note that this procedure depends on the same properties as Rao-Blackwellized estimates: that we be able to compute our messages and beliefs at a new set of points given the message weights at the other nodes.

Both [41] and [35] suggest using the current belief at each iteration to form a new proposal distribution. In [41], parametric density estimates are formed using the message-weighted samples at the current iteration, which form the sampling distributions for the next phase. In [35], a short Metropolis-Hastings MCMC sequence is run at a single node, using the Rao-Blackwellized belief estimate to compute an acceptance probability. A third possibility is to use a sampling/importance resampling (SIR) procedure, drawing a large number of samples, computing weights, and probabilistically retaining only N. Finally, one could draw samples from the current beliefs, as approximated by Rao-Blackwellized estimation over a fine grid of particles.

Finite sample behavior

Another attractive aspect of PBP is that it admits finite sample analysis of the error it introduces, at least on simple graphs. For example, it is shown in [35] that, on a tree-structured pairwise factor graph with k nodes and N particles per node, the following holds

simultaneously for all variables X_s and particles $x_s^{(i)}$ with probability $1 - \delta$:

$$\widehat{b}_s(x_s^{(i)}) \in b_s(x_s^{(i)}) \left(1 \pm \mathcal{O}\left(k\sqrt{N^{-1} R_q \log(kN/\delta)}\right)\right), \tag{4.9}$$

where $\widehat{b}_s$ is the belief at variable X_s computed by PBP, b_s is the belief that exact message-passing would produce, and R_q is a constant defined as follows:

$$R_q = \max_{\{u,s\}:X_s \in \boldsymbol{X}_u} \max_{\boldsymbol{X}_u} \frac{f_u(\boldsymbol{x}_u) \prod_{X_t \in \boldsymbol{X}_u \setminus X_s} m_{X_t \to f_u}(x_t)}{q_t(x_t) m_{f_u \to X_s}(x_s)}. \tag{4.10}$$

4.2 Extending particle belief propagation

Particle belief propagation provides a simple, flexible framework for dealing with continuity in the context of approximate marginalization. At a high level, it can be viewed as an interactive procedure with three stages:

1. Draw a particle set for each variable from its proposal distribution.
2. Run importance-reweighted message-passing over the particle domains.
3. Adjust the proposals to incorporate the results of step (2).

A key feature of PBP, for the purposes of this chapter, is the loose coupling between its approach to handling continuity and the underlying discrete inference algorithm. The only constraint that PBP places on the this algorithm is that it incorporate the particle importance weights when performing marginalization. Since this involves only a simple reweighting operation, there is significant flexibility in the choice of discrete inference algorithm. In this section we show how PBP enables the application of two variational message-passing algorithms – tree-reweighted BP and naive mean field – to continuous graphical models.

4.2.1 Tree-reweighted PBP

To apply TRW to continuous graphical models, we simply embed its message-passing updates in the discrete inference phase of PBP. Taking the particle importance weights into account, the message updates become:

$$m_{f_u \to X_{u_k}}\left(x_{u_k}^{(j)}\right) \propto \frac{1}{N^{b-1}} \sum_{\boldsymbol{i}: i_k = j} \left[f_u\left(\boldsymbol{x}_u^{(\boldsymbol{i})}\right)^{1/\rho_u} \prod_{l \neq k} \frac{m_{X_{u_l} \to f_u}\left(x_{u_l}^{(i_l)}\right)}{q_{x_{u_l}}\left(x_{u_l}^{(i_l)}\right)} \right] \tag{4.11}$$

$$m_{X_s \to f_u}\left(x_s^{(j)}\right) \propto \frac{\prod_{f_v \in F_s} m_{f_v \to X_s}\left(x_s^{(j)}\right)^{\rho_v}}{m_{f_u \to X_s}\left(x_s^{(j)}\right)} \tag{4.12}$$

$$b_s\left(x_s^{(j)}\right) \propto \prod_{f_u \in F_s} m_{f_u \to X_s}\left(x_s^{(j)}\right)^{\rho_u}. \tag{4.13}$$

This parallels the development in Section 4.1.5, except here we use factor weights $\boldsymbol{\rho}$ to compute messages according to TRW rather than standard loopy BP.

Just as in discrete problems, it is often desirable to obtain estimates of the log partition function for use in goodness-of-fit testing or model comparison. Our implementation of TRW-PBP gives us a stochastic estimate of an upper bound on the true partition function. Using other message passing approaches that fit into this framework, such as mean field, can provide a similar a lower bound. These bounds provide a possible alternative to Monte Carlo estimates of marginal likelihood [16].

To exactly evaluate the TRW upper bound on a continuous model, we use the same expression as in Equation 1.39 but with summations over variable domains replaced by integrals:

$$\begin{aligned} A^{TRW}(\boldsymbol{\theta}) = &\sum_{u=1}^{m} \int_{\boldsymbol{X}_u} \log f_u(\boldsymbol{x}_u) b_u(\boldsymbol{x}_u) - \sum_{u=1}^{m} \rho_u \int_{\boldsymbol{X}_u} b_u(\boldsymbol{x}_u) \log b_u(\boldsymbol{x}_u) \\ &+ \sum_{s=1}^{n} \left(1 - \sum_{f_u \in \boldsymbol{F}_s} \rho_u\right) \int_{X_s} b_s(x_s) \log b_s(x_s). \end{aligned} \tag{4.14}$$

Note that all of the integrals correspond to expectations with respect to a variable or factor belief. Since we can evaluate these beliefs at any point by requesting a new set of incoming messages (Section 4.1.5), one can directly approximate these integrals using importance sampling. Alternatively, one can construct a fine-grained piecewise-constant approximation to the beliefs, again using Rao-Blackwellized message computations, and compute exact expectations with respect to these approximate beliefs.

In either case, the resulting quantity is no longer a strict bound on the log-partition function due to approximations introduced both in the message-passing and in the bound computation, itself. Instead, this method computes a stochastic approximation $\widehat{A}^{TRW}(\boldsymbol{\theta})$ that converges to the true bound as $N \to \infty$.

4.2.2 Mean field PBP

Similarly, we can embed mean field updates within the PBP algorithm. The particle-based update equations are as follows:

$$m_{X_s \to f_u}\left(x_s^{(j)}\right) \propto \prod_{f_v \in \boldsymbol{F}_s} \exp\left(m_{f_v \to X_s}(x_s^{(j)})\right) \tag{4.15}$$

$$m_{f_u \to X_{u_k}}\left(x_s^{(j)}\right) \propto \frac{1}{N^{b-1}} \sum_{\boldsymbol{i}: i_k = j} \log f_u(\boldsymbol{x}_u^{(\boldsymbol{i})}) \prod_{l \neq k} m_{X_{u_l} \to f_u}\left(x_{u_l}^{(i_l)}\right) \tag{4.16}$$

$$b_s\left((x_s^{(j)}\right) \propto \prod_{f_u \in \boldsymbol{F}_s} \exp\left(m_{f_u \to X_s}\left(x_s^{(j)}\right)\right) \tag{4.17}$$

As with TRW-PBP, we can use the results of this procedure to compute a sample-based approximation to the mean field lower bound. Again, the resulting quantity $\widehat{A}^{MF}$ will not itself be a bound, but will converge to the lower bound A^{MF} as $N \to \infty$.

4.2.3 Primal bounds on the log-partition function

One of the advantages of methods like mean field and tree-reweighted BP is that they yield bounds on the log-partition function. As seen in Sections 4.2.1-4.2.2, the PBP framework extends this functionality to continuous graphical models, but it introduces a significant caveat: the "bounds" produced by TRW-PBP and MF-PBP are stochastic approximations of the true TRW and MF bounds. That is, due to error introduced by the approximate integration, the MF-PBP and TRW-PBP bounds are not actually guaranteed to bound the log-partition function except in the limit of infinite particles. Furthermore, since the bounds depend on beliefs (dual parameters) that must be estimated via iterative optimization procedures, it is difficult to quantify the difference introduced by the Monte Carlo integration in the PBP variants of the "bounds."

Weighted mini-bucket [48] offers an alternate means of bounding the log-partition function. The WMB bound (Equation 1.51) is computed from the primal form of the log-partition function, and computation of the bound can be viewed as variable elimination on a covering graph. This view is particularly convenient when we consider approximating the bound using PBP: the finite-sample analysis of PBP on tree-structured graphs [35] extends naturally to quantify the concentration of the approximate WMB-PBP bound around the true WMB bound.

Bernstein's Inequality

The primary inequality used in the analysis of PBP [35] is a variant of Bernstein's inequality: given n IID random variables $\{X_i\}_{i=1}^n$, if $0 \leq X_i \leq R\mathbb{E}[X_i]$ with probability 1, then with

probability at least $1 - \delta$ over the choice of values $x_1, \ldots, x_n$,

$$\frac{1}{n} \sum_{i=1}^{n} x_i \in \mathbb{E}[X_i](1 \pm \epsilon(R, n, \delta)) \tag{4.18}$$

where

$$\epsilon(R, n, \delta) = \sqrt{\frac{R}{n}} \left(\eta + \sqrt{\eta^2 + 2 \ln(2/\delta)} \right) \tag{4.19}$$

and

$$\eta = \frac{\ln(2/\delta)}{3} \sqrt{\frac{R}{n}}.$$

When $n \gg R$, this simplifies to $\epsilon \approx \sqrt{2 \ln(2/\delta) R/n}$.

Weighted mini-bucket PBP

We now apply this inequality to the message updates in WMB-PBP, resulting in a probabilistic bound on the difference between the actual WMB upper bound (denoted here as Z_u) and the stochastic WMB-PBP approximation to Z_u (denoted $\widehat{Z}_u$). With continuous variables, the WMB message update becomes:

$$m_{k \to l}(\bar{\boldsymbol{x}}_{s_{kl}}) = \left(\int_{\bar{x}_k} \left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) \prod_{j : k \in \bar{pa}(j)} m_{j \to k}(\bar{\boldsymbol{x}}_{s_{jk}}) \right)^{1/\bar{w}_k} \right)^{\bar{w}_k}. \tag{4.20}$$

Approximating the integral using importance sampling, as in PBP, yields the following WMB-PBP message update:

$$\widehat{m}_{k \to l}(\bar{\boldsymbol{x}}^{(\boldsymbol{h})}_{s_{kl}}) = \left(\frac{1}{m} \sum_{\boldsymbol{i} : \boldsymbol{i} \downarrow s_{kl} = \boldsymbol{h}} \frac{1}{\bar{q}_k(\bar{x}^{(i_1)}_k)} \left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}^{(\boldsymbol{i})}_{c_k}) \prod_{j : k \in \bar{pa}(j)} \widehat{m}_{j \to k}(\bar{\boldsymbol{x}}^{(\boldsymbol{i} \downarrow s_{jk})}_{s_{jk}}) \right)^{1/\bar{w}_k} \right)^{\bar{w}_k}, \tag{4.21}$$

where $\boldsymbol{i} \downarrow s_{kl}$ is the subvector of $\boldsymbol{i}$ corresponding to the variables in s_{kl}.

To apply Bernstein's inequality to the empirical expectation in the above formula, we must first show that the random variables being averaged have a finite upper bound, R_q. Let

$$R_q = \max_{(k,l):l\in\bar{pa}(k)} \max_{\bar{\boldsymbol{x}}_{c_l},\bar{\boldsymbol{x}}_{c_k}} \frac{1}{\bar{q}_k(\bar{x}_k) m_{k\to l}(\bar{\boldsymbol{x}}_{c_l})^{1/\bar{w}_k}} \left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) \prod_{j:k\in\bar{pa}(j)} m_{j\to k}(\bar{\boldsymbol{x}}_{c_k}) \right)^{1/\bar{w}_k}. \tag{4.22}$$

Then we have

$$\begin{aligned} &\frac{1}{\bar{q}_k(\bar{x}_k)} \left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) \prod_{j\in\bar{pa}(k)} m_{j\to k}(\bar{\boldsymbol{x}}_{c_k}) \right)^{1/\bar{w}_k} \\ &\quad \le R_q m_{k\to l}(\bar{\boldsymbol{x}}_{c_l})^{1/\bar{w}_k} \\ &\quad = R_q E_{\bar{x}\sim\bar{q}_k} \left[\frac{1}{\bar{q}_k(\bar{x}_k)} \left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}) \prod_{j:k\in\bar{pa}(j)} m_{j\to k}(\bar{\boldsymbol{x}}_{c_k}) \right)^{1/\bar{w}_k} \right], \end{aligned} \tag{4.23}$$

and we can use Bernstein's inequality to bound the error introduced by Equation 4.21. Using notation similar to [35], define $M_k(\bar{\boldsymbol{x}}_{c_k}) = \prod_{j:k\in\bar{pa}(j)} m_{j\to k}(\bar{\boldsymbol{x}}_{s_{jk}})$ for convenience.

Theorem 1. *For a mini-bucket tree with $\bar{n}$ nodes, if we sample m particles for each variable and compute the messages as defined by (4.21), then with probability at least $1-\delta$ over the choice of particles we have that:*

$$\begin{aligned} Z_u \left(1 - \bar{n}\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound-1}+1} \right) \right) \le \widehat{Z}_u \le \\ Z_u \exp\left(\bar{n}\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound-1}+1} \right) \right) \end{aligned} \tag{4.24}$$

where Z_u is the exact WMB upper bound, and $\widehat{Z}_u$ is the estimate resulting from the PBP approximation.

Proof. The proof follows the same structure as in [35]. First, apply a union bound to the

probability that the Bernstein inequality (4.18) holds for all $(\bar{n}-1)$ messages and m^{ibound} values of each message. With probability $1-\delta$, the following holds simultaneously for all messages $m_{k\to l}$ and particle sets $\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})}$:

$$\begin{aligned}\frac{1}{m}\sum_{\boldsymbol{i}:\boldsymbol{i}\downarrow s=\boldsymbol{h}}\frac{1}{q_k(\bar{x}_k^{i_1})}\left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}^{(\boldsymbol{i})})\widehat{M}_k(\bar{x}_{c_k}^{(\boldsymbol{i})})\right)^{1/\bar{w}_k}\\ \in m_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})^{1/\bar{w}_k}\left(1\pm\epsilon\left(R_q,m,\frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right)\end{aligned} \tag{4.25}$$

Exponentiating, we have:

$$\begin{aligned}\left(\frac{1}{m}\sum_{\boldsymbol{i}:\boldsymbol{i}\downarrow s=\boldsymbol{h}}\frac{1}{q_k(\bar{x}_k^{i_1})}\left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}^{(\boldsymbol{i})})\widehat{M}_k(\bar{x}_{c_k}^{(\boldsymbol{i})})\right)^{1/\bar{w}_k}\right)^{\bar{w}_k}\\ \in m_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\left(1\pm\epsilon\left(R_q,m,\frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right)^{\bar{w}_k}\end{aligned} \tag{4.26}$$

To make the size of this bound the same for every message, replace each of the weights with their upper bound, 1:

$$\begin{aligned}\left(\frac{1}{m}\sum_{\boldsymbol{i}:\boldsymbol{i}\downarrow s=\boldsymbol{h}}\frac{1}{q_k(\bar{x}_k^{i_1})}\left(\bar{f}_{c_k}(\bar{\boldsymbol{x}}_{c_k}^{(\boldsymbol{i})})\widehat{M}_k(\bar{x}_{c_k}^{(\boldsymbol{i})})\right)^{1/\bar{w}_k}\right)^{\bar{w}_k}\\ \in m_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\left(1\pm\epsilon\left(R_q,m,\frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right)\end{aligned} \tag{4.27}$$

Let d_k be the number of descendants of node k, including k itself, in the mini-bucket tree. Given (4.27) we can prove the following lower and upper bounds:

$$\widehat{m}_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\geq m_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\left(1-d_k\epsilon\left(R_q,m,\frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right) \tag{4.28}$$

$$\widehat{m}_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\leq m_{k\to l}(\bar{\boldsymbol{x}}_{s_{kl}}^{(\boldsymbol{h})})\exp\left(d_k\epsilon\left(R_q,m,\frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right) \tag{4.29}$$

The proof of (4.28)-(4.29) proceeds by induction exactly as in [35]. Using the same argument,

the product of all messages into the root node r is bounded by:

$$\widehat{M}_r(\bar{\boldsymbol{x}}_{c_r}^{(\boldsymbol{i})}) \geq M_r(\bar{\boldsymbol{x}}_{c_r}^{(\boldsymbol{i})}) \left(1 - (\bar{n}-1)\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right) \tag{4.30}$$

$$\widehat{M}_r(\bar{\boldsymbol{x}}_{c_r}^{(\boldsymbol{i})}) \leq M_r(\bar{\boldsymbol{x}}_{c_r}^{(\boldsymbol{i})}) \exp\left((\bar{n}-1)\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound}}\right)\right) \tag{4.31}$$

Finally, incorporating the probability that (4.18) *also* holds simultaneously for the final elimination in the root node, the following holds with probability $1-\delta$:

$$\widehat{Z}_u \geq Z_u \left(1 - \bar{n}\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound}+1}\right)\right) \tag{4.32}$$

$$\widehat{Z}_u \leq Z_u \exp\left(\bar{n}\epsilon \left(R_q, m, \frac{\delta}{(\bar{n}-1)m^{ibound}+1}\right)\right) \tag{4.33}$$

□

Note that this bound characterizes the gap between the $\widehat{Z}_u$ and Z_u, the exact WMB bound. Since $Z_u > Z$, we can also treat Equation 4.33 as a probabilistic upper bound on the true partition function, Z.

4.3 Experimental results

To illustrate that the improvements expected for discrete systems carry over into the continuous domain, we demonstrate the performance of our algorithm on two systems. In the first, we run the particle representation versions of the algorithm on a simple Ising model, first on a discrete system to show convergence toward the exact discrete algorithm, then on a

similar but continuous-valued system. We then use the algorithm on the sensor localization task described in [34], showing that the reweighted version can provide better estimates of uncertainty when the true marginals are multi-modal.

We explore the characteristics of TRW-PBP on three different models: the 2-D attractive Ising model, an analogous continuous system, and a sensor localization problem. Since we use TRW solely as a convenient stand-in for any of a wide array of discrete algorithms for approximate inference, we do not aim to achieve state-of-the-art performance on any real world problems. Rather, we study these three simple models to gain a qualitative understanding of the benefits this framework has to offer.

4.3.1 Case study: Ising-like models

The Ising model corresponds to a graphical model, typically a grid, over binary-valued variables with pairwise factors. Originating in statistical physics, similar models are common in many applications including image denoising and stereo depth estimation. Ising models are well understood, and provide a simple example of how BP can fail and the benefits of more general forms such as TRW. We initially demonstrate the behavior of our particle-based algorithms on a small (3×3) lattice of binary-valued variables to compare with the exact discrete implementations, then show that the same observed behavior arises in an analogous continuous-valued problem.

Ising model

Our factors consist of single-variable and pairwise functions, given by

$$f(x_s) = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \qquad f(x_s, x_t) = \begin{bmatrix} \eta & 1-\eta \\ 1-\eta & \eta \end{bmatrix} \tag{4.34}$$

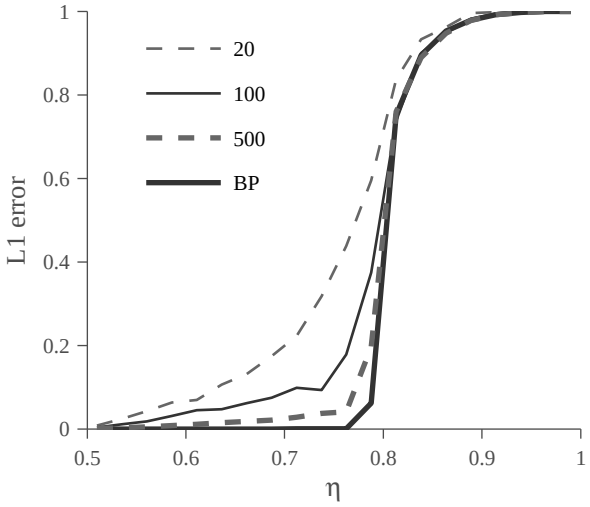
for $\eta > .5$. By symmetry, it is easy to see that the true marginal of each variable is uniform, [.5 .5]. However, around $\eta \approx .78$ there is a phase transition; the uniform-marginal fixed point becomes unstable and several others appear, becoming more skewed toward one state or another as η increases. As the strength of coupling in an Ising model increases, the performance of BP often degrades sharply, while TRW is comparatively robust and remains near the true marginals [75].

Figure 4.2 shows the performance of PBP and TRW-PBP on this model. Each data point represents the median L_1 error between the beliefs and the true marginals, across all nodes and 40 randomly initialized trials, after 50 iterations. The first plot (BP) clearly shows the phase shift; in contrast, the error of TRW remains low even for very strong interactions. In both cases, as N increases the particle versions of the algorithms converge to their discrete equivalents.

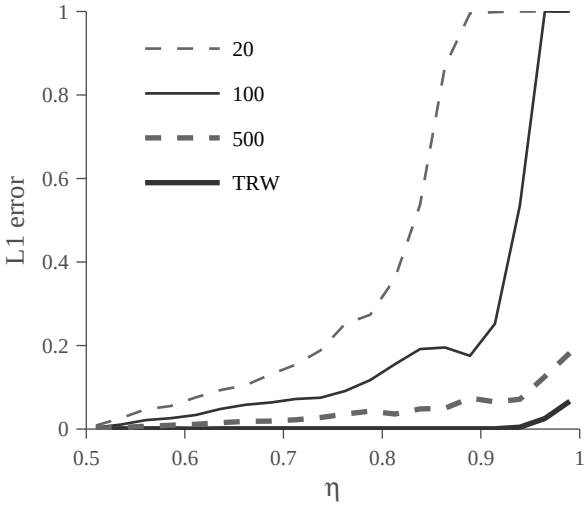
Continuous grid model

The results for discrete systems, and their corresponding intuition, carry over naturally into continuous systems as well. To illustrate on an interpretable analog of the Ising model, we use the same graph structure but with real-valued variables, and factors given by:

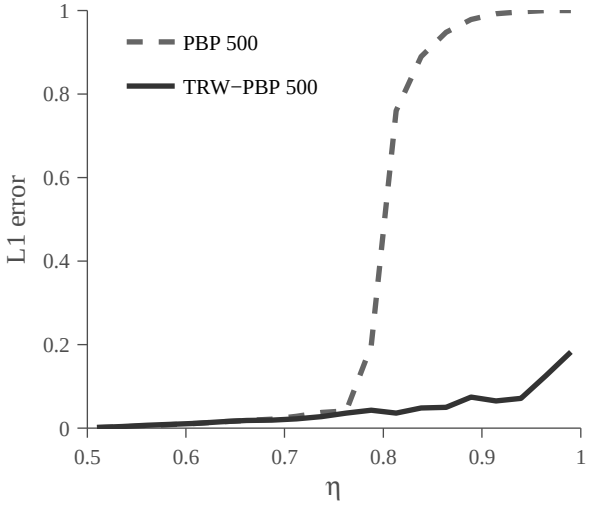
$$f(x_s) = \exp\left(-\frac{x_s^2}{2\sigma_l^2}\right) + \exp\left(-\frac{(x_s-1)^2}{2\sigma_l^2}\right) \quad f(x_s, x_t) = \exp\left(-\frac{|x_s - x_t|^2}{2\sigma_p^2}\right). \tag{4.35}$$



(a)

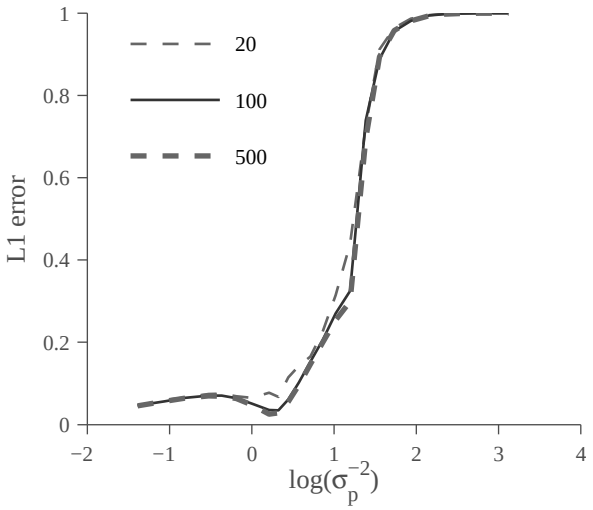


(b)

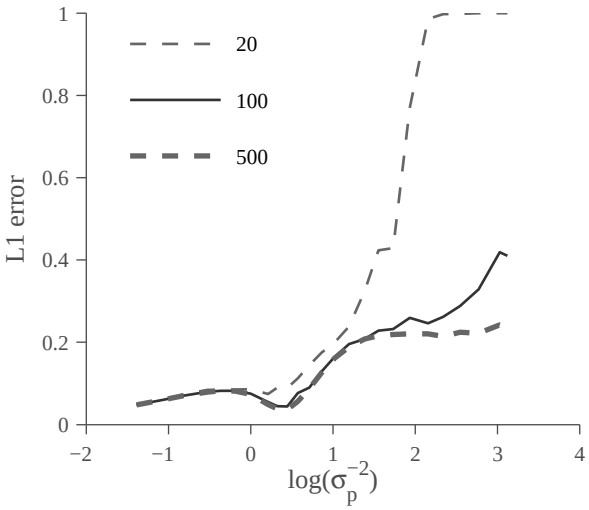


(c)

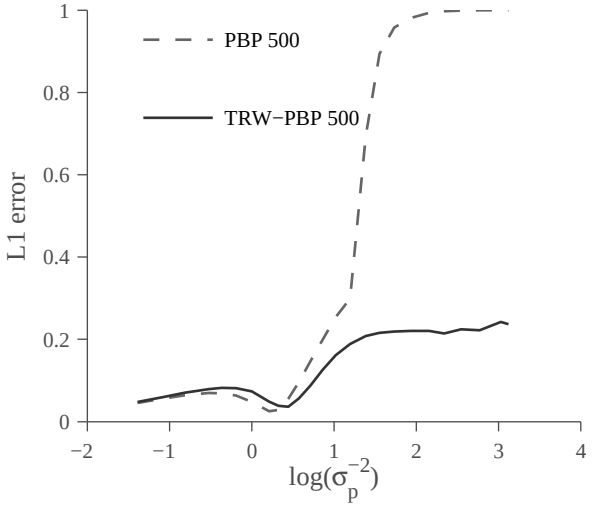
Figure 4.2: 2-D Ising model performance. L1 error for PBP (a) and TRW-PBP (b) for varying numbers of particles; (c) PBP and TRW-PBP juxtaposed to reveal the gap for high η.



(a)



(b)



(c)

Figure 4.3: Continuous grid model performance. L1 error for PBP (a) and TRW-PBP (b) for varying numbers of particles; (c) PBP and TRW-PBP juxtaposed to reveal the gap for low σ_p.

Local factors consist of bimodal Gaussian mixtures centered at 0 and 1, while pairwise factors encourage similarity using a zero-mean Gaussian on the distance between neighboring variables. We set $\sigma_l = 0.2$ and vary σ_p analogously to η in the discrete model. Since all potentials are Gaussian mixtures, the joint distribution is also a Gaussian mixture and can be computed exactly.

Figure 4.3 shows the results of running PBP and TRW-PBP on the continuous grid model, demonstrating similar characteristics to the discrete model. The first panel reveals that our

continuous grid model also induces a phase shift in PBP, much like that of the Ising model. For sufficiently small values of σ_p (large values on our transformed axis), the beliefs in PBP collapse to unimodal distributions with an L_1 error of 1. In contrast, TRW-PBP avoids this collapse and maintains multi-modal distributions throughout; its primary source of error (0.2 at 500 particles) corresponds to overdispersed bimodal beliefs. This is expected in attractive models, in which BP tends to "overcount" information leading to underestimates of variance; TRW removes some of this overcounting and may overestimate uncertainty.

As mentioned in Section 4.1.5, we can use the results of TRW-PBP to estimate the TRW upper bound on the log partition function and MF-PBP to estimate the MF lower bound. The resulting "bounds", computed for a continuous grid model in which mean field marginals collapse to a single mode, are shown in Figure 4.4. With sufficiently many particles, the values produced by TRW-PBP and MF inference bound the true value, as they should. With only 20 particles per variable, however, TRW-PBP occasionally fails and yields "upper bounds" below the true value. This is not surprising; the consistency guarantees associated with the importance-reweighted expectation take effect only when the number of particles is sufficiently large.

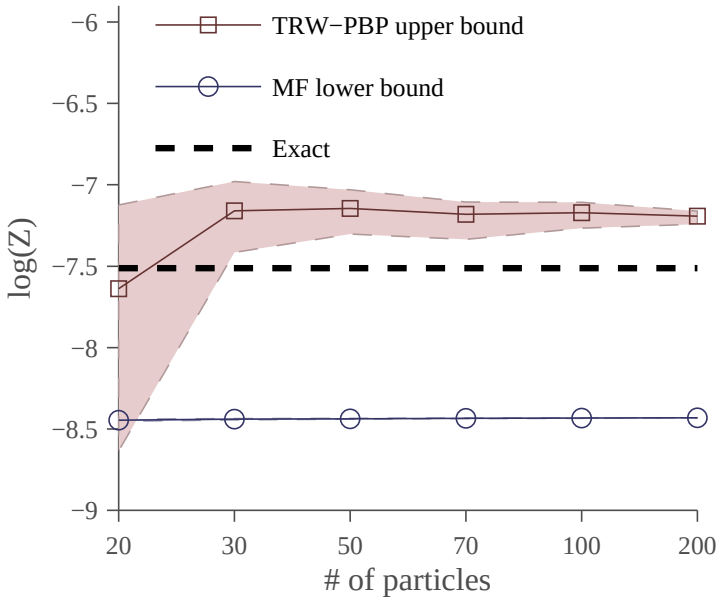


Figure 4.4: Bounds on the log partition function. Solid lines connect the median values and shaded regions indicate the inter-quartile ranges (the MF-PBP bounds had extremely low variance, so the shaded region is barely visible). TRW-PBP has high variance with fewer than 30 particles per variable, but quickly concentrates around the TRW bound as the number of particles increases.

4.3.2 Application to sensor self-localization

We also demonstrate the presence of these effects in a simulation of a real-world application. *Sensor self-localization* considers the task of estimating the position of a collection of sensors in a network given noisy estimates of a subset of the distances between pairs of sensors, along with known positions for a small number of *anchor nodes*. Typical localization algorithms operate by optimizing to find the most likely joint configuration of sensor positions. A classical model consists of (at a minimum) three anchor nodes, and a Gaussian model on the noise in the distance observations.

In [34], this problem is formulated as a graphical model and an alternative solution is proposed using nonparametric belief propagation to perform approximate marginalization. A significant advantage of this approach is that by providing approximate marginals, we can estimate the degree of uncertainty in the sensor positions. Gauging this uncertainty can be particularly important when the distance information is sufficiently ambiguous that the posterior belief is multi-modal, since in this case the estimated sensor position may be quite far from its true value. Unfortunately, belief propagation is not ideal for identifying multimodality, since the model is essentially attractive. BP may underestimate the degree of uncertainty in the marginal distributions and (as in the case of the Ising-like models in the previous section) collapse into a single mode, providing beliefs which are misleadingly overconfident.

Figure 4.5 shows a set of sensor configurations where this is the case. The distance observations induce a fully connected graph; the edges are omitted for clarity. In this network the anchor nodes are nearly collinear. This induces a bimodal uncertainty about the locations of the remaining nodes – the configuration in which they are all reflected across the crooked line formed by the anchors is nearly as likely as the true configuration. Although this example is anecdotal, it reflects a situation which can arise regularly in practice [52].

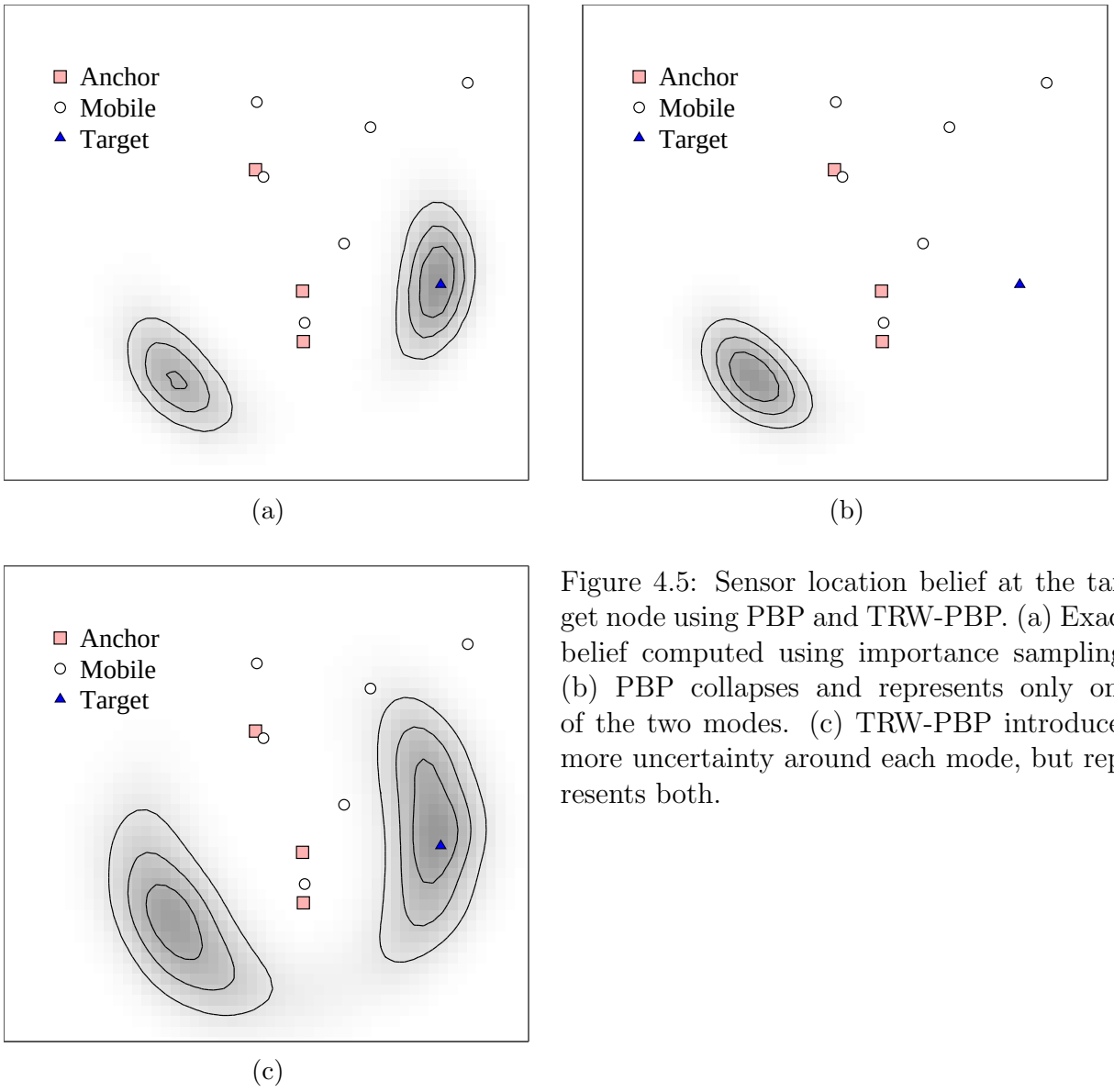


Figure 4.5: Sensor location belief at the target node using PBP and TRW-PBP. (a) Exact belief computed using importance sampling. (b) PBP collapses and represents only one of the two modes. (c) TRW-PBP introduces more uncertainty around each mode, but represents both.

Figure 4.5a shows the true marginal distribution for one node, estimated exhaustively using importance sampling with 5×10^6 samples. It shows a clear bimodal structure – a slightly larger mode near the sensor's true location and a smaller mode at a point corresponding to the reflection. In this system there is not enough information in the measurements to resolve the sensor positions. We compare these marginals to the results found using PBP.

Figure 4.5b displays the Rao-Blackwellized belief estimate for one node after 20 iterations of PBP with each variable represented by 100 particles. Only one mode is present, sug-

gesting that PBP's beliefs have "collapsed," just as in the highly attractive Ising model. Examination of the other nodes' beliefs (not shown for space) confirms that all are unimodal distributions centered around their reflected locations. It is worth noting that PBP converged to the alternative set of unimodal beliefs (supporting the true locations) in about half of our trials. Such an outcome is only slightly better; an accurate estimate of confidence is equally important.

The corresponding belief estimate generated by TRW-PBP is shown in Figure 4.5c. It is clearly bimodal, with significant probability mass supporting both the true and reflected locations. Also, each of the two modes is less concentrated than the belief in 4.5b. As with the continuous grid model we see increased stability at the price of conservative overdispersion. Again, similar effects occur for the other nodes in the network.

4.4 Summary of Contributions

We propose a framework for extending recent advances in discrete approximate inference for application to continuous systems. The framework directly integrates reweighted message passing algorithms such as TRW into the lifted, discrete phase of PBP. Furthermore, it allows us to iteratively adjust the proposal distributions, providing a discretization that adapts to the results of inference, and allows us to use Rao-Blackwellized estimates to improve our final belief estimates.

The main contributions in this chapter are:

- We introduce a general framework for extending variational message-passing algorithms to work on continuous graphical models via PBP.
- For the case of TRW, we demonstrate experimentally that its qualitative characteris-

tics (overdispersed marginals, bound on the partition function) carry over directly to continuous problems.

- We provide a finite sample analysis of the WMB-PBP primal bound on the partition function.
- In a simulated sensor self-localization problem, we demonstrate that TRW-PBP represents uncertainty more accurately than PBP when the true marginal distributions are multimodal.

Chapter 5

Conclusion

To conclude, we provide a brief recap of the dissertation's main contributions and explore some directions for future research.

5.1 Summary of contributions

Chapter 2 explored a new application of graphical models and approximate inference to the multi-target tracking domain. We developed a factor graph formulation of the track posterior distribution of the TOMHT, which allows us to efficiently approximate the marginal probabilities of the many possible tracks. In experiments on simulated data we compared the accuracy of several track marginal estimators. The results revealed that BP led to more accurate marginals than a k-best hypothesis estimator, particularly on larger models. GBP was able to increase accuracy over BP at the cost of increased computation time.

Chapter 3 developed an EM algorithm for parameter estimation in the TOMHT model. The algorithm depends on track marginals, which effectively reduce the multi-target E-

and M-steps to weighted versions of the familiar single-target EM algorithm. Plugging in approximate marginals computed via one of the methods introduced in Chapter 2 results in an efficient, approximate EM algorithm. We showed experimentally that this approximate EM algorithm with BP-derived marginals was effective at increasing tracker robustness to parameter misspecification and adapting to moderate nonstationarity in target dynamics.

Chapter 4 presents a framework for extending discrete variational inference algorithms to work on continuous-valued graphical models. We showed that PBP provides a natural separation between the continuous value space of a model and the underlying algorithm used to perform inference. This separation allows us to "plug in" a wide variety of recent variational inference algorithms that until now had no continuous analog. We demonstrated experimentally, both on an Ising-like grid and in a simulated sensor localization problem, that PBP-TRW confers similar benefits to inference in continuous-valued models as in discrete systems. We showed that PBP variants of TRW and MF could be used to approximate the corresponding upper and lower bounds, and also showed how WMB-PBP provides a stochastic *primal* bound on the partition function, which particularly useful for continuous-valued models where we cannot exactly optimize the dual.

5.2 Future directions

Here we briefly discuss some interesting directions for future research related to the work of this dissertation.

5.2.1 More compact representations of data association hypothesis space

In Chapter 2 we refer to the superiority of the TOMHT over its predecessor, the HOMHT, which stems from its ability to compactly represent a much larger set of hypotheses. The price of this representational power is the increased computational expense required to perform inference, both optimization (via the integer program) or marginalization (as explored in Chapter 2). The benefit is a larger space of candidate hypotheses. It is possible to take this trade-off to the extreme, using a representation so compact that *no* pruning is required (beyond gating) for the price of even more intractable inference. Some existing work has begun exploring this direction. One body of work formulates the MAP data association problem as a multidimensional assignment problem and solves it approximately using a Lagrangian relaxation approach [57, 58]. Another, the Bayesian Network Tracking Database (BNTD) [55], performs a kind of approximate inference in a model defined jointly over a compact data association space and target state space. The success of these compact representations is heavily dependent on the accuracy of a suitable approximate inference algorithm. It would be interesting to explore the application of standard variational inference algorithms to the model used in the BNTD. Also, joint modeling of target states and data association variables will tend to introduce multimodality into target state densities; another interesting extension would be to estimate target states nonparametrically, e.g. using PBP.

5.2.2 More complex probabilistic queries for the TOMHT

As mentioned in Section 2.3, the factor graph representation of the TOMHT admits additional queries beyond marginalization. Mixed inference queries [49] could be useful to improve pruning or, perhaps, used interactively to marginalize over specific segments of

space and time and optimize within a specific region of interest. Similarly, the diverse m-best modes [**?**] could both help improve pruning (by preserving a diverse set of candidates rather than collapsing into a set of hypotheses very similar to the MAP), or to provide easily interpretable output to a human operator. How to clearly convey the uncertainty in a multi-target data association posterior is a difficult problem in itself, and a combination of mixed inference and m-best mode queries could potentially provide a much richer summary than the standard MAP estimate.

5.2.3 Particle belief propagation on region graphs

As seen in Section 2.2.3, moving from BP on a factor graph to GBP on a region graph can result in significantly more accurate inference. Particle BP, unfortunately, is only defined for factor graphs. An extension of particle BP that operates on region graphs would provide another significant step forward for general-purpose inference algorithms on continuous-valued models. The main technical difficulty associated with this extension is the resampling procedure – regions have overlapping scopes, and the particle sets for each pair of regions must share the same values for the variables they have in common. One potential approach is to use a single joint proposal distribution rather than sampling each region separately, similar to a technique used for discrete models in the 2012 Pascal Approximate Inference Competition [28].

5.3 Parting thoughts

The rich probabilistic structure in multi-target tracking makes it well suited to the application of graphical models and approximate inference methods. Early approaches to multi-target tracking made severe modeling assumptions to obtain efficient, exact inference.

Since then computational resources have increased, enabling more complex and accurate approaches like the TOMHT. Simultaneously approximate inference algorithms have grown more capable, but the impact of these advances on multi-target tracking has not yet been fully explored. This dissertation demonstrates the ability of BP and GBP to efficiently approximate track marginals within the standard TOMHT model. Going forward, we expect that the flexibility granted by recent approximate inference algorithms will enable new approaches based on still more complex models.

Bibliography

[1] K. Achan, M. Isard, and J. MacCormick. Continuously-adaptive discretization for message-passing algorithms. *Advances in Neural Information Processing Systems (NIPS)*, pages 737–744, 2008.

[2] E. Allman, C. Matias, and J. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A):3099–3132, 2009.

[3] M. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[4] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29(6):82–100, 2009.

[5] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley-Interscience, New York, NY, 2004.

[6] D. Batra. An efficient message-passing algorithm for the m-best MAP problem. *Uncertainty in Artificial Intelligence (UAI)*, pages 121–130, 2012.

[7] D. Batra and P. Yadollahpour. Diverse m-best solutions in markov random fields. *Computer VisionECCV . . .*, 2012.

[8] B. Bidyuk and R. Dechter. Cutset sampling for Bayesian networks. *Journal of Artificial Intelligence Research*, 28(1):1–47, 2007.

[9] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[10] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, Jan. 2004.

[11] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, Norwood, MA, 1999.

[12] D. Blackwell. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, 18:105–110, 1947.

[13] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[14] O. Cappé. Online EM algorithm for hidden Markov models. *Journal of Computational and Graphical Statistics*, 20(3):728–749, Jan. 2011.

[15] L. Chen, M. J. Wainwright, M. Çetin, and A. S. Willsky. Data association based on optimization in graphical models with application to sensor networks. *Mathematical and Computer Modelling*, 43(9-10):1114–1135, May 2006.

[16] S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321, 1995.

[17] R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Uncertainty in Artificial Intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc., 1996.

[18] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[19] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, Mar. 2003.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[21] V. Digalakis, J. Rohlicek, and M. Ostendorf. ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442, 1993.

[22] A. Ess, K. Schindler, B. Leibe, and L. Van Gool. Object detection and tracking for autonomous navigation in dynamic environments. *International Journal of Robotics Research*, 29(14):1707–1725, 2010.

[23] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, July 1983.

[24] M. Fromer and A. Globerson. An LP view of the m-best MAP problem. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pages 567–575, 2009.

[25] A. Gelfand, K. Kask, and R. Dechter. Stopping rules for randomized greedy triangulation schemes. In *AAAI Conference on Artificial Intelligence*, pages 54–60, 2011.

[26] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, Nov. 1984.

[27] A. Globerson, D. Sontag, and T. Jaakkola. *Approximate inference – How far have we come? (NIPS'08 Workshop)*, 2008.

[28] V. Gogate, P. Domingos, and R. Dechter. Structured propagation-based and sampling-based algorithms for graphical models (Pascal approximate inference competition), 2012.

[29] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*, pages 177–184, Clearwater Beach, FL, 2009.

[30] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 2 edition, 1988.

[31] T. Heskes. Stable fixed points of loopy belief propagation are local minima of the Bethe free energy. In *Advances in Neural Information Processing Systems (NIPS)*, pages 343–350, 2002.

[32] P. Horridge and S. Maskell. Searching for, initiating and tracking multiple targets using existence probabilities. In *Information Fusion (FUSION)*, pages 611–617, 2009.

[33] A. Ihler. Accuracy bounds for belief propagation. In *Uncertainty in Artificial Intelligence (UAI)*, pages 183–190, 2007.

[34] A. Ihler, J. Fisher, and A. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6:905–936, 2006.

[35] A. Ihler and D. McAllester. Particle Belief Propagation. In D. van Dyk and M. Welling, editors, *Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 256–263, Clearwater Beach, FL, Apr. 2009. JMLR: W&CP.

[36] ILOG. CPLEX: High-performance software for mathematical programming and optimization, 2013.

[37] M. Jordan, Z. Ghahramani, and T. Jaakkola. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[38] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree decompositions for reasoning in graphical models. *Artificial Intelligence*, 166(1-2):165–193, Aug. 2005.

[39] K. Kask, A. Gelfand, L. Otten, and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *AAAI Conference on Artificial Intelligence*, pages 1043–1048, 2011.

[40] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, first edition, 2009.

[41] D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Uncertainty in Artificial Intelligence (UAI)*, pages 324–333, 1999.

[42] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization & beyond via dual decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011.

[43] A. Kozlov and D. Koller. Nonuniform dynamic discretization in hybrid networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 314–325, 1997.

[44] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[45] T. Kurien. Issues in the design of practical multitarget tracking algorithms. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*, volume 1, pages 43–83. Artech House, 1990.

[46] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, pages 282–289, 2001.

[47] H. Leung, M. Blanchette, and J. Litva. An efficient decentralized multiradar multitarget tracker for air surveillance. *IEEE Transactions on Aerospace and Electronic Systems*, 33(4):1357–1363, Oct. 1997.

[48] Q. Liu and A. Ihler. Bounding the partition function using Hölder's inequality. In *International Conference on Machine Learning (ICML)*, pages 849–856, 2011.

[49] Q. Liu and A. Ihler. Variational algorithms for marginal MAP. In *Uncertainty in Artificial Intelligence (UAI)*, pages 453–462, 2011.

[50] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, Hoboken, NJ, second edition, 2008.

[51] T. Minka. Divergence measures and message passing. *Microsoft Research Cambridge, Tech. Rep. MSR-TR-2005-173*, 2005.

[52] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 50–61, 2004.

[53] C. L. Morefield. Application of 0-1 integer programming to multitarget tracking problems. *IEEE Transactions on Automatic Control*, 22(3):302–312, June 1977.

[54] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Uncertainty in Artificial Intelligence (UAI)*, pages 467–475, 1999.

[55] F. H. Obermeyer and A. B. Poore. A Bayesian network tracking database. *Signal and Data Processing of Small Targets*, 5428:400–418, 2004.

[56] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, 1988.

[57] A. B. Poore. Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking. *Computational Optimization and Applications*, 3(1):27–57, Mar. 1994.

[58] A. B. Poore and A. J. Robertson III. A new Lagrangian relaxation based algorithm for a class of multidimensional assignment problems. *Computational Optimization and Applications*, pages 129–150, 1997.

[59] R. L. Popp, K. R. Pattipati, and Y. Bar-Shalom. m-best SD assignment algorithm with application to multitarget tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 37(1):22–39, 2001.

[60] G. Pulford. Taxonomy of multiple target tracking methods. *IEE Proceedings - Radar, Sonar and Navigation*, 152(5):291–304, 2005.

[61] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[62] X. Ren, Z. Huang, D. Liu, and J. Wu. Multiple object video tracking using GRASP-MHT. *Information Fusion (FUSION)*, pages 330–337, 2012.

[63] B. Ristic, B.-N. Vo, D. Clark, and B.-T. Vo. A metric for performance evaluation of multi-target tracking algorithms. *IEEE Transactions on Signal Processing*, 59(7):3452–3457, July 2011.

[64] C. P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer, New York, second edition, 2004.

[65] J. Schiff, D. Antonelli, A. Dimakis, D. Chu, and M. Wainwright. Robust message-passing for statistical inference in sensor networks. In *Information Processing in Sensor Networks (IPSN)*, pages 109–118, Apr. 2007.

[66] D. Schulz, W. Burgard, and D. Fox. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research*, 22(2):99–116, 2003.

[67] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[68] D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1393–1400. MIT Press, Cambridge, MA, 2007.

[69] E. Sudderth, A. Ihler, W. Freeman, and A. Willsky. Nonparametric belief propagation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 605–612, 2003.

[70] C. Sutton and A. McCallum. Improved dynamic schedules for belief propagation. In *Uncertainty in Artificial Intelligence (UAI)2*, pages 376–383, 2007.

[71] D. Svensson, J. Wintenby, and L. Svensson. Performance evaluation of MHT and GM-CPHD in a ground target tracking scenario. In *Information Fusion (FUSION)*, pages 300–307, 2009.

[72] O. Tange. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*, pages 42–47, Feb. 2011.

[73] D. Tarlow, K. Swersky, R. Zemel, R. Adams, and B. Frey. Fast exact inference for recursive cardinality models. In *Uncertainty in Artificial Intelligence*, pages 825–834, Oregon, 2012. AUAI Press Corvallis.

[74] Y. Teh and M. Welling. The unified propagation and scaling algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 953–960, 2002.

[75] M. Wainwright, T. Jaakkola, and A. Willsky. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335, 2005.

[76] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[77] Y. Weiss and W. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, 2001.

[78] M. Welling, A. Gelfand, and A. Ihler. A cluster-cumulant expansion at the fixed points of belief propagation. In *Uncertainty in Artificial Intelligence (UAI)*, pages 883–892. AUAI Press Corvallis, 2012.

[79] W. Wiegerinck and T. Heskes. Fractional belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 438–445, 2002.

[80] J. L. Williams and R. A. Lau. Data association by loopy belief propagation. In *Information Fusion (FUSION)*, pages 1–8, 2010.

[81] J. Yarkony, C. Fowlkes, and A. Ihler. Covering trees and lower-bounds on quadratic assignment. In *Computer Vision and Pattern Recognition (CVPR)*, pages 887–894, 2010.

[82] J. S. Yedidia and W. T. Freeman. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

[83] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–236. Morgan Kaufmann, San Francisco, CA, first edition, 2003.

[84] A. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, 2002.

[85] N. Zhang and D. Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.

[86] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1):45–57, 2001.

Appendix A

Derivation of the TOMHT Track Posterior Distribution

This derivation closely follows the development in [45] and is provided here for convenience.

To derive the track posterior distribution in 2.7, it is convenient to begin with an alternate representation of data association space. Let ω^k be a categorical variable that specifies whether each observation in scan k is a false alarm, the beginning of a new track, or a continuation of a preexisting track. If an observation continues an existing track, w^k also specifies *which* track it continues. Thus, $\Pr(\boldsymbol{w}^{1:k} \mid \boldsymbol{z}^{1:k})$ is an alternate representation of the data association posterior for scans 1 through k.

The first step is to decompose the posterior, pulling out the most recent scan:

$$\Pr(\omega^{1:k} \mid \boldsymbol{z}^{1:k}) \propto \Pr(\boldsymbol{z}^k \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}) \Pr(\omega^{1:k} \mid \boldsymbol{z}^{1:k-1}) \tag{A.1}$$

$$= \Pr(\boldsymbol{z}^k \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}) \Pr(\omega^k \mid \omega^{1:k-1}, \boldsymbol{z}^{1:k-1}) \Pr(\omega^{1:k-1} \mid \boldsymbol{z}^{1:k-1}) \tag{A.2}$$

$$= \Pr(\boldsymbol{z}^k \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}) \Pr(\omega^k \mid \omega^{1:k-1}) \Pr(\omega^{1:k-1} \mid \boldsymbol{z}^{1:k-1}) \tag{A.3}$$

Note that equation (A.3) decomposes the posterior into the product of three terms: a likelihood term for the current scan, a prior term on the number of births/missed detections/false alarms at the current scan, and a recursive term for historical data. Next we examine the first two terms in greater detail.

The current-scan likelihood factors as a product over all observations in the scan. The terms of this product take different functional forms depending on whether ω^k indicates that they are continuing an existing track, initiating a new track, or a false alarm. Thus, we can write

$$\Pr(\boldsymbol{z}^k \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}) = \prod_{j=1}^{m_k} f(z^{k,j} \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}), \tag{A.4}$$

$$\text{where } f(z^{k,j} \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1}) = \begin{cases} \Pr(z^{k,j} \mid \omega^{1:k-1}, \boldsymbol{z}^{1:k-1}) & \text{if } z^{k,j} \text{ continues an existing track.} \\ 1/V & \text{if } z^{k,j} \text{ begins a new track.} \\ 1/V & \text{if } z^{k,j} \text{ is a false alarm.} \end{cases} \tag{A.5}$$

where V is the volume of the surveillance region.

The second term in (A.3) represents the prior probability on the number of false alarms, missed detections, and new tracks in the current scan. Since we this term is not conditioned on $\boldsymbol{z}^k$, all values of ω^k with the same number of false alarms, missed detections, and new tracks are equally likely. Given our assumptions of Poisson distributed numbers of new tracks

and clutter, as well as a Bernoulli model for track death, we may write:

$$\Pr(\omega^k \mid \omega^{1:k-1}) = \frac{\frac{\lambda_\nu^{N_\nu(\omega^k)} e^{-\lambda_\nu}}{N_\nu(\omega^k)!} \frac{\lambda_\phi^{N_\phi(\omega^k)} e^{-\lambda_\phi}}{N_\phi(\omega^k)!} \binom{N_T(\omega^{k-1})}{N_D(\omega^k)} (p_D)^{N_D(\omega^k)} (1-p_D)^{N_{\neg D}(\omega^k)}}{\binom{m_k}{N_\nu(\omega^k)} \binom{m_k - N_\nu(\omega^k)}{N_\phi(\omega^k)} \binom{m_k - N_\nu(\omega^k) - N_\phi(\omega^k)}{N_D(\omega^k)} \binom{N_T(\omega^{k-1})}{N_D(\omega^k)} N_D(\omega^k)!} \tag{A.6}$$

$$= \frac{\frac{N_T(\omega^{k-1})!}{N_D(\omega^k)!(N_T(\omega^{k-1}) - N_D(\omega^k))!} \frac{\lambda_\nu^{N_\nu(\omega^k)} e^{-\lambda_\nu}}{N_\nu(\omega^k)!} \frac{\lambda_\phi^{N_\phi(\omega^k)} e^{-\lambda_\phi}}{N_\phi(\omega^k)!} (p_D)^{N_D(\omega^k)} (1-p_D)^{N_{\neg D}(\omega^k)}}{\frac{m_k!}{N_\nu(\omega^k)!(m_k - N_\nu(\omega^k))!} \frac{(m_k - N_\nu(\omega^k))!}{N_\phi(\omega^k)! N_D(\omega^k)!} \frac{N_T(\omega^k)!}{(N_T(\omega^{k-1}) - N_D(\omega^k))!}} \tag{A.7}$$

$$= \frac{1}{m_k!} \left(\lambda_\nu^{N_\nu(\omega^k)} e^{-\lambda_\nu}\right) \left(\lambda_\phi^{N_\phi(\omega^k)} e^{-\lambda_\phi}\right) (p_D)^{N_D(\omega^k)} (1-p_D)^{N_{\neg D}(\omega^k)} \tag{A.8}$$

where $N_\nu(\omega^k)$ and $N_\phi(\omega^k)$ are the numbers of false alarms and detections according to ω^k, $N_D(\omega^k)$ and $N_{\neg D}(\omega^k)$ are the number of detected and missed targets, respectively, $N_T(\omega^{k-1})$ is the number of preexisting tracks according to ω^{k-1}, m_k is the number of observations in scan k, and the denominator of (A.6) is the total number of assignment vectors with the same number of new targets, false alarms, and detections as in ω^k.

Substituting (A.4) and (A.8) into (A.3), we get:

$$\Pr(\omega^{1:k} \mid \boldsymbol{z}^{1:k}) \propto \left(\prod_{j=1}^{m_k} f(z^{k,j} \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1})\right) \left(\lambda_\nu^{N_\nu(\omega^k)}\right) \left(\lambda_\phi^{N_\phi(\omega^k)}\right) \tag{A.9}$$

$$(p_D)^{N_D(\omega^k)} (1-p_D)^{N_{\neg D}(\omega^k)} \Pr(\omega^{1:k-1} \mid \boldsymbol{z}^{1:k-1}) \tag{A.10}$$

To convert from this representation to that of Equation 2.7, take the ratio of this posterior

to the probability of the "all clutter" hypothesis:

$$\begin{aligned}
&\frac{\Pr(\omega^{1:k} \mid \boldsymbol{z}^{1:k})}{\Pr(\mathbf{0}^{1:k} \mid \boldsymbol{z}^{1:k})} \\
&= \frac{\left(\prod_{j=1}^{m_k} f(z^{k,j} \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1})\right)\left(\lambda_\nu^{N_\nu(\omega^k)}\right)\left(\lambda_\phi^{N_\phi(\omega^k)}\right)(p_D)^{N_D(\omega^k)}(1-p_D)^{N_{\neg D}(\omega^k)}}{\left(\prod_{j=1}^{m_k} 1/V\right)\left(\lambda_\phi^{m_k}\right)} \qquad \text{(A.11)} \\
&\qquad\qquad \times \frac{\Pr(\omega^{1:k-1} \mid \boldsymbol{z}^{1:k-1})}{\Pr(\mathbf{0}^{1:k-1} \mid \boldsymbol{z}^{1:k-1})} \\
&= \prod_{i=1}^{N}\left(\frac{\lambda_\nu}{\lambda_\phi}\right)\prod_{j=2}^{\ell_i}\left(\frac{p_D \Pr(z_{i,j} \mid \omega^{1:k}, \boldsymbol{z}^{1:k-1})}{\lambda_\phi 1/V}\right)^{\mathbb{1}_{[z_{i,j}\neq 0]}}\left(\frac{1-p_D}{1}\right)^{\mathbb{1}_{[z_{i,j}=0]}}, \qquad \text{(A.12)}
\end{aligned}$$

where the outer product is over the tracks selected by $\omega^{1:k}$. To follow the above derivation, note that all observations marked by $\omega^{1:k}$ as false alarms contribute the same terms to both the numerator and denominator (canceling each other), and the terms associated with the remaining observations have been grouped by their associated tracks. The form of Equation 2.7 in terms of track indicator variables follows directly.