Distributed Belief Propagation and its Generalizations for Location-aware Networks

by

Ulric John Ferner

B.Eng., Electrical and Electronic Engineering
University of Auckland (2007)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author .....
Department of Aeronautics and Astronautics
January 29, 2010

Certified by .....
Prof. Moe Win
Associate Professor
Thesis Supervisor

Accepted by .....
Prof. Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Distributed Belief Propagation and its Generalizations for Location-aware Networks

by

Ulric John Ferner

Submitted to the Department of Aeronautics and Astronautics
on January 29, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis investigates the use of generalized belief propagation (GBP) and belief propagation (BP) algorithms for distributed inference. The concept of a network region graph is introduced, along with several approximation structures that can be distributed across a network. In this formulation, clustered region graphs are introduced to create a network "backbone" across which the computation for inference is distributed. This thesis shows that clustered region graphs have good structural properties for GBP algorithms. We propose the use of network region graphs and GBP for location-aware networks. In particular, a method for representing GBP messages non-parametrically is developed. As an special case, we apply BP algorithms to mobile networks without infrastructure, and we propose heuristics to optimize degree of network cooperation. Numerical results show a five times performance increase in terms of outage probability, when compared to conventional algorithms.

Thesis Supervisor: Prof. Moe Win
Title: Associate Professor

Acknowledgments

I would like to gratefully acknowledge the many people who contributed to this thesis.

First and foremost, I wish to thank my advisor, Professor Moe Win, for his encouragement and guidance over the past two years. From him, I have gained invaluable insight and self-confidence, not just with respect to my work, but also my future. I would also like to thank him for giving me great freedom in exploring the spaces of inference and distributed systems. I cannot express the full extent of my gratitude for his support and patience.

I am greatly indebted to Henk Wymeersch for his cheerful and consistent guidance throughout my time with the group. Without his generous help, technical expertise, and unwavering support this thesis certainly would not have been possible. I also wish to thank Henk for the numerous lunches and for educating me on how good "good restaurants" can be.

I am grateful to all the members of the research group: Wes, Yuan, Pedro, William, and Ae, for their friendship and support. I have truly enjoyed getting to know each of you, both in and outside of work. Thank you for sharing your knowledge, for making the this a fun place to be, and for accompanying me on searches for free food.

To all my friends, old and new, thank you for sharing both the ups and the downs of this process with both empathy and understanding.

I wish to thank my parents, John and Sibylle, for their un-wavering support in everything I do. Your encouragement of me and my pursuits helps me more than you will ever know: you are my rock.

Finally I wish to thank Fulbright New Zealand for nudging me to leap to a place such as MIT, and for allowing me to travel to the other side of the world, literally, in pursuit of a dream.

Contents

List of Figures

Chapter 1

Introduction

The increasing demands of modern information systems call for the use of ever-more accurate inference algorithms. At the same time, modern networks are exploding in size which necessitates distributed algorithms.

Self-inference is a developing field in which a network must infer states about all or a subset of nodes within itself. Self-inference algorithms have one of two subtly different objectives; to estimate the joint probability distribution function (PDF) of the entire network or to estimate the joint PDFs of only subsets of nodes in the network. This thesis will focus exclusively on the latter case. The simplest case in this scenario is that the target subsets are simply each individual network node, in which case only the marginals of each node need to be estimated.

Large networks often have a large state space with a joint PDF that has non-trivial structure; this makes it prohibitively complex to optimally estimate target distributions [3]. To reduce algorithmic complexity, we search for PDFs that approximate the true PDF, but with simpler structure for computational tractability. A variety of graphical models have successfully been used to understand the structures that govern network-inference problems and to generate inference approximations. This includes directed graphical models such as Bayesian networks and region graphs, and undirected models such as Markov random fields and factor graphs [4]. This manuscript will focus on region graphs, which are constructed

from factor graphs.

Message passing algorithms are a class of algorithms that run across various graphical models and BP is a popular member of this class that runs across factor graphs. The "messages" in message passing algorithms are always sent from each node in the graphical model to a subset of its neighbors. The importance of this class of algorithms is evidenced by both their successful application in a variety of fields, as well the retrospective casting of classic algorithms as specific instances of message passing. For example, the Viterbi algorithm [5,6], the Kalman filter [7], and the sum-product algorithm for low-density parity check codes [8] can be cast as message-passing algorithms closely related to BP.

It has been shown that a factor graph with a large number of short cycles often causes BP algorithms to behave unpredictably; both in terms of convergence and inference accuracy. Significant work has been done to analyze the difference between the true marginals and those estimated by BP on arbitrary graphs with cycles, and then to develop bounds on the approximation error. For instance, it has been shown that the errors between the true and estimated marginals increase as the the divergence between the true PDF and its approximate tree representation [9]. This fact has also been observed by the empirical finding that BP performs well on graphs that are well approximated by trees.[1] Further, it is known that when measured under a tree distribution, the functions between variables that interact weakly within the graph have little effect on the accuracy of BP [9].

A general method of reducing the negative effects of short cycles in a factor graph $\mathcal{F}$ is to cluster or stretch random vertices that form short cycles into single factors in $\mathcal{F}$ [10]. However, the computational complexity of any algorithm that runs across $\mathcal{F}$ will increase exponentially with the number of variables in a cluster, or equivalently the size of a cluster. As an extreme case, if all variables in $\mathcal{F}$ are included in one cluster, then the inference problem can only be solved using a centralized and "brute-force" method.

One recently successful class of inference algorithms is generalized belief propagation (GBP) and the corresponding "region graph" approximation, as presented in [11]. GBP

[1] An example of a graph that is well approximated by a tree is one composed of a tree with the addition of a small number of long cycles.

algorithms run across region graphs, which are a class of graphical models constructed from factor graphs. The notion of grouping factors and variables of a factor graph into sets, as described above, is extended in GBP algorithms by allowing these sets to contain common elements. Non-disjoint groups allows GBP to exploit structure between random variables within a single group, which cannot be done when grouping variables in BP. As the name implies, BP is a special case of GBP, and GBP is a method of potentially overcoming the limitations of BP in factor graphs having many short cycles. The region graph method is extremely flexible but not every region-based approximation is accurate. It is still an open area of research as to how variables in a factor graph can be optimally grouped using region graph approximations, and this is a key problem in the development of GBP algorithms [11–15].

In their ordinary form, centralized GBP algorithms cannot be used for self-inference. In self-inference problems a graphical model represents some physical network graph and thus distributed, as opposed to centralized, inference algorithms are usually required. Cooperative networks are an emerging paradigm that enable distributed inference. Nodes in cooperative systems, each with unique objectives, cooperate or pool together their resources for mutual benefit.[2] A variety of cooperative techniques have been proposed for distributed inference, including Monte Carlo sequential estimation [16] and distributed belief propagation [17].

In self-inference systems the factors and variables in a factor graph can be divided into disjoint sets where each set represents a unique physical network node. If a message in the factor graph travels between two elements that are both associated with the same node, then that message is communicated internally within that node. On the other hand, if a message travels between two elements that are associated with different physical nodes, then that message is physically communicated between two different nodes. This method of distributed processing makes the computation of each node's marginal particularly elegant by multiplying the messages along each edge. Example applications include distributed localization [1, 18, 19], time-aware networks [20], relaying [21], and network discovery problems [22].

[2]This is in contrast to "collaborative" networks in which nodes pool their individual resources to achieve a common objective in a distributed manner.

When performing self-inference in location-aware networks, it is usually not feasible to do so in a centralized manner. This can be due to a lack of network infrastructure or limited computational resources. Further, location-aware networks in ad-hoc environments using distributed algorithms can operate more effectively due to their inherent robustness to failure.

There has been recent interest in BP-based algorithms for location-aware networks due to orders of magnitude increases in performance when compared to traditional methods [19]. In particular, a distributed BP algorithm for large-scale mobile networks called SPAWN (sum-product algorithm over a wireless network) was introduced in [1]. The SPAWN algorithm can deal with general network structures and it simplifies the computational complexity by not requiring computing ratios of messages. Although the SPAWN algorithm and the algorithms presented in this thesis are agnostic to the underlying communication technology, we evaluate the performance of these algorithms for location-aware networks employing ultra-wideband (UWB) transmission [23]. UWB is an attractive choice for simultaneous ranging and communication due to its ability to resolve multipath and penetrate obstacles [24, 25].

Despite the recent performance gains of BP algorithms for localization, no algorithm has yet come close to the performance bounds for such systems. Specifically, Fig. 1-1 compares the performance of the SPAWN algorithm [1] in UWB-networks to its performance bounds [2] in terms of outage probability, a common measure of localization performance. The key observation to be made here is that the performance of BP is still an order of magnitude away from the bound, and thus there exists potential opportunity for improved algorithms. This performance gap is due to large number of short cycles that the factor graphs representing location-aware networks contain. A class of distributed self-inference algorithms based on GBP could overcome this limitation. Furthermore, a GBP algorithm would provide direct access to the joint beliefs of groups or teams of network nodes for use in applications such as coordinated decision-making and path planning. Such information cannot be obtained using classic BP algorithms.

The goal of this thesis is to develop a distributed GBP algorithm for self-inference prob-

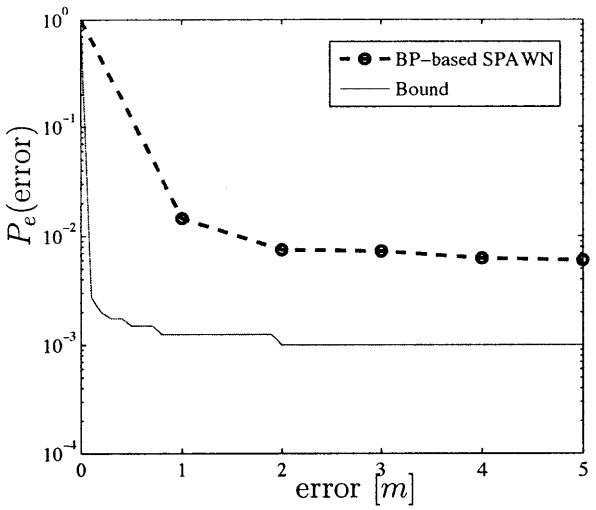


Figure 1-1: Outage probability comparison between the BP-based localization algorithm presented in [1] and the bound for TOA-based localization, as per [2]. Outage probability is defined as $P(e) = E\{\mathcal{I}[\|i - \hat{i}\| > e]\}$, where $\mathcal{I}$[P] is the indicator function. The variable $\hat{i}$ is the estimated location of node i, and e is the error threshold for the system in meters. Numerical results were obtained for networks of 100 nodes over 50 random topologies in a homogeneous environment. The BP-based algorithm is still an order of magnitude from the limit.

lems. This includes both the construction of distributed RGs as well as the execution of distributed GBP with arbitrary message structure. Finally, in the context of location-aware networks, existing BP algorithms are extended to account for mobile agents.

The main contributions of this thesis are the following:

- We introduce the notion of a network region graph (NRG), which allows inference algorithms such as GBP to be distributed across networks.
- We demonstrate that subgraphs forming CDGs can be used as "network-backbones," across which favorable NRG properties can be guaranteed. In particular, it is shown that CDG subgraphs that form trees or single cycles guarantee maxent-normality.
- We develop a method for representing GBP messages using particles, allowing for the representation of beliefs with arbitrary structure.
- Finally, we explore the use of BP-based algorithms, a special case of GBP, for mobile

location-aware networks and demonstrate a five times improvement in outage probability when compared with conventional techniques.

The remainder of this thesis is organized as follows. In Chapter 2 we present a set of mathematical and conceptual preliminaries. Chapter 3 introduces the notions of network- and clustered-region graphs. Chapter 4 develops an algorithm for representing GBP messages using particles in the context of location-aware networks. Chapter 5 presents BP algorithms for location-aware networks and selected numerical results. Finally, conclusions and directions for future work are presented in Chapter 6.

Chapter 2

Preliminaries

This chapter will introduce selected topics in inference including region graphs, maxent-normality, node clustering and particle-based functions. It will also introduce the reader to techniques that generate factor graphs from network graphs and techniques that generate region graphs from those factor graphs.

Let $\mathbf{X} = \{X_1, \ldots, X_N\}$ be a set of discrete-valued random variables (RVs) that represent the state of a system of interest, and let x_i and $\mathbf{x}$ be realizations or *states* of X_i and $\mathbf{X}$, respectively. For brevity and when there can be no confusion, we denote x_i simply as i. The PDF of interest $p_{\mathbf{X}}(x_1, \ldots, x_N)$ is denoted as $p(\mathbf{x})$. We assume that the PDF can be written in the general form

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{a \in \mathcal{A}} f_a(\mathbf{x}_a) \,, \tag{2.1}$$

where Z is the normalizing partition function, a is an index from $\mathcal{A} = \{1, 2, \ldots, A\}$ for A functions, and where each $f_a(\mathbf{x}_a)$ is a positive and well-defined function of some subset of $\mathbf{x}$ [26]. Generally, we are interested in the PDF of subsets of the variables $\mathbf{X}$: if X_s is some subset of $\mathbf{X}$, then denote the joint PDF of those variables as $p(\mathbf{x}_s)$ which can be computed by

$$p(\mathbf{x}_s) = \sum_{\mathbf{x} \backslash \mathbf{x}_s} p(\mathbf{x}) \tag{2.2}$$

where $\mathbf{x} \backslash \mathbf{x}_s$ denotes marginalization over all variables not in S.

A factor graph is a graphical representation of a global function of several variables in terms of its factors and their variables [10, 27]. Specifically, a factor graph is a bi partite graph containing both variable and factor vertices, where an edge connects a variable to a factor vertex if that factor is a function of that variable.[1] Although factor graphs provide a general tool applicable to a wide range of problems, in this manuscript we deal exclusively with functions which are (scaled) probability distributions.

2.1 Network Inference

In network inference, we are given a network graph $\mathcal{G} = (V(\mathcal{G}), E(\mathcal{G}))$, where V is a set of physical nodes and $E \subset V \times V$ is a set of edges connecting V. If two nodes can physically communicate over the network then an edge exists between them. In self-inference the goal of $\mathcal{G}$ is to infer some state of itself based on some incomplete and noisy information. As defined previously, let $p(\mathbf{x})$ be the PDF for the state of $\mathcal{G}$, where $p(i)$ is the state of node $i \in V(\mathcal{G})$. We define b as a PDF that approximates p and refer to b as the *belief* of p. For instance, $b(i)$ is the belief of node i.

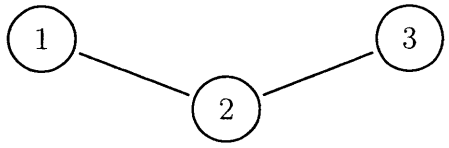
To capture the probabilistic dependencies in $\mathcal{G}$ we generate a factor graph from $\mathcal{G}$. Each pair of connected nodes (i, j) in $\mathcal{G}$ can extract information from one another and we denote the PDF of the information extracted by node i from node j as $\phi_{ij}(i, j)$.[2] As an example, consider the network and corresponding factor graph given in Fig. 2-1. In this case, we write the joint belief of this three-node network as

$$b(1,2,3) = b(1)\, b(2)\, b(3)\, \phi_{12}(1,2)\, \phi_{21}(2,1)\, \phi_{23}(2,3)\, \phi_{32}(3,2)\,. \tag{2.3}$$

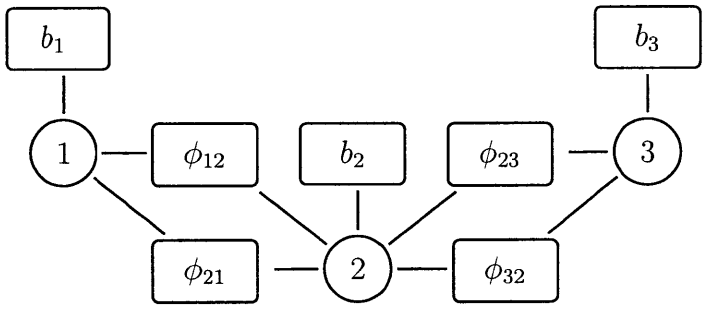
When given a network graph, we can always "expand" each edge in $E(\mathcal{G})$ using the same method as in Fig. 2-1(b). Specifically, each variable vertex i is connected to a single belief

[1] For instance, a function with factorization $f(1,2,3) = f_{\mathrm{A}}(1,2)\, f_{\mathrm{B}}(2,3)\, f_{\mathrm{C}}(3)$ gives rise to a factor graph with 3 factor vertices (f_{A}, f_{B} and f_{C}) and 3 variable vertices (1, 2 and 3), where variable 1 is connected to vertex f_{A}, variable 2 is connected to vertices f_{A} and f_{B}, and variable 3 connected to vertices f_{B} and f_{C}.

[2] It is not necessary that ϕ_{ij} is the same as ϕ_{ji}, i.e., it is possible that $\phi_{ij}(i,j) \neq \phi_{ji}(j,i)$



(a) A trivial network graph with only three nodes.



(b) A factor graph that corresponds to the network graph in Fig. 2-1(a). Factors are denoted by rectangular vertices and variables by circular vertices.

Figure 2-1: A three-node network graph and the corresponding factor graph.

vertex $b(i)$ and each edge $(i,j) \in E(\mathcal{G})$ has two associated functions $\phi_{ij}(i,j)$ and $\phi_{ji}(j,i)$. Note that if $\phi_{ij}(i,j) \equiv 0$ then no information is extracted from node i by node j so $\phi_{ij}(i,j)$ can be removed from the factor graph. We refer to the above process as a *factor graph expansion* of network graph $\mathcal{G}$.

2.2 Region Graphs and Region-based Approximations

This section defines region graphs and free energy functions. We then explore the design of region-based approximations using free-energy functions.

As discussed in Chapter 1, one method of reducing the effect of short cycles in a factor graph is to cluster vertices into disjoint sets. This can be generalized by allowing vertices to be members of more than one cluster or, equivalently, we can define hyper-edges over the factor graph's vertices. The common nodes between these clusters are made apparent using region graphs [11].

Definition 1. A *zone* r of a factor graph is a set of factor vertices A_r and a set of variable nodes V_r such that if factor vertex $f \in A_r$, then all neighboring variable nodes of f are in V_r.

Let $\mathbf{x}_r$ be the collective set of variable nodes in zone r. The PDF of all variables in r is given by $p(\mathbf{x}_r)$ and the belief approximating $p(\mathbf{x}_r)$ is denoted by $b(\mathbf{x}_r)$.

Definition 2. A *region graph* is defined as a labeled directed graph $\mathcal{R} = (V(\mathcal{R}), E(\mathcal{R}), L(\mathcal{R}))$ that is composed of a set of vertices V, a set of directed edges $E \subset V \times V$, and a set of labels L, where each vertex is assigned to a unique label. Each label contains a zone from a factor graph and a counting number, and each vertex is referred to as a region.

Throughout this manuscript we abbreviate $r \in V(\mathcal{R})$ as $r \in \mathcal{R}$. In a region graph, if a directed edge points from vertex u to vertex v, we say that u is a *parent* of v, and that v is a *child* of u. Further, if there exists a directed path from u to v we say that u is an *ancestor* of v and that v is a *descendant* of u. The set of regions in $\mathcal{R}$ with no parents are referred to as *outer* regions $\mathcal{R}_0$ and all others are referred to as *inner* regions [28]. Further, the set $r \in \mathcal{R}_k$ denotes all regions for which k is the length of the largest path from each r to any element in $\mathcal{R}_0$.

Each region's label contains a region and a "counting number." The counting number for region r is denoted by c_r and is computed using

$$c_r = 1 - \sum_{j \in \mathcal{A}(r)} c_j \,, \tag{2.4}$$

where $\mathcal{A}(r)$ is the set of ancestors for region r. If a region has no ancestors, then its counting number is equal to one. Eq. (2.4) ensures that each factor f and variable i in the factor graph is "counted" only once in $\mathcal{R}$, i.e.,

$$\sum_{r \in \mathcal{R}} c_r \mathcal{I}(f \in A_r) = \sum_{r \in \mathcal{R}} c_r \mathcal{I}(i \in V_r) = 1 \tag{2.5}$$

where $\mathcal{I}$ is the indicator function.

There are numerous RGs that can be generated from any given factor graph. Each RG will dictate or constrain the set of beliefs $\{b_r\}$ for which we perform inference. The design of RG generation algorithms therefore requires an understanding of the expected errors due to constraining our beliefs through $\mathcal{R}$. This process of constraining our beliefs is referred to as a *region-based approximation* to p.

Error analysis for region-based approximations has historically been pursued in the field of statistical physics using free-energy functions: these functions provide key insight into expected inference errors, thereby guiding RG construction. Equations (2.6)–(2.14) in the following, together with the *maxent-normal* property, help steer the RG generation process, and the maxent-normal property will play a pivotal role in subsequent chapters.

The energy E of a random variable $\mathbf{x}$ is given by

$$E(\mathbf{x}) = -\sum_{a=1}^{A} \ln f_a(\mathbf{x}_a)\,. \tag{2.6}$$

Given some belief b, the variational free energy of a system $F(b)$ is defined as

$$F(b) = U(b) - H(b) \tag{2.7}$$

where $U(b)$ is the variational average free energy given by

$$U(b) = \sum_{\mathbf{x}} b(\mathbf{x}) E(\mathbf{x})\,, \tag{2.8}$$

and $H(b)$ is the variational entropy given by

$$H(b) = -\sum_{\mathbf{x}} b(\mathbf{x}) \ln b(\mathbf{x})\,. \tag{2.9}$$

Using the above definitions it directly follows that

$$F(b) = -\ln Z + D(b||p) \tag{2.10}$$

where $D(b||p)$ is the KL-divergence given by[3]

$$D(b||p) \equiv \sum_{\mathbf{x}} b(\mathbf{x}) \ln \frac{b(\mathbf{x})}{p(\mathbf{x})} . \tag{2.11}$$

This implies that the minimization of the variational free energy $F(b)$, with respect to $b(\mathbf{x})$, will also minimize the divergence between p and b. If b is unconstrained, (2.10) is minimized when $b = p$ and the partition function Z will be recovered. On the other hand, when beliefs are constrained through a RG $\mathcal{R}$, the region-based free energy $F_{\mathcal{R}}(\cdot)$ is defined as

$$F_{\mathcal{R}}(\{b_r\}) = U_{\mathcal{R}}(\{b_r\}) - H_{\mathcal{R}}(\{b_r\}) \tag{2.12}$$

where $U_{\mathcal{R}}(\{b_r\})$ and $H_{\mathcal{R}}(\{b_r\})$ are the region-based average energy and the region-based entropy, given respectively by

$$U_{\mathcal{R}}(\{b_r\}) = \sum_{r \in \mathcal{R}} c_r U_r(b_r) \tag{2.13}$$

and

$$H_{\mathcal{R}}(\{b_r\}) = \sum_{r \in \mathcal{R}} c_r H_r(b_r) . \tag{2.14}$$

To find the set of beliefs $\{b_r\}$ that is closest to p, we wish to minimize $F_{\mathcal{R}}$. Further, it has been shown that if the beliefs $\{b_r(\mathbf{x}_r)\}$ are equal to the corresponding exact marginal probabilities $\{p_r(\mathbf{x}_r)\}$, then $U_{\mathcal{R}}(\{b_r\}) = U(b)$, i.e., the approximation made by region-based average free energy is exact irrespective of the functional form of b.

On the other hand, the region-based entropy does not have a similar property: it is certainly not guaranteed that the region-based entropy is equal to the variational entropy. The magnitude of the error encapsulated by the region-based entropy is highly dependent on the structure of the corresponding RG $\mathcal{R}$, and its analysis is considered key in the development of region-based algorithms [11].

The amount of error due to the region-based entropy depends not only on the structure

[3]KL-divergence $D(b||p)$ is based on a local and proper cost function that measures the closeness of b to p.

of $\mathcal{R}$ but also on the functional form of b. Recall from (2.12) that we are interested in the set $\{b_r\}$ that results in large values of $H_{\mathcal{R}}$. Note also that If the region-based entropy is not maximized when all the beliefs take on a particularly simple functional form, then it is highly unlikely that $\mathcal{R}$ will generate good approximations when b has more elaborate structure [29]. One of the simplest belief structures is if all beliefs are independent and identically distributed. Since all variables are discrete, and if all variables have the same domain, then the above motivate the following desirable property of RGs [11, 28].

Definition 3. *(Maxent-normality)* Suppose a region graph $\mathcal{R}$ is over set of discrete RVs. The graph $\mathcal{R}$ is considered to be *maxent-normal* if its region-based entropy $H_{\mathcal{R}}$ is maximized when all beliefs are uniform.

Note that the maxent-normal (MN) RGs do not guarantee to provide the best region-based approximation over all $\mathcal{R}$. For any factor graph there can be numerous MN RGs. However, if a RG is MN then the region-based free-energy does reliably give more accurate estimates of marginal probabilities than the corresponding "Bethe approximation," used by BP algorithms. The interested reader is referred to [9] for details.

2.3 Clustering

Clustering algorithms are commonly used in the communications and networking community for applications such as routing and efficient bandwidth sharing [30]. Suppose we are given a network graph $\mathcal{G}$, distributed clustering algorithms aim to divide all vertices $V(\mathcal{G})$ into subsets or clusters (which are not necessarily disjoint), such that certain constraints are met or a particular cost function is optimized.

A common problem is to identify a subset of $V(\mathcal{G})$, called a dominating set (DS), such that each network node is either in the DS or is directly connected to at least one member of the DS. A connected dominating set (CDS) of $\mathcal{G}$ is a DS $V(\mathcal{D})$ such that $\mathcal{D}$ induces a connected subgraph. The minimum connected dominating set (MCDS) of is the CDS that contains the minimum number of nodes $i \in V(\mathcal{G})$. Network nodes $i \in V(\mathcal{D})$ are referred

to as *dominating* nodes and all other network nodes $j \notin V(\mathcal{D})$ are referred to as *unmarked* nodes. For each node $i \in V(\mathcal{G})$, a "marking function" is defined as

$$m(i) = \begin{cases} T & i \in V(\mathcal{D}), \\ F & \text{otherwise}. \end{cases} \tag{2.15}$$

For arbitrary graphs, finding the MCDS is NP-complete [31], but there exist distributed algorithms for finding approximations to the MCDS [30,32]. Similar ideas for approximating the MCDS will be used in this manuscript.

2.4 Particle-based Functions

Functions that cannot be represented algebraically in closed-form can often be represented non-parametrically using particles. Suppose we wish to represent a belief $b(i)$ as N_s weighted particles, $\left\{i^{(k)}, w^{(k)}\right\}$, where $i^{(k)}$ is a sample drawn from some distribution with the same support as $b(i)$, and $w^{(k)}$ is the appropriate weight of $i^{(k)}$. Throughout the manuscript we use the terms particles and samples interchangeably.

If

$$\sum_{k=1}^{N_s} w^{(k)} = 1 \tag{2.16}$$

and

$$\sum_{k=1}^{N_s} w^{(k)} g(i^{(k)}) \approx \int g(i) b(i) di \tag{2.17}$$

for any integrable function g, then we can represent b non-parametrically.

There are different ways to obtain a set of weighted samples. Popular choices are (i) direct sampling, where we draw N_s independent samples from $b(i)$, each with weight $1/N_s$; (ii) importance sampling (IS), where we draw N_s independent, identically distributed (i.i.d.) samples from a *sampler* distribution $q(i)$, with a support that includes the support of $b(i)$, and

set the weight corresponding to sample $i^{(k)}$ as

$$w^{(k)} = b(i^{(k)})/q(i^{(k)})\,. \tag{2.18}$$

In both cases, it can easily be verified that the variance of the approximation reduces with N_s (and for IS that it depends on q). However, it is important to note that when using particles the variance does not depend on the dimensionality of i. A list of N_s equally-weighted samples can be obtained from $\left\{i^{(k)}, w^{(k)}\right\}$ through resampling, by drawing from $\{i^{(k)}\}$ with associated probabilities $\{w^{(k)}\}$ [33].

When approximating $b(i)$ using particles, it is often required to interpolate between each sample. Given a sample representation $\left\{i^{(k)}, w^{(k)}\right\}$ we approximate $b(i)$ as

$$\hat{b}(i) = \sum_{k=1}^{N_s} w^{(k)} K_\sigma(i - i^{(k)}), \tag{2.19}$$

where K_σ is the so-called *kernel* with bandwidth σ, and (2.19) is the so-called kernel density estimate (KDE). The kernel is a symmetric distribution with an adjustable width parameter σ. For instance, a Gaussian kernel is given by

$$K_\sigma(\mathbf{x}) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right). \tag{2.20}$$

While the choice of kernel affects the performance of the estimate to some extent, the crucial parameter is the width σ, which needs to be estimated from the samples $\left\{i^{(k)}, w^{(k)}\right\}$. A large choice of σ makes $\hat{b}(i)$ very smooth, but may no longer capture the interesting features of $b(i)$, while a small choice of σ may result in $\hat{b}(i)$ exhibiting artificial structure not present in $b(i)$ [34].

Chapter 3

Network Region Graphs

The goal of this chapter is to develop a graphical model which allows inference algorithms such as GBP to be distributed across networks. We begin by discussing general distributed inference problems with RGs, and will later restrict our treatment to distributed self-inference.

In regular distributed inference problems with RGs, we are given some pre-existing region graph (RG) $\mathcal{R}$ that may or may not contain probabilistic information contained *within* $\mathcal{G}$. Our goal is to develop a structure called a NRG that enables the computation of GBP algorithms to be distributed *across* $\mathcal{G}$.

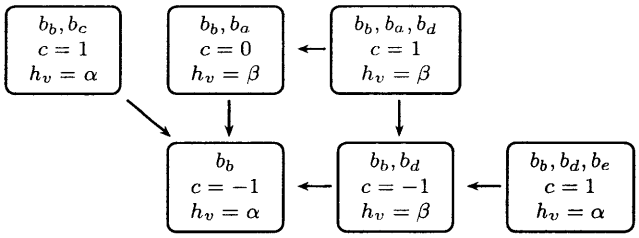
Definition 4. *(Network region graph)* Let $\mathcal{A}$ be a set of indices for variable and factor vertices in a factor graph and let $\mathcal{G}$ be a network graph. A *network region graph* is defined as a labeled and directed graph $\mathcal{R} = (V, E, (L, h_v))$ that is composed of a set of vertices V, a set of directed edges $E \subset V \times V$, and a set of labels L, where each vertex is assigned to a unique label $l \in L$. The label l of vertex r includes a controlling node through the label vector $h_v = i \in V(\mathcal{G})$, a zone of $\mathcal{A}$, and a counting number c_r. As a slight abuse of notation, we refer to each vertex in $r \in \mathcal{R}$ as a region.

A NRG is then an existing RG in which the control of each region is mapped or assigned to a unique network node. Any network node may be assigned control of multiple regions, while other network nodes may have no regions assigned to them; thus and so the computation of GBP is distributed across a subgraph of $\mathcal{G}$.

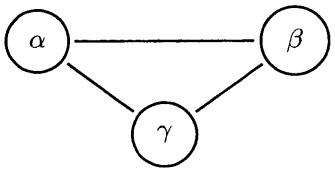
An example of a NRG and its associated network graph is shown in Fig. 3-1. Network graph $\mathcal{G}$ is composed of three nodes α, β, and γ. We will describe the notation of the NRG through example in the following. The pre-existing RG contains six regions overall: three outer regions with no parents and with counting number equal to one, and three inner regions. The top-left region is an outer region that contains factors b_b, b_c, which are a function of variables b and c, respectively. The computation and control of the top-left region is assigned to network node $h_v = \alpha$. In this example, the control of each region in $\mathcal{R}$ has been assigned to either α or β, and thus only a subgraph of $\mathcal{G}$ controls $\mathcal{R}$. Note that the variables and factors contained in $\mathcal{R}$ do not directly correspond to variables and factors in $\mathcal{G}$, but only to some pre-existing factor graph that is not shown.

We now shift our focus to the design of NRGs for self-inference problems, i.e., to NRGs in which the contained factors and variables correspond to the state of $\mathcal{G}$ itself. In self-inference problems, we are given a network graph $\mathcal{G}$ from which we first generate a factor graph, and we then generate $\mathcal{R}$ from this factor graph. To perform distributed self-inference, both the construction of $\mathcal{R}$ as well as the message passing across $\mathcal{R}$ need to be distributed. We generate RGs directly from network graphs through a systematic, but implied, factor graph expansion as described in Chapter 2.1. It should be understood that each RG generated from a network graph is done through factor graph expansion, even if the corresponding factor graph is not shown explicitly. This will simplify our discussions of RGs and of the distributed processing for GBP.

We divide the design of a RG generation algorithm into two sub-problems in the following. Sec. 3.1 develops the desired structure of the RG generated from the given network graph. Sec. 3.2 then develops a distributed algorithm to generate a NRG—with the structure identified in Sec. 3.1—in which the control and computation for regions in $\mathcal{R}$ is assigned to network nodes.



(a) An example NRG $\mathcal{R}$ in which each region in $r \in \mathcal{R}$ has been mapped to a controlling node in the network graph $\mathcal{G}$, shown in Fig. 3-1(b), through a controlling function $h_v(r)$. The factor graph from which $\mathcal{R}$ was generated is not shown.



(b) An example network graph $\mathcal{G}$ where physical nodes are represented by vertices and edges denote communication between nodes. The control of each region $r \in \mathcal{R}$ has been mapped to a unique node in $\mathcal{G}$.

Figure 3-1: A toy network graph made up of three nodes labelled α, β, and γ. A corresponding network region graph is also shown, where each region contains a set of factors and variables. Factors are denoted by b's and the associated variables for each factor are denoted by subscripts.

3.1 Clustered Region Graphs

We introduce the notion of a clustered RG which will be used to systematically generate a RG from any given $\mathcal{G}$ in a centralized manner. The reader is reminded that in our final algorithm this clustered RG $\mathcal{R}$ must be generated in distributed form. We are interested in structures that are guaranteed to be maxent-normal (MN). Unless otherwise stated, we will use the set notation as described in Table 3.1. (The terms in Table 3.1 not yet fully defined will be clarified later in this chapter.)

A key consideration when designing a RG generation algorithm, be it distributed or not, is that different regions in the same graph are usually generated using different sub-

Table 3.1: Description of set notation used throughout the manuscript.

Set	Description
$\mathcal{G}$	a network-graph
$E(\mathcal{B})$	the set of edges in graph $\mathcal{B}$
$V(\mathcal{B})$	the set of vertices in graph $\mathcal{B}$
$\mathcal{N}(i)$	the set of nodes that neighbor node $i \in V(\mathcal{G})$
$\mathcal{N}[i]$	$\mathcal{N}(i) \cup i$
$\mathcal{D}$	a CDG and subgraph of $\mathcal{G}$
$\mathcal{D}_{\mathrm{t}}$	a spanning tree across $\mathcal{D}$
$\mathcal{R}$	a region graph
$\mathcal{R}_0$	the set of outer-regions of $\mathcal{R}$
$\mathcal{R}(i)$	the set of regions in $\mathcal{R}$ controlled by node i

algorithms. In particular, once a set of outer regions is defined, then standard techniques exist for generating the respective inner regions. However, standard methodologies to generate outer regions $\mathcal{R}_0$ is still an open research problem. We will develop a technique to generate $\mathcal{R}_0$. The inner regions will be constructed by the commonly used "Kikuchi method," as described in [35–37].

The number of factors in each outer region should be kept as small as possible to minimize the complexity of our GBP algorithm. On the other hand, an increase in the number of factors in each region will likely improve the accuracy of our marginal estimates. As described in [11], this well-known trade-off motivates keeping the number of factors in each region small, whilst capturing a large number of short cycles in each region.

We propose to use of a subgraph $\mathcal{D}$, which forms a CDG, to act as a "network inference backbone:" the subgraph $\mathcal{D} \subset \mathcal{G}$ will be used as a basis from which we generate our RG. Throughout the paper, we insist on our subgraph $\mathcal{D}$ to have the *inter-cluster property*, which will serve to guarantee the maxent-normality of our generated RGs.

Definition 5. *(Inter-cluster property)* Suppose a subgraph $\mathcal{D}$ of a network graph $\mathcal{G}$ is a CDG. An unmarked node i ($i \notin V(\mathcal{D})$) is called an *inter-cluster node* if each neighboring dominating node j ($j \in \mathcal{N}(i) \cap V(\mathcal{D})$) is connected to all other neighbors of i. If every node i is an inter-cluster node, we say that the graph pair $(\mathcal{G}, \mathcal{D})$ has the *inter-cluster property.*

We now generate a clustered RG using a CDG with the inter-cluster property.

Definition 6. *(Clustered region graph)* Let $(\mathcal{G}, \mathcal{D})$ have the inter-cluster property. A RG is called a *j-clustered region graph* (C^j-RG) if the kth outer region contains exactly all factors and variables associated with edges in $\mathcal{G}$ that connect nodes within j hops of dominating node k, and inner regions are defined using the Kikuchi method.

Remarks:

- Any graph pair $(\mathcal{G}, \mathcal{D})$ maps to a unique C^j-RG.
- In the following, we will consider C^j-RGs in which $j = 1$. This choice minimizes the computational complexity of GBP, whilst still capturing a large number of short cycles in the underlying factor graphs (FGs).

We will now analyze the properties of C^1-RGs. As previously discussed, MN is a desirable property of a RG for improving estimates of marginals. We will systematically identify the structures of C^1-RGs that are MN, and this process can be simplified by identifying nodes and edges in $(\mathcal{G}, \mathcal{D})$ that have no effect on the MN of $\mathcal{R}$. We begin by analyzing the effect of edges between unmarked nodes in $\mathcal{G}$ on the MN property, and then analyze the effect of adding and removing unmarked nodes in $\mathcal{G}$.

Lemma 1. *Suppose we have a C^1-RG $\mathcal{R}$ that is MN. Removing edges between unmarked nodes in $\mathcal{G}$ does not affect the MN property of $\mathcal{R}$. Further, adding edges that do not violate the inter-cluster property between unmarked nodes in $\mathcal{G}$ does not affect the MN property of $\mathcal{R}$.*

Proof. See Appendix A. □

Lemma 2. *Suppose we have a C^1-RG $\mathcal{R}$ that is MN. Attaching any number of new unmarked nodes to the CDG does not affect the MN property of $\mathcal{R}$.*

Proof. See Appendix A. □

Theorem 1. *A C^1-RG generated from network graph $(\mathcal{G}, \mathcal{D})$ is MN if and only if the C^1-RG generated by only $\mathcal{D}$ is MN.*

Proof. Suppose a C^1-RG generated from $(\mathcal{D}, \mathcal{D})$ is MN. Without loss of generality, we can reconstruct $\mathcal{G}$ from $\mathcal{D}$ by attaching any number of unmarked nodes to $\mathcal{D}$. By Lemma 2, a C^1-RG generated from $(\mathcal{G}, \mathcal{D})$ is MN.

Suppose a C^1-RG generated from $(\mathcal{G}, \mathcal{D})$ is MN. Without loss of generality, we can remove edges and unmarked nodes from $(\mathcal{G}, \mathcal{D})$ until only $\mathcal{D}$ remains. By Lemma's 1–2, a C^1-RG generated from $(\mathcal{D}, \mathcal{D})$ is MN. □

Remark: Theorem 1 allows one to test for the MN property of a C^1-RG by only looking at the structure of $\mathcal{D}$. We now systematically identify the structures of C^1-RGs that are MN.

Lemma 3. *If $\mathcal{G}$ is associated with a $\mathcal{D}$ that forms a chain of finite length, then any C^1-RG generated from $(\mathcal{G}, \mathcal{D})$ is MN.*

Proof. See Appendix A. □

Remark: It is apparent from the proof of Lemma 3 that we can always use the mutual information of the an outer region to rewrite the negative entropy term from each region in $\mathcal{R}_1$ since $|\mathcal{R}_0| > |\mathcal{R}_1|$, where each region in $\mathcal{R}_1$ has $c_r = -1$.

Lemma 4. *If $\mathcal{G}$ is associated with a $\mathcal{D}$ that is a cycle of finite length not equal to four, then any C^1-RG generated from $(\mathcal{G}, \mathcal{D})$ is MN.*

Proof. Let $\mathcal{D}_3 \subset \mathcal{G}$ form a single cycle of length three. The C^1-RG $\mathcal{R}$ generated from $(\mathcal{G}, \mathcal{D}_3)$ is trivially MN because $\mathcal{R}$ will contain three outer regions each with counting number equal to 1 and only a single inner region r_1 with counting $c_{r_1} = -2$. Considering the region-based entropy $H_{\mathcal{R}}$, we can cancel the negative term $-2H_{r_1}$ by rewriting the entropy of two outer regions as a mutual information term and a positive entropy whose variables does not include those in r_1. The region-based entropy will then be a sum of positive entropy and negative mutual information terms so the C^1-RG is MN. Now let $\mathcal{D}_4 \subset \mathcal{G}$ form a single cycle of length 4, as depicted in Fig. 3-2. The C^1-RG generated from $(\mathcal{G}, \mathcal{D}_4)$ will contain four regions in $\mathcal{R}_0$, but six regions in $\mathcal{R}_1$, implying that the graph may or may not be MN.

Now let $\mathcal{D}_i \subset \mathcal{G}$ form a single cycle of length i, where $i \geq 5$. In this case, the C^1-RG $\mathcal{R}$ has the same structure as that generated from a chain $\mathcal{D}$ of length i, except that $|\mathcal{R}_1| = |\mathcal{R}_0|$, due to the extra edge in $\mathcal{D}_i$ connecting the beginning and end of the chain. The counting numbers for $r \in \mathcal{R}_0$ will be $c_r = 1$ and the counting numbers for each $r \in \mathcal{R}_1$ will be $c_r = -1$. Since the magnitude of the sum of the counting numbers of the inner regions does not exceed the sum of the counting numbers of the outer regions, we can again rewrite each term using the definition of mutual information, as per the proof for Lemma 3. □

Remarks

- The MN of single cycle CDGs mirrors the accurate performance of BP in factor graphs that form a single cycle.
- It was observed in [11] that GBP performs poorly in Ising models, which are lattice factor graphs with multiple cycles of length four. Lemma 4 provides some insight as to the reasons why this is so from a graph theoretical perspective.

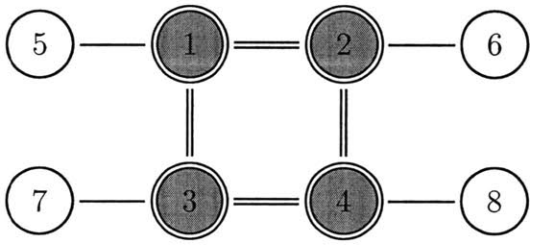


Figure 3-2: An example network graph that contains a cyclic CDG $\mathcal{D}$ of length four.

We now shift our analysis to subgraphs that form trees. Trees are a particularly attractive because there exist distributed algorithms to construct trees from arbitrary network graphs.

Theorem 2. *If $\mathcal{G}$ is paired with both a $\mathcal{D}$ and a corresponding subgraph $\mathcal{D}_t$ that forms a tree of finite size, then any C^1-RG generated from $(\mathcal{G}, \mathcal{D}_t)$ that does not include edges in $\mathcal{D}_t^c \cap \mathcal{D}$ is MN.*

Proof. We only need to consider the structure of $\mathcal{D}_t$, by Theorem 1. Each $i \in V(\mathcal{D})$ generates one unique $r \in \mathcal{R}_0$. The region r will have a common zone with other regions generated by its

neighbors dominating nodes. Without loss of generality, we only need to analyze nodes with connectivity strictly greater than 2 due since the C^1-RG generated by each chain segment in $\mathcal{D}$ is MN, by Lemma 3. Let the ith dominating node (DN) have $n_i > 2$ neighboring DNs, and let N_D be the number of DNs in $\mathcal{D}$.

Consider the structure and counting numbers for all regions in $\mathcal{R}_1$. Since $\mathcal{D}_\mathrm{t}$ is a tree, $|V(\mathcal{R}_1)| = N_D - 1$, each $r_1 \in \mathcal{R}_1$ has 2 parents in $\mathcal{R}_0$, and thus r_1 will have $c_{r_1} = -1$. Consider the structure and counting numbers for all regions in $\mathcal{R}_2$. Node i will generate a single $r_2 \in \mathcal{R}_2$, with a zone that only contains b_i, and r_2 will have n_i parents, each from $\mathcal{R}_1$. The tree structure of $\mathcal{D}_\mathrm{t}$ implies that $n_i + 1$ regions in $\mathcal{R}_0$ will be ascendants of r_2, thus

$$\begin{aligned} c_{r_2} &= 1 - \sum_{j \in \mathcal{A}(r_2)} c_j \\ &= 1 - \left\{ \sum_{j \in \mathcal{R}_1 \cap \mathcal{A}(r_2)} c_j + \sum_{j \in \mathcal{R}_0 \cap \mathcal{A}(r_2)} c_j \right\} \\ &= 1 - \{-n_i + n_i + 1\} = 0\,. \end{aligned} \tag{3.1}$$

Each region in $\mathcal{R}_2$ contains only a single belief, thus the C^1-RG is composed of only three "layers" $\mathcal{R}_0$, $\mathcal{R}_1$, and $\mathcal{R}_2$. All regions in $\mathcal{R}_2$ have counting number 0, and $|\sum_{r \in \mathcal{R}_0} c_r| > |\sum_{r \in \mathcal{R}_1} c_r|$, thus $\mathcal{R}$ is MN.

□

Remarks

- Numerous distributed algorithms exist that efficiently construct spanning trees, including depth-first and breadth-first search variants [31]. This allows algorithms such as Dijkstra's algorithm to be used as a pre-processor for creating a tree that can be used to generate a C^1-RG.

- A number of trees and rings cannot be connected arbitrarily whilst still guaranteeing MN because it is known that every connected graph decomposes canonically into 2-connected subgraphs (and bridges) which can be arranged as a tree. Every 2-connected

subgraph can be constructed from starting with a single cycle and then adding successive H-paths to that graph [38]. However, if $\mathcal{D}$ contains only a single cycle then a C^1-RG generated from $(\mathcal{G}, \mathcal{D})$ will be MN, by Theorem 2 and Lemma 4.

The notion of C^1-RGs allows us to systematically generate an MN RG from a given $\mathcal{G}$. In the next section we propose a distributed algorithm to generate these C^1-RG.

3.2 Network Region Graph Generation

The previous section identified a C^1-RG as the structural form of our target RG that will be generated from $\mathcal{G}$ in a distributed manner. In this section we will design a distributed algorithm that (1) generates our target C^1-RG from $\mathcal{G}$ and (2) assigns control of each region to form a NRG.

3.2.1 Problem Statement

Given a network graph $\mathcal{G}$, we consider that each physical node $i \in V(\mathcal{G})$ has (1) knowledge of the set of neighboring nodes $\mathcal{N}(i)$ from which i can receive information; (2) has access to bidirectional communications with each of its neighbors; and (3) has a prior distribution or marginal belief $b(i)$. Our goal is to construct a NRG $\mathcal{R}$ with the same structural form as C^1-RG using a distributed algorithm.

3.2.2 Algorithm Design

To generate $\mathcal{R}$ requires assigning a controlling network node to each region. The role of any $i \in V(\mathcal{G})$ that has "control" of $r \in \mathcal{R}$ is as follows. Node i will be assigned to physically receive all incoming GBP messages into r, and it will be assigned to compute all messages outgoing from r, while simultaneously computing the joint belief of all nodes contained in r.

We propose to use of a subgraph $\mathcal{D}_\mathrm{t}$, which forms a CDG, to act as a "network inference backbone" and to control all regions in $\mathcal{R}$. Each node $i \in V(\mathcal{D})$ will be used to generate one

outer region $r \in \mathcal{R}_0$, and node i is naturally assigned control of that region. All other inner regions will also be generated and controlled by nodes in $V(\mathcal{D})$.

Prior to discussing details of generating a RG using $\mathcal{D}$ as a basis, we describe some advantageous properties of using CDGs, from a network-design perspective:

- A dominating graph ensures all network nodes are within at most one communication hop of the backbone. This simplifies network communication overhead as the routing requirements are minimized.[1]
- A connected backbone means that marked nodes can communicate between themselves without requiring assistance from unmarked nodes; this can further simplify network communications.
- If each DN in a dominating graph forms a team with nodes around it, then each unmarked node can trivially determine which teams it belongs to.

Using network graph $\mathcal{G}$, we propose constructing a CNRG, which is simply a NRG with the same structural properties as a C^1-RG.

Definition 7. A *clustered network region graph* (CNRG) $\mathcal{R}$ is defined as a 1-clustered region graph in which the control of each region $r \in \mathcal{R}$ has been assigned to one dominating node $i \in V(\mathcal{D})$.

To demonstrate the value of CNRGs, we describe a set of CNRG properties that can simplify algorithm design. Each node $i \in V(\mathcal{D})$ that controls an $r \in \mathcal{R}$, whether r is an inner or outer region, naturally has access to at least the following information: (1) the IDs of all node variables $j \in A_r$; (2) all regions in the RG to which i belongs; (3) the set of neighboring regions to r, $\mathcal{N}(r)$ and (4) the directions of the edges that connect r to each of its neighbors.

The assignment of every region to a controlling node $i \in V(\mathcal{D})$ is relatively simple because every region in our target C^1-RG contains at least one DN variable.

[1] If the backbone is not a dominating graph then communication across the network is still possible, but it would require the use of unmarked nodes as relays to route all information into $\mathcal{D}$.

Proposition 1. Each region in a CNRG contains at least 1 DN variable i.

Proof. Suppose there exists a region $r \in \mathcal{R}$ that does not contain any DN variables. Since r is non-empty, r contains at least one unmarked network node variable j. Since all $r \in \mathcal{R}_0$ must contain at least one DN variable, then $r \notin \mathcal{R}_0$, implying that r is an inner region. Thus r must have at least two ancestor regions that do contain DNs m and n. Hence, in the network graph j must be connected to nodes m and n such that $(m, n) \notin E(\mathcal{G})$. This implies that j is not an inter-cluster node, contradicting the inter-cluster property of $(\mathcal{G}, \mathcal{D})$. □

Further, the controlling node $i \in V(\mathcal{D})$ for region $r \in \mathcal{R}$ can locally and efficiently compute the counting number c_r as follows: if all nodes connected to i exchange their neighbor sets with node i, then i knows all DNs connected to its neighbors. In particular, node i knows which of its neighbors are connected and hence it knows all inner regions that have common zones with r. The set of common zones includes the ancestors $\mathcal{A}(r)$ of r, which is required in (2.4).

It is now apparent that a CNRG is a natural construction for a maxent-normal (MN) NRG generated from $\mathcal{G}$. To design an algorithm to create this CNRG we take the following three steps. First, from $\mathcal{G}$ we generate $\mathcal{D}_\mathrm{t}$ in a distributed manner, and use $\mathcal{D}_\mathrm{t}$ to generate $\mathcal{R}_0$. Second, from $\mathcal{R}_0$ we generate all other regions in $\mathcal{R}$. Third, we allocate control of each region and allocate edges in the NRG to carry messages either internally within a physical node, or externally between two nodes.

The first step is to generate $\mathcal{D}_\mathrm{t}$ from $\mathcal{G}$. A variety of algorithms exist for generating the subgraph $\mathcal{D}_\mathrm{t}$; for an example see Algorithm 1. We assume that every network node i has a unique identifier $I(i)$. Lines 2–7 in Algorithm 1 are similar to the clustering algorithm in [32]. Line 3 of Algorithm 1 forms a CDG and thereafter, lines 5–10 decrease the size of this initially constructed CDG.

In the following, we show that Algorithm 1 generates a CDG with the inter-cluster property. Line 8 generates a set of DNs that are super-set of the DNs generated by the algorithm described in [32], thus Algorithm 1 creates a CDG. We now show that Algorithm 1 creates a $\mathcal{D}_\mathrm{t}$ with the inter-cluster property.

Algorithm 1 A distributed $\mathcal{D}_\mathrm{t}$ generation algorithm, in which $\mathcal{D}_\mathrm{t}$ has the inter-cluster property.

1: Every node v exchange $\mathcal{N}(v)$ with its neighbors
2: **if** $\exists i, j : i, j \in \mathcal{N}(v)$, $(i, j) \notin E(\mathcal{G})$ **then**
3: Mark v as a DN, $m(v) \leftarrow T$
4: **end if**
5: **if** $u, v \in V(\mathcal{D})$, $\mathcal{N}[v] \subset \mathcal{N}[u]$, and $I(v) < I(u)$ **then**
6: $m(v) \leftarrow F$
7: **end if**
8: **if** u and w are two marked neighbors of $v \in V(\mathcal{D})$, $\mathcal{N}(v) \subset \mathcal{N}(u) \cup \mathcal{N}(w)$, $(u, w) \in E(\mathcal{G})$ and $ID(u) = \min\{I(u), I(v), I(w)\}$ **then**
9: $m(v) \leftarrow F$
10: **end if**
11: Create a spanning tree $\mathcal{D}_\mathrm{t} \subset \mathcal{D}$

Proposition 2. Any $(\mathcal{G}, \mathcal{D})$ pair generated using Algorithm 1 has the inter-cluster property.

Proof. After running Algorithm 1, suppose we have two connected unmarked nodes $i, j \in V(\mathcal{G})$, such that $\nexists k \in V(\mathcal{D})$, $i, j \in \mathcal{N}(k)$, i.e., i or j are not inter-cluster nodes. To generate this $\mathcal{D}_\mathrm{t}$, the algorithm proceeded as follows: (1) $m(i) = m(j) = T$ because i and j must have unconnected neighbors; (2) without loss of generality, let $I(i) < I(j)$, causing j to remain a DN, because $I(j) \neq \min\{I(i), I(j), \ldots\}$, by Line 8. This is a contradiction, thus $(i, j) \notin E(\mathcal{G})$. □

Line 11 generates a graph $\mathcal{D}_\mathrm{t}$ that is a spanning tree of $\mathcal{D}$. This will guarantee the MN of the C^1-RG generated from the pair $(\mathcal{G}, \mathcal{D}_\mathrm{t})$, by Theorem 2. Note that the final step of transforming $\mathcal{D}$ into $\mathcal{D}_\mathrm{t}$ requires the construction of a spanning tree across $\mathcal{D}$. We propose the construction of a minimum spanning tree, for which numerous distributed algorithms exist. The corresponding weights on each edge of $\mathcal{D}$ can be defined using a several metrics. In the context of location-aware networks one can use, for example, signal-to-noise ratio (SNR), ranging information intensity [39], or the entropy of the distributions along each edge.

The second step is to generate the set of outer regions $\mathcal{R}_0$ using the factors and variables in the factor graph expansion of $\mathcal{G}$. This done as per the definition of clustered RGs. The third step is to define all inner regions in $\mathcal{R}$, which are generated from $\mathcal{R}_0$ using the Kikuchi

method. To accomplish this, we need to determine which variables and factors in the FG will be allocated to each inner region. Region $r \in \mathcal{R}_0$ is composed of the set of variables V_r, which includes its controlling node i, $\{m \in \mathcal{N}(i) \colon (i,m) \in \mathcal{D}_\mathrm{t}\}$, and $\{n \in \mathcal{N}(i) \cap V(\mathcal{D})^C\}$. Region r also contains the set of factors made up of $\{b(j) \colon j \in V_r\}$; $\{\phi_{im}, \phi_{mi}\}$; and $\{\phi_{in}, \phi_{ni}, \phi_{kn}, \phi_{nk} : k \in V_r, k \in V(\mathcal{D})^C\}$.

We generate the remainder of the region graph $\mathcal{R}$ in a distributed fashion using Algorithm 2. Specifically, each node $i \in V(\mathcal{D})$ that controls a outer region $r \in \mathcal{R}_0$ determines the common zones that exist between it and neighboring dominating nodes in $\mathcal{D}_\mathrm{t}$ (lines 2–15). Finally, lines 20–22 describe the procedure to assign control of each region and to determine which messages in the RG will be sent between physical nodes, and which messages will remain internal to a particular node.

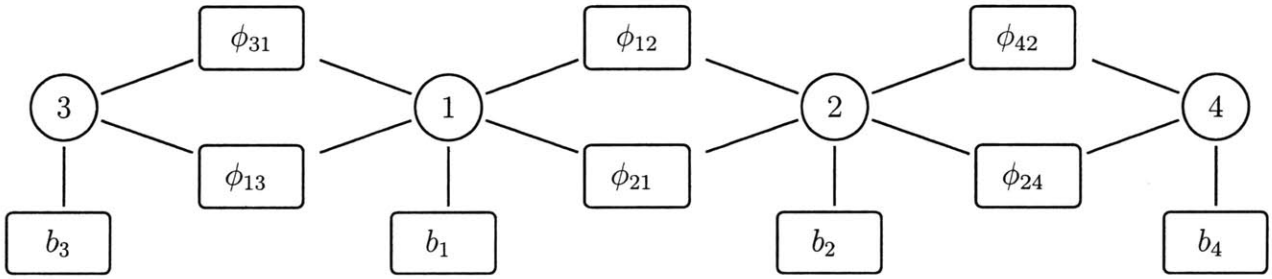
An example of a NRG generated using Algorithm 2 is shown in Fig. 3-3(a): the FG expansion of this network is shown in Fig. 3-3(b). In Fig. 3-3(d) we introduce a compressed notation for NRGs, from which the corresponding variables and factors in each region can be inferred. This notation will be utilized in more elaborate RGs later in the manuscript.

As per Theorem 2, using a C^1-RG together with a CNRG will create a NRG that is MN. In Algorithm 2 we are making a trade-off between MN and the use of all information available in the network: line 11 in Algorithm 2 does not utilize the information in any edges in the subgraph $\mathcal{D}$ that are not also contained in $\mathcal{D}_\mathrm{t}$. Intuitively the information in these discarded edges could aid in inference accuracy. However, as others have observed with GBP algorithms [11,40], attempting to fuse large sets of information can actually hinder inference performance. The design of distributed self-inference algorithms based on GBP that operate at optimal points along this continuum—exploiting all information versus removing "hindering" information—remains an open problem. Nonetheless, the amalgamation of Theorem 2, C^1-RGs, and CNRGs provide access to a new operating points in this spectrum.

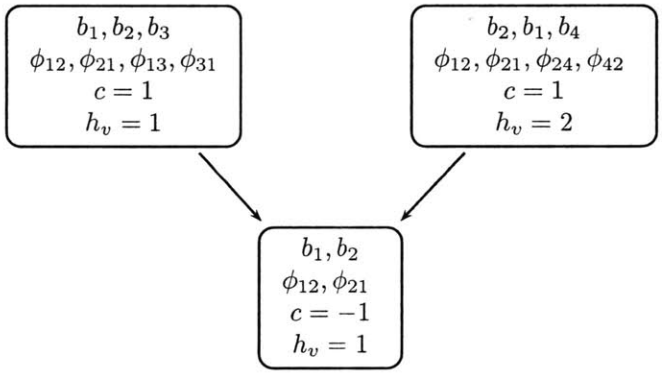
Algorithm 2 Network region graph generation

for $r \in \mathcal{R}_0$ **do**
  Let $h_v(r) = i$. For $j \in \mathcal{R}_0, j \neq r$ node i finds the largest set of variables $\{z_k\}$ such that $z_k \in r, z_k \in j$. Denote this set O_{rj}.
  Append to O_{rj} as follows, without duplication:
  for $z_k \in O_{rj}$ **do** {Subregion creation}
    if $(i, z_k) \in (E(\mathcal{G}) \cap E(\mathcal{D})^c) \cup E(\mathcal{D}_\mathrm{t})$ **then**
      $O_{rj} \leftarrow \{b_i, b_{z_k}, \phi_{iz_k}, \phi_{z_k i}\}$.
    else
      $O_{rj} \leftarrow b_{z_k}$.
    end if
    if $z_k \in \mathcal{N}(i), z_l \in \mathcal{N}(i), (z_k, z_l) \in E(\mathcal{D}_\mathrm{t}) \cup (E(\mathcal{G}) \cap E(\mathcal{D})^c)$ **then**
      $O_{rj} \leftarrow \{b_i, b_{z_k}, \phi_{iz_k}, \phi_{z_k i}\}$.
    end if
    Let $r' \in \mathcal{R}_1, r' \leftarrow O_{rj}$. Connect arcs in the NRG from r to r' and from j to r'.
  end for
end for
for $r \in \mathcal{R}_1$ **do**
  Let $h_v(r) = i$. For $j \in \mathcal{R}_1, j \neq r$ node i finds the largest set of variables and factors $\{z_k\}$ such that $z_k \in r, z_k \in j$. Denote this set O_{rj}.
  Let $r' \in \mathcal{R}_2, r' \leftarrow O_{rj}$. Connect arcs in the NRG from r to r' and from j to r'.
  for $j \in V(\mathcal{D}_\mathrm{t}) \cap \{r\}$ **do** {Control of subregions and internal vs. external message flags}
    if $ID(j) = \min\{ID(j_1), \ldots, ID(j_m)\}$, where j_n is the nth DN in r **then**
      j determines which DN in r will control this region, according to a uniform probability law.
    end if
  end for
end for

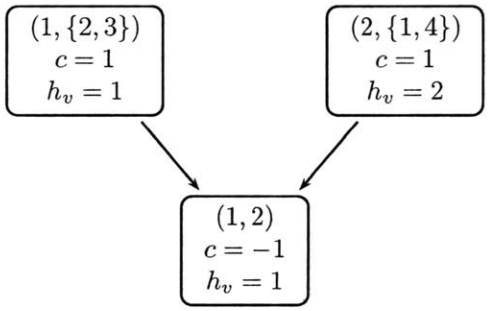
(a) An example network graph composed of four nodes, in which nodes 1 and 2 make up the CDS.



(b) A FG expansion of the network graph shown in Fig. 3-3(a). The belief of node i is represented as b_i, and ϕ_{ij} represents the uncertainty from information sharing between nodes i and j.



(c) Region-graph of Figs. 3-3(a) and 3-3(b), in which we have three regions: two controlled by node **1** and the third controlled by node **2**.



(d) A redrawn region-graph of Fig. 3-3(c), in which we have three regions: two controlled by node **1** and the third controlled by node **2**. Notation has been compressed for readability, such that the relevant factors and variables can be inferred from the included edges. For example, the edge set $(2, \{1, A\})$ denotes edges $(2, 1)$ and $(2, A)$ in $E(\mathcal{G})$.

Figure 3-3: An example of a network graph that is expanded to a factor graph, which is in turn mapped to a 1-clustered region graph.

Chapter 4

A Detailed Example: Distributed GBP for Location-aware Networks

The algorithms presented thus far can be applied to general self-inference problems that utilize region graphs. Different self-inference problems require different variants of GBP and different techniques for message represenatation. In this chapter we will determine an appropriate variant of GBP and an appropriate message representation by applying our framework to location-aware networks. Two primary issues are addressed herein. First, the information extraction model that network nodes are subject to when performing localization is briefly described. Secondly, a technique for representing GBP messages non-parametrically is developed. Although these message representation techniques are designed for location-aware networks, it is likely that ideas in this chapter can be applied to other network-inference problems.

4.1 Information Extraction Model

The measurement model used in this thesis is a ultrawide bandwidth (UWB) ranging model that is utilized in [1,18,41,42]. It operates as follows. Node i obtains information from each of its neighbors $j \in \mathcal{N}(i)$ to estimate the physical distance between nodes i and j. Let $\hat{d}_{ji}$

be a ranging estimate message sent by node j to node i, containing a ranging measurement related to the true distance $d = \|x_i - x_j\|$. Node i uses TOA ranging to form its estimate $\hat{d}_{ji}$ of the true d_{ji}; let the PDF $\phi_{ji} = p(\hat{d}_{ji}|d = \|x_i - x_j\|)$ characterize the uncertainty in the ranging measurements.

To accurately characterize the ranging uncertainty ϕ_{ji} we use the empirical set of distributions developed in [42] for UWB-based TOA ranging in cluttered indoor environments. This distribution has multiple convenient properties. For example if ϕ_{ij} is represented non-parametrically—see Sec. 4.3—we can then generate samples from $\sum_i \phi_{ji} b(i)$ using IS. Ranging uncertainty is represented through a Gaussian mixture sampler distribution with parameters that are a function of d_{ji}. Node i can draw a set of N_s samples $\{d_{ij}^{(k)}\}$ from a sampler distribution $q(d_{ji}|\hat{d}_{ji})$. (The distribution q can be sampled from trivially because node i has knowledge of $\hat{d}_{ji}$.) The parameters of q and a more detailed discussion of this model can be found in [42]. To compute the weights for each $d_{ji}^{(k)}$ we use

$$w^{(k)} \propto \frac{p\left(\hat{d}_{ji}|d_{ji}^{(k)}\right)}{q\left(d_{ji}^{(k)}|\hat{d}_{ji}\right)}, \tag{4.1}$$

which can be evaluated exactly using a parametric representation of q.

4.2 GBP Algorithms

There exist a number of known GBP algorithms that minimize the same free-energy cost function. The two most common variants are the parent-child and the two-way algorithms [11]. In the parent-child algorithm messages in $\mathcal{R}$ only travel from parent regions to their children. On the other hand, in the two-way algorithm messages in $\mathcal{R}$ travel both from parents to regions and from regions to parents. Each GBP variant has its own advantages and disadvantages that depend on the structure of the RGs encountered and on the application of interest.

We now describe selected details of both the parent-child and two-way algorithm to

determine which of these variants is best suited for distributed processing in the context of location-aware networks. For this discussion, define $\mathcal{P}(r)$ as the set of parents of region r, and $\mathcal{C}(r)$ as the set of children of r.

In the parent-child algorithm, m_{pr} denotes the message that travels to region r from parent region $p \in \mathcal{P}(r)$. The message m_{pr} is constructed using the following sets. The set $s(r)$ consists of all descendants of r, and $s[r]$ consists of r and all the descendants of r, i.e., $s[r] = s(r) \cup \{r\}$. Let (i,j) be a connected pair of regions in $\mathcal{R}$. The set $s(p,r)$ is then defined as[1]

$$s(p,r) \triangleq \{(i,j)\colon i \in s[p] \backslash s[r], j \in s[r]\} \backslash \{p,r\}\}\,, \tag{4.2}$$

and the set $n(p,r)$ is defined as

$$n(p,r) \triangleq \{(i,j)\colon j \in s[p],\ j \notin s[r],\ i \notin s[p]\}\,. \tag{4.3}$$

The message m_{pr} is then defined as

$$m_{pr}(\mathbf{x_r}) = \sum_{\mathbf{x}_p \backslash \mathbf{x}_r} \prod_{a \in A_p \backslash A_r} f_a(\mathbf{x_a}) \prod_{(i,j) \in n(p,r)} m_{ij}(\mathbf{x_j}) \Big/ \prod_{(i,j) \in s(p,r)} m_{ij}(\mathbf{x_j})\,. \tag{4.4}$$

We now describe the equivalent message construction for the two-way algorithm to determine which of these GBP variants is most suitable for location-aware networks. The messages for the two-way algorithm are constructed in the following order. Let $q_r \triangleq (1 - c_r)/|\mathcal{P}(r)|$, and $\beta_r \triangleq 1/(2 - q_r)$.[2] To construct GBP messages firstly compute

$$f_r^c(\mathbf{x}_r) \triangleq \left(\prod_{a \in A_r} f_a(\mathbf{x}_a) \right)^{c_r}\,. \tag{4.5}$$

Pseudo-messages n_{rp}^0 and m_{rc}^0 are then constructed between region r and each parent region p or child region c, respectively. The structure of these messages is closely related to ordinary

[1]The reader should note that the definition of $s(p,r)$ differs from the equivalent definition proposed in [11]

[2]When a region has no parent, i.e. $|\mathcal{P}(r)| = 0$, we set $q_r = \beta_r = 1$.

BP messages. Specifically,

$$n_{rp}^0(\mathbf{x}_r) = f_r^c(\mathbf{x}_r) \prod_{p' \in \mathcal{P}(r) \setminus p} m_{p'r}(\mathbf{x}_r) \prod_{c \in \mathcal{C}(r)} n_{cr}(\mathbf{x}_c)\,, \tag{4.6}$$

and

$$m_{rc}^0(\mathbf{x}_c) = \sum_{\mathbf{x}_r \setminus \mathbf{x}_c} f_r^c(\mathbf{x}_r) \prod_{p \in \mathcal{P}(r)} m_{pr}(\mathbf{x}_r) \prod_{c' \in \mathcal{C}(r) \setminus c} n_{c'r}(\mathbf{x}_{c'})\,. \tag{4.7}$$

Note that message n_{rp}^0 does not require marginalization because by definition $\{\mathbf{x}_c\} \subset \{\mathbf{x}_r\}$. The final GBP messages to be passed along the edges of $\mathcal{R}$ are given by

$$n_{rp}(\mathbf{x}_r) = \left(n_{rp}^0(\mathbf{x}_r)\right)^{\beta_r} \left(m_{pr}^0(\mathbf{x}_r)\right)^{\beta_r - 1} \tag{4.8}$$

and

$$m_{rc}(\mathbf{x}_c) = \left(n_{cr}^0(\mathbf{x}_c)\right)^{\beta_r - 1} \left(m_{rc}^0(\mathbf{x}_c)\right)^{\beta_r}\,. \tag{4.9}$$

The belief of region r can then be computed using

$$b_r(\mathbf{x}_r) = f_r^c(\mathbf{x}_r) \prod_{c \in \mathcal{C}(r)} n_{cr}(\mathbf{x}_c) \prod_{p \in \mathcal{P}(r)} m_{pr}(\mathbf{x}_p)\,. \tag{4.10}$$

To compare the parent-child and two-way algorithms in the context of distributed processing and location-aware networks, we make the following remarks.

- Unlike the two-way algorithm, the parent-child algorithm requires knowledge of messages in addition to those that travel directly in and out of region r through the sets $s(p,r)$ and $n(p,r)$. This can create complications for the controlling node i for region r, because i needs to obtain all messages m_{ij}, where $(i,j) \in s(p,r)$.[3]

- Unlike the parent-child algorithm, the messages for each region r in the two-way algorithm require knowledge of counting number c_r.

[3]Node i could could obtain these messages through routing information in $\mathcal{G}$.

- As shown in (4.4), the messages in the parent-child algorithm require taking ratios of messages, which can be non-trivial depending on the representation of messages [17]. Although, the two-way algorithm does not require taking ratios of messages, it requires the exponentiation of some intermediate or pseudo-messages.

The two-way algorithm is particularly attractive for distributed processing due to each region $r \in \mathcal{R}$ only requiring knowledge of messages going directly into r. For this reason, we will exclusively use the two-way algorithm for distributed location-aware networks in the remainder of this manuscript.

4.3 Message Representation Techniques

As per the previous chapter, the messages that are transmitted in GBP are functions of probability distributions, and these distributions must be represented in a manner that can be transmitted as packets by physical network nodes. Ultimately, the chosen technique of message representation has a high effect on the accuracy of inference and on the complexity of the algorithm under test. In traditional communications problems, such as decoding, messages can be represented efficiently and exactly through, for instance, log-likelihood ratios [43]. Due to the complexity of GBP messages, exact parametric representation is not feasible in a large class of inference problems, so we must resort to approximate message representation [1, 18]. Any representation must be able to capture the salient properties of the true message, and must enable efficient computation of the algorithm's key steps. We consider three types of message representation: parametric, discretized, and sample-based.

4.3.1 Parametric Message Representation

If messages can be represented exactly using parametric functions then the manipulation of GBP messages is greatly simplified and can become highly computationally efficient. In the context of location-aware networks, there is no known method of representing beliefs parametrically in an optimal fashion, so messages can instead be approximated using functions.

For example, in [41], we exploited the known shape of singe-agent beliefs in two-dimensions by constructing a parametric $\mathcal{D}$-distribution for a each agent's belief. The $\mathcal{D}$-distribution is only applicable in homogeneous environments of infinite size. The PDF of each node's belief is defined as

$$\mathcal{D}(\mathbf{x}; m_1, m_2, \sigma^2, \rho) \propto \exp\left\{-\frac{1}{2\sigma^2}\left[\sqrt{(x_1 - m_1)^2 + (x_2 - m_2)^2} - \rho\right]^2\right\}, \tag{4.11}$$

where $[m_1, m_2]$ is the midpoint of the distribution, ρ is the radius, σ^2 is the variance, and the proportionality sign denotes a normalization constant not shown. As a special case, note that if $\rho = 0$ then (4.11) collapses to a two-dimensional Gaussian.

Analysis of the errors introduced into the localization process by such approximations is an open problem and, when beginning to explore the use of GBP algorithms for location-aware networks, this motivates alternative message representations that will increase localization accuracy, even at the sacrifice of some computational complexity.

4.3.2 Discretized Message Representation

A simple method of representing the beliefs $\{b(i)\}$ and ranging functions ϕ_{ij} is to uniformly discretize the domain of each function. For example, to represent the belief $b(i)$ of node i, we divide the environment up into a set of N_s quantized points $\{1, \ldots, N_s\}$. The distribution $b(i)$ is then approximated as a finite list of values, $\{b(i)\}$.

The main complications of discretized message representation are that (1) N_s scales exponentially with the dimensionality of i; and (2) an extremely large number of points is required to capture the fine features of GBP messages and joint beliefs. Due to the accurate ranging of UWB transmission, the number of points required to capture ranging estimates is prohibitively high in UWB-based location-aware networks. For example, in a 100m by 100m environment with 1 cm resolution—typical for a UWB system—N_s = 100 million is required. This is not only impractical from a computational standpoint, but also infeasible to communicate messages with so many values between nodes. These drawbacks motivate

us to find an alternative method of representing beliefs as a discrete set of samples.

4.3.3 Particle-based Message Representation

In location-aware networks, the flexibility and computational scalability offered by particle-based messages makes this implementation strategy particularly attractive. Our particle methods will be based on IS due to the low-dimensionality of region messages and due to the existence of natural sampling distributions for each ϕ_{ij}, as discussed in Sec. 4.1.

If all joint beliefs $\{b_r\}$ are represented using equally-weighted samples, then we need to manipulate these samples as described by (4.5)–(4.9). The following section describes techniques to represent such messages.

4.4 GBP Message Construction

We assume that each node i has belief $b(i)$ represented by N_s equally-weighted samples. As per (4.5), a set of particles needs to be multiplied together and then exponentiated by c_r.

Each term $f_a(\mathbf{x}_a)$ is either an agent's belief $b_i(i)$ or a ranging distribution $\phi_{ij}(i,j)$, and to multiply each $f_a(\mathbf{x}_a)$ we take the following steps. We arrange the functions in A_r into disjoint "sampling groups." Specifically, let $\{B_i\}$ be a collection of disjoint subsets of A_r such that $\bigcup_i B_i = A_r$. For each belief $b_i \in A_r$ we construct a unique $B_i = \Phi_i \cup b_i$, where $\Phi_i = \{\phi_{ij}(i,j_1),\ldots,\phi_{ij}(i,j_N)\}$, i.e., B_i is made up of node i's belief and all factors that represent transmissions going out of node i. To sample from $B_i = b_i(i)\phi_{ij}(i,j_1)\cdots\phi_{ij}(i,j_N)$ we use IS. Set $b_i(i)$ as the sampler distribution and then let

$$w^{(k)} = \phi_{ij}(i^{(k)}, j_1^{(k)})\cdots\phi_{ij}(i^{(k)}, j_N^{(k)})\,. \tag{4.12}$$

Similar to [42], we now sample from $\prod_{i\in V_r} B_i$ using IS with the sum $\sum_{i\in V_r} B_i$ as a sampler distribution. This creates N_s equally weighted samples that represent $f_r(b_r)$. The interested reader is referred to [42] for details.

The next step in constructing a GBP message requires exponentiating the N_s equally-weighted particles that represent $f_r(j) = \prod_{a \in A_r} f_a(\mathbf{x}_a)$ by c_r, as per (4.5).[4] Raising a belief to a positive power should concentrate the distribution toward the regions with a high density of particles. We propose the following method to raise a set of particles to a power. Similar to [1, 17, 18], we use kernel density estimation (KDE) to approximate $f_r(j)$ as a Gaussian mixture, i.e.,

$$f_r(j) \approx \sum_k w^{(k)} \mathcal{N}(j^{(k)}, \sigma^2), \tag{4.13}$$

where σ^2 is the variance as per KDE and $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian PDF with mean μ and variance σ^2. When the number of samples N_s is large then computing $f_r^c(j)$ for each sample k directly becomes computationally expensive so we require a simplifying approximation for this step; we do so as follows. Taking logs of both sides of (4.13) we obtain

$$\begin{aligned} \ln f_r(j) &= a_1 + \ln \left\{ \sum_k \exp(\lambda^{(k)}) \exp\left(-(j - j^{(k)})^2 / 2\sigma^2\right) \right\} \\ &= a_1 + \mathcal{M}(L^{(1)}(j) + \lambda^{(1)}, \ldots, L^{(N_s)}(j) + \lambda^{(N_s)}), \end{aligned} \tag{4.14}$$

where $L^{(k)}(j) = -(j - j^{(k)})^2 / 2\sigma^2$, $w^{(k)} = \exp\left(\lambda^{(k)}\right)$, a_1 is a constant, and $\mathcal{M}$ is the Jacobian logarithm defined as

$$\mathcal{M}(x^{(1)}, \ldots, x^{(N_s)}) \triangleq \ln \left\{ \sum_k \exp\left(x^{(k)}\right) \right\}. \tag{4.15}$$

Note that $\mathcal{M}$ is also recursive, i.e., $\mathcal{M}(x^{(1)}, \ldots, x^{(N_s)}) = \mathcal{M}(x^{(1)}, \mathcal{M}(x^{(2)}, \ldots, x^{(N_s)}))$. In typical location-aware networks, the number of samples N_s used to represent $f_r(j)$ is small so $\|L^{(i)} + \lambda^{(i)} - L^{(k)} - \lambda^{(k)}\|$ is usually large if $i \neq k$. (Typical systems use 500–2000 samples [18, 19, 43].) Substituting the kth sample $j^{(k)} \in \{f_r(j)\}$ into (4.14) and using the recursive property of $\mathcal{M}$ we obtain [44, 45]

$$\ln f_r(j^{(k)}) = a_2 + \max\left\{ L^{(1)}(j^{(k)}) + \lambda^{(1)}, \ldots, L^{(N_s)}(j^{(k)}) + \lambda^{(N_s)} \right\} \tag{4.16}$$

[4] In this case j is a random vector that includes all variables in V_r.

where a_2 is a constant. Eq. (4.16) is similar to approximating a Gaussian mixture distribution by its most dominant component. Now let us define $L_{\max}(j^{(k)}) = \max_{i \neq k} \left\{ -(j^{(i)} - j^{(k)})^2 \right\}$, and if we have equally weighted samples then

$$\ln f_r(j^{(k)}) \approx a_2 + L_{\max}(j^{(k)}) + \lambda^{(k)} . \tag{4.17}$$

This shows that we can approximate the height of $f_r(j^{(k)})$ in the log-domain as a function of $L_{\max}(j^{(k)})$. We use this fact to approximate the height of $f_r(j^{(k)})^{c_r}$ by defining $w^{(k)c_r} \triangleq \exp\left\{(c_r - 1)L_{\max}(j^{(k)})\right\}$; intuitively, if c_r is positive then the closer the nearest sample to $j^{(k)}$, the greater its weight will become. The weight of the exponentiated distribution will then be focused around regions in which samples are close to one-another, as desired. If c_r is negative, then the opposite effect will occur. Algorithm 3 summarizes the exponentiation of a set of equally weighted samples to approximate $f_r^c(\mathbf{x}_r)$.

Algorithm 3 The exponentiation of equally-weighted samples $\{f_r(j^{(k)})\}$ to a power c_r.

1: Perform KDE on the equally weighted samples and obtain a variance σ^2
2: **for** samples $k = 1$ to N_s **do**
3: Find $L_{\max}(j^{(k)}) = \max_{k \neq i} \left\{ -(j^{(i)} - j^{(k)})^2 \right\}$
4: Find weights:
$w^{(k)c_r} = \exp\left\{(c_r - 1)L_{\max}(j^{(k)})\right\}$
5: **end for**

The next step in manipulating the set of particles $\{\mathbf{x}_r^{(k)}\}$ is to multiply $\{\mathbf{x}_r^{(k)}\}$ with sets of messages incoming to r, as per (4.6) and (4.7). To multiply messages we first approximate each message in the sets $\mathcal{P}(r)$ and $\mathcal{C}(r)$ using KDE (this allows us to evaluate the multiplication of these messages at any given point). Finally, the messages $n_{rp}(\mathbf{x}_r)$ and $m_{rc}(\mathbf{x}_c)$ are computed by exponentiating this set of samples by β_r, as per (4.8) -(4.9). Algorithm 4 summarizes these steps to construct a particle-based GBP message.

Algorithm 4 The two-way GBP algorithm for region r.

1: Multiply factors in r using KDE and then resample to create a set of equally weighted particles
2: Compute $f_r^c(\mathbf{x}_r)$ using Algorithm 3
3: Receive all incoming messages from neighboring regions
4: Multiply incoming messages with $f_r^c(\mathbf{x}_r)$ as per (4.8) and (4.9)
5: Use KDE to approximate the multiplication of these messages and then resample to create a set of equally-weighted particles
6: Compute β_r
7: For each sample in $m_{rc}^0(\mathbf{x}_c)$, discard dimensions of (4.9) that are not required
8: Exponentiate the pseudo-messages by β_r and $\beta_r - 1$, as per (4.8) and (4.9)
9: Multiply the exponentiated pseudo-messages using KDE and then resample to create a set of equally weighted particles
10: Compute the belief of region r by using (4.10)

4.5 An Example of Particle-based Message Representation

We now describe an example CNRG that was generated from a network graph using a subgraph $\mathcal{D}_\mathrm{t}$. We then give instances of GBP messages that flow in this CNRG.

Consider Fig. 4-1, which shows a network graph of 10 nodes. The shaded nodes form a CDS and a subgraph $\mathcal{D}_\mathrm{t} \subset \mathcal{G}$ is denoted by shaded nodes and double-line edges. The subgraph $\mathcal{D}_\mathrm{t}$ was generated using Algorithm 1, as outlined in Sec. 3.1. In this location-aware network, agents/nodes that do not know their position are denoted by numbers 1–3 and anchors that know their position are denoted A–G. The subgraph $\mathcal{D}_\mathrm{t}$ forms a tree so the edge $(1, B)$ will not be included in the CNRG, and this guarantees the CNRG generated from $(\mathcal{G}, \mathcal{D}_\mathrm{t})$ will be MN. Note that both agents and anchors may be dominating nodes included in $V(\mathcal{D})$. In a location-aware network, each agent requires at least three links to localize without ambiguity, and Fig. 4-1(a) meets this requirement. Anchors A–G know their positions *a priori* so no ranging information needs to be transmitted to them by their neighboring agents. This means that the edges connecting anchor i to agent j consist of only a single $\phi_{ij}(i, j)$ function from i to j, and that $\phi_{ji}(j, i) \equiv 0$.

We now turn our attention to Fig. 4-1(b), a CNRG generated from Fig. 4-1(a) that is

MN by Theorem 2. The set of outer regions $\mathcal{R}_0$ is shown in the first two rows of Fig. 4-1(b). Each $r \in \mathcal{R}_0$ is generated by clustering or grouping the neighbors of one DN. All remaining regions are generated using the Kikuchi method, as per Algorithm 2.

To construct GBP messages using Algorithm 4, let us denote the following three regions each in $\mathcal{R}_0$: let r_{12A} denote the region containing edges $(1, A)$ and $(1, 2)$, let r_{23G} denote the region containing edges $(2, G)$ and $(2, 3)$, and let r_{32B} denote the region containing edges $(3, 2)$ and $(3, B)$. As an example, we define the sets A_r for these regions explicitly,

$$A_{r_{12A}} = \{b_1, b_2, b_A, \phi_{A1}, \phi_{12}, \phi_{21}\} \tag{4.18}$$

$$A_{r_{23G}} = \{b_1, b_2, b_3, b_G, \phi_{12}, \phi_{21}, \phi_{G2}, \phi_{23}, \phi_{32}\} \tag{4.19}$$

$$A_{r_{32B}} = \{b_3, b_2, b_B, b_E, b_F, \phi_{23}, \phi_{32}, \phi_{3B}, \phi_{B3}, \phi_{E3}, \phi_{FE}\}\,. \tag{4.20}$$

The next step requiring constructing the set $\{B_i\}$ for each A_r above. As an example, let us consider only r_{32B}, so $A_{r_{32B}} = \bigcup_i B_i$ is constructed as

$$\begin{aligned} A_{r_{32B}} =& \{b_3, \phi_{32}, \phi_{3B}\} \cup \{b_2, \phi_{23}\} \cup \{b_B, \phi_{B3}\} \\ & \cup \{b_E, \phi_{E3}\} \cup \{b_F, \phi_{FE}\}\,. \end{aligned} \tag{4.21}$$

The other $\{B_i\}$ sets associated with each region can be constructed in a similar manner. The sampling distribution for $B_3 = b_3\phi_{32}\phi_{3B}$ is then b_3 and the weight of the kth sample of b_3 is given by

$$\begin{aligned} w^{(k)} &= p/q \\ &= b_3\phi_{32}(x_3^{(k)}, x_2^{(k)})\phi_{3B}(x_3^{(k)}, x_B^{(k)})/b_3(x_3^{(k)}) \\ &= \phi_{32}(x_3^{(k)}, x_2^{(k)})\phi_{3B}(x_3^{(k)}, x_B^{(k)})\,. \end{aligned} \tag{4.22}$$

The remaining sets and samples are constructed accordingly and Algorithm 3 is then used to compute $f_{r_{32B}}(\mathbf{x}_{r_{32B}})$ as per (4.5).

Let us now construct a message from the region r_{23} containing the edge $(2, 3)$ in $\mathcal{R}_1$, shown

in the third row of Fig. 4-1(b). To construct this message, let r_3 denote the region in $\mathcal{R}_2$ containing only node 3 and let r_2 denote the region in $\mathcal{R}_2$ containing only node 2. Let us now construct the message $m_{r_{23}r_2}$. We compute $\beta_{r_{23}} = 1$. As per (4.9), $m_{r_{23}r_2}(\mathbf{x}_{r_2}) = m^0_{r_{23}r_2)}(\mathbf{x}_{r_2})$ and

$$m_{r_{23}r_2}(\mathbf{x}_{r_2}) = \sum_{x_3} f^c_{r_{23}}(x_2, x_3) m_{r_{23G}r_{23}}(x_2, x_3) m_{r_{32B}r_{23}}(x_2, x_3) n_{r_3r_{23}}(x_3)\,. \tag{4.23}$$

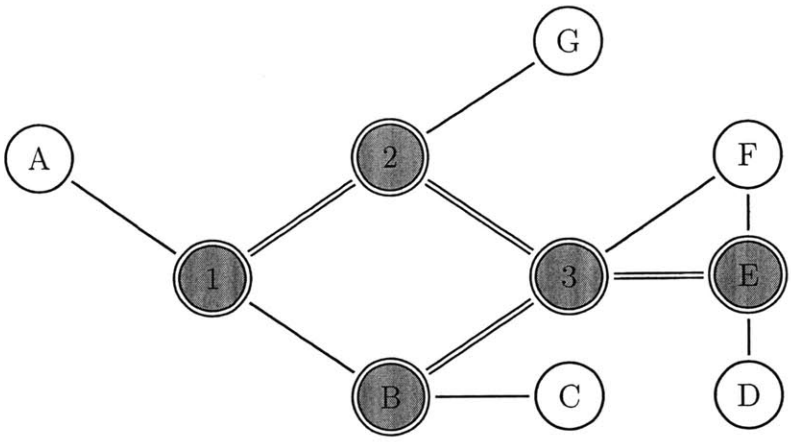
The belief of region r_{23} is given by

$$b_{r_{23}}(x_2, x_3) = f^c_{r_{23}}(x_2, x_3) n_{r_2r_{23}}(x_2) n_{r_3r_{23}}(x_3) m_{r_{21G}r_{23}}(x_2, x_3) m_{r_{32B}r_{23}}(x_2, x_3)\,. \tag{4.24}$$

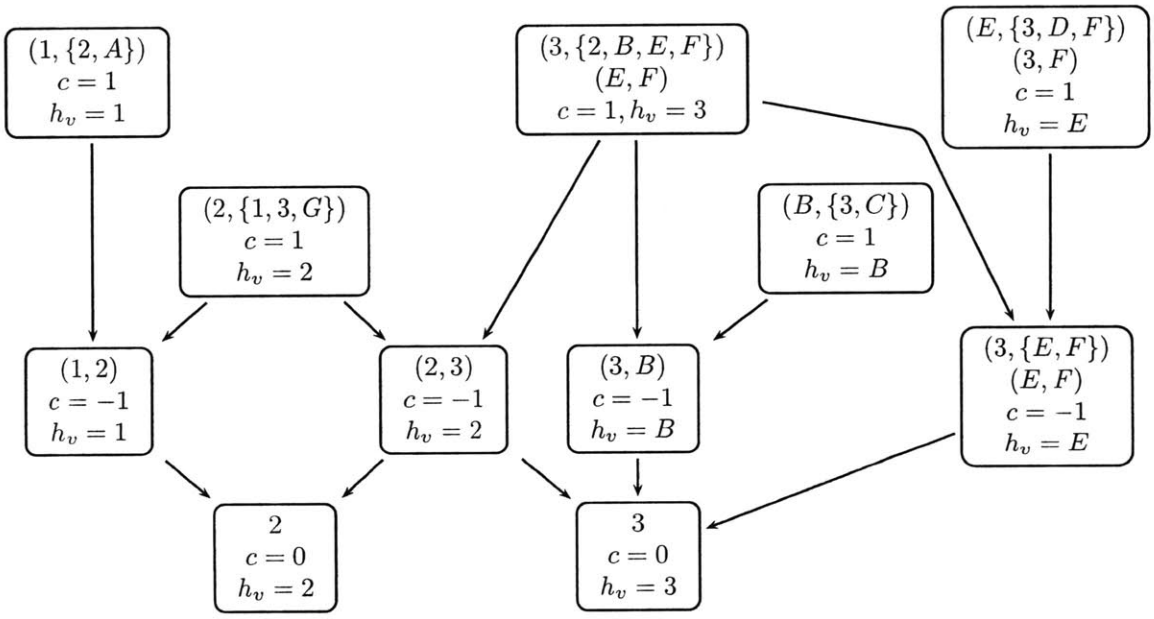
Note that when implementing GBP in any location-aware network it is only required to compute the beliefs of the outer regions. For instance, to compute the belief $b_{r_{23}}(x_2, x_3)$, one can equivalently compute the belief $b_{r_{23}}$ through $b_{r_{32B}}(x_2, x_1, x_3, x_G)$

$$b_{r_{32B}}(x_2, x_1, x_3, x_G) = f^c_{r_{32B}}(x_2, x_1, x_3, x_G) \prod_{c \in C(r_{23})} n_{(c)(r_{23})}(\mathbf{x}_c)\,. \tag{4.25}$$

The beliefs of r_{23} would then be represented by 4-dimensional samples and so to compute $b_{r_{23}}$ one can discard the dimensions that represent x_1, x_G.



(a) A network graph for cooperative localization. Nodes that do not know their position are denoted by the numbers 1–3 and anchors that have knowledge of their position are denoted by the letters A–G. Nodes 1–3 have at least three neighbors each so that they can all be localized without ambiguity.



(b) A CNRG generated from the network graph $\mathcal{G}$, as depicted in Fig. 4-1.

Figure 4-1: An example of a mapping from a network graph to a CNRG.

Chapter 5

Variants of BP for Location-aware Networks

Chapter 3 developed a framework for GBP algorithms for self-inference problems, and Chapter 4 demonstrated the GBP algorithms with particle-based messages for location-aware networks. The goal of this chapter is to show the performance of simplified GBP algorithms, that use particle-based messages, and to compare them with conventional localization algorithms.

There is a large number of parameters that need to be optimized when implementing any GBP-based algorithm in location-aware networks. Numerous questions remain about how to find optimized or appropriate parameter values for such networks. This holds true for both GBP as well as BP algorithms, GBP's simplified counterpart. For BP algorithms in particular, issues such as agent mobility parameters, methods to overcome the potential lack of localization infrastructure, the degree of inter-agent cooperation, and computational complexity are all open questions. Prior to addressing these issues for GBP algorithms, such areas should be understood for the simplified BP counterparts. Hence, to begin exploring this space and to shed insight on these issues, in this chapter we will simplify our GBP algorithms to BP.

Key strides can be made to the full deployment of BP algorithms by overcoming the

following two key limitations. First, all previously proposed BP algorithms for location-aware networks have been restricted to static networks. However, the majority of location-aware networks are highly mobile. Second, there exists a lack of understanding of the effect of varying degrees of cooperation—a key component of computational complexity—on localization accuracy.

This chapter will explore the issues of mobile networks and computational complexity by

- extending and quantifying the performance of existing BP-based algorithms for localization [1], by allowing for mobile agents and by removing all fixed infrastructure; and

- proposing a set of heuristics that reduce the computational complexity of BP-based algorithms by systematically reducing the quantity of messages passed in the network.

The algorithms explored in this chapter are those presented by the author in [1, 18, 41]. In these papers, the inference back-bone defined by subgraph $\mathcal{D}$ is equal to $\mathcal{G}$, and as shown in [11], all BP region graphs are maxent-normal (MN). We now address the following problem.

Consider a wireless network of N nodes. We denote the position of node i at the beginning of time slot t as $i^{(t)}$, and the aggregated positions at time t as $\mathbf{x}^{(t)}$. In general $\mathbf{x}^{(t)}$ may include other information such as orientation and velocity for all or for a subset of the nodes. The set of nodes from which node i can receive transmissions at time slot t is denoted by $\mathcal{N}_i^{(t)}$. At time t node i receives packets from node $j \in \mathcal{N}_i^{(t)}$ and extracts signal metric $\hat{d}_{ji}^{(t)}$. Further, node i may estimate its own movement from time slot $t-1$ to time slot t using a local signal metric $d_{i,\text{self}}^{(t)}$ (obtained from an odometer, for example). We represent all measurements in the network up to time t by[1] $\hat{\mathbf{d}}^{(1:t)}$, which can be decomposed into $\hat{\mathbf{d}}_{\text{self}}^{(1:t)}$ and $\hat{\mathbf{d}}_{\text{rel}}^{(1:t)}$. The vector $\hat{\mathbf{d}}_{\text{self}}^{(1:t)}$ denotes all intranode measurements while $\hat{\mathbf{d}}_{\text{rel}}^{(1:t)}$ denotes all internode measurements.

[1] Similarly, $\mathbf{x}^{(0:t)}$ denotes $\left[\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(t)}\right]$.

5.1 BP in Mobile Networks

In this section we extend and quantify the performance of existing BP-based algorithms for localization [1], by allowing for mobile agents and by removing all fixed infrastructure. The results presented in this chapter can be found in [18]. In [1], a BP-based algorithm was presented for self-tracking with anchors or existing infrastructure. In this work, the work in [1] is extended to remove the infrastructure/anchor requirement.

We assume that each node i knows the following probability distributions:

- $b\big(i^{(0)}\big)$;
- $p\big(i^{(t)}\big|i^{(t-1)}, \hat{d}^{(t)}_{i,\text{self}}\big)$ for $t > 0$; and
- $p\big(\hat{d}^{(t)}_{ji}\big|i^{(t)}, j^{(t)}\big)$ for $t > 0$.

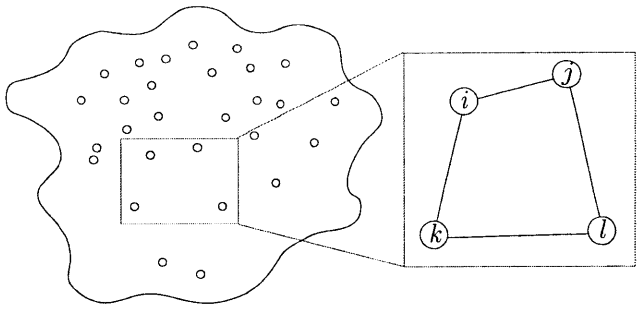
The goal of each node i is to estimate its position $i^{(t)}$ at time t.

5.1.1 BP Algorithm Description

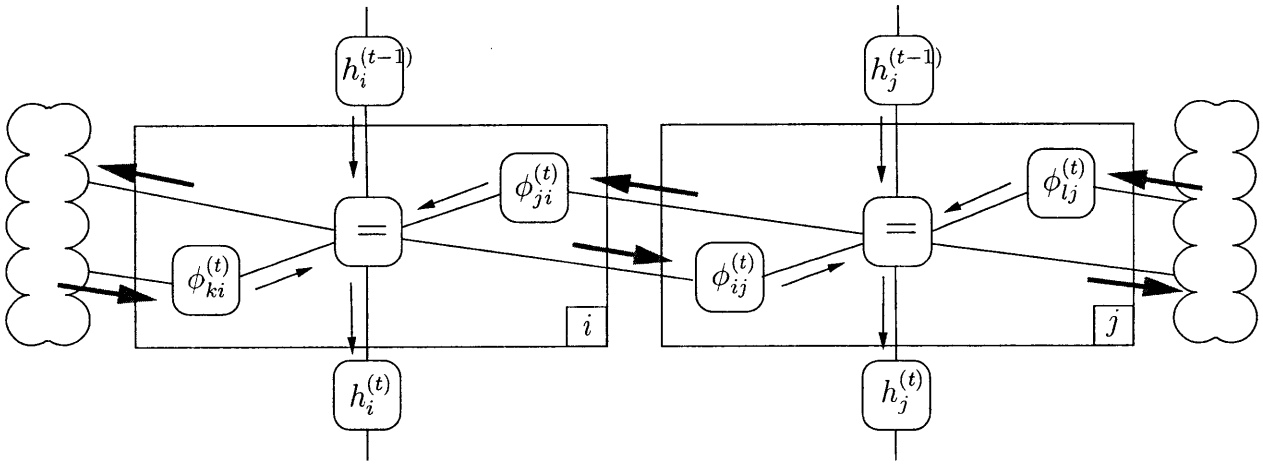
The reader is reminded that although BP is a special case of GBP, it is natural for BP messages to be defined as traveling across a factor graph, as opposed to a region graph. We use this convention here and so factor graph messages will be denoted by μ (as opposed to m, which is reserved for the region graph counterparts in earlier chapters). Our BP algorithm is derived as follows: we create a factor graph of the factorization of $p\big(\mathbf{x}^{(0:t)}|\hat{\mathbf{d}}^{(1:t)}\big)$ and map this factor graph onto the network topology. The factors are then grouped via the local information available at each node, as demonstrated in Fig. 5-1. We then introduce a message schedule that accounts for time-varying network connectivity.

Due to conditional independence, $p\big(\mathbf{x}^{(0:t)}|\hat{\mathbf{d}}^{(1:t)}\big)$ can be written as

$$p\big(\mathbf{x}^{(0:t)}|\hat{\mathbf{d}}^{(1:t)}\big) \propto \prod_{u=1}^{t} p\big(\mathbf{x}^{(u)}\,\big|\,\mathbf{x}^{(u-1)}, \hat{\mathbf{d}}^{(u)}_{\text{self}}\big)p\big(\hat{\mathbf{d}}^{(u)}_{\text{rel}}\,\big|\,\mathbf{x}^{(u)}\big)$$



(a) An example network of homogeneous nodes at time t operating within some environment, where each node is depicted as a circle. Four nodes in the network i, j, k and l are highlighted and their network connectivity is shown as internode lines, e.g.) $\mathcal{N}_i^{(t)} = \{j, k\}$.



(b) A segment of the time-varying factor graph that corresponds to of Fig. 5-1(a). The clouds in the figure indicate the remainder of the factor graph outside of i and j.

Figure 5-1: A network of nodes placed in an environment. A subset of the network connectivity is shown in Fig. 5-1(a) and a portion of the corresponding factor graph is shown in Fig. 5-1(b). The partial factor graph of a $p\big(\mathbf{x}^{(t)}|\hat{\mathbf{d}}^{(t-1)}\big)$ shows only the vertices that are mapped to nodes i and j. We have introduced $h_i^{(t)} = p\big(i^{(t+1)}|i^{(t)}, \hat{d}_{i,\text{self}}^{(t+1)}\big)$ and $\phi_{ji}^{(t)} = p\big(\hat{d}_{ji}^{(t)}|i^{(t)}, j^{(t)}\big)$. The thin arrows denote messages within a node while the bold arrows indicate internode messages.

where $p\big(\mathbf{x}^{(u)}|\mathbf{x}^{(u-1)}, \hat{\mathbf{d}}_{\text{self}}^{(u)}\big)$ and $p\big(\hat{\mathbf{d}}_{\text{rel}}^{(u)}|\mathbf{x}^{(u)}\big)$ can further be factorized as $\prod_{i=1}^{N} p\big(i^{(u)}|i^{(u-1)}, \hat{d}_{i,\text{self}}^{(u)}\big)$ and $\prod_{i=1}^{N}\prod_{j\in\mathcal{N}_i^{(u)}} p\big(\hat{d}_{ji}^{(u)}|i^{(u)}, j^{(u)}\big)$, respectively.

At time t nodes only have available past signal metrics $\hat{\mathbf{d}}^{(1:t)}$ so messages on the factor graph must flow from past to present (from top to bottom in Fig. 5-1). In Fig. 5-1, we

have abbreviated $p\big(i^{(t+1)}|i^{(t)}, \hat{d}_{i,\text{self}}^{(t+1)}\big)$ by $h_i^{(t)}$. At time t, node i first computes $\mu_{h_i^{(t-1)}i^{(t)}}\big(x_i^{(t)}\big)$, based on the message $\mu_{i^{(t-1)}h_i^{(t-1)}}\big(i^{(t-1)}\big)$ and its own mobility model $p\big(i^{(t)}|i^{(t-1)}, \hat{d}_{i,\text{self}}^{(t)}\big)$; no information from other nodes is required. We call this the *prediction* step. Node i then uses the message $\mu_{h_i^{(t-1)}i^{(t)}}\big(i^{(t)}\big)$ and all the relative signal metrics at time t to compute $\mu_{i^{(t)}h_i^{(t)}}\big(i^{(t)}\big)$. We call this the *correction* step. The exchange of information between nodes is depicted in Fig. 5-1 by bold arrows. By proper scheduling, the internode messages sent by node i do not depend on the recipient node, and thus node i can *broadcast* its messages.

Algorithm 5 Outline of BP-based localization for mobile networks

1: **nodes** $i = 1$ to N **in parallel** {prediction step}
2:

$$\mu_{h_i^{(t-1)}X_i^{(t)}}\left(i^{(t)}\right) \propto \sum_{i^{(t-1)}} p\left(i^{(t)}|i^{(t-1)}\hat{d}_{i,\text{self}}^{(t)}\right) \mu_{X_i^{(t-1)}h_i^{(t-1)}}\left(i^{(t-1)}\right)$$

3: $b^{(0)}\big(i^{(t)}\big) = \mu_{h_i^{(t-1)}X_i^{(t)}}\big(i^{(t)}\big)$
4: **end parallel**
5: node i receives packets from $j \in \mathcal{N}_i^{(t)}$ and extracts signal metrics $\hat{d}_{ji}^{(t)}$ $\forall i, j$
6: **for** $l = 1$ to N_{iter} **do** {iterative correction step}
7: **nodes** $i = 1$ to N **in parallel**
8: broadcast $b^{(l-1)}\big(i^{(t)}\big)$
9: $\forall j \in \mathcal{N}_i^{(t)}$: receive $b^{(l-1)}\big(j^{(t)}\big)$ and convert to a distribution over $i^{(t)}$

$$\mu_{\phi_{ji}^{(t)}i^{(t)}}^{(l)}\left(i^{(t)}\right) \propto \sum_{j^{(t)}} p\left(\hat{d}_{ji}^{(t)}|i^{(t)}, j^{(t)}\right) b^{(l-1)}\left(j^{(t)}\right)$$

10: update belief

$$b^{(l)}\left(i^{(t)}\right) \propto \prod_{j\in\mathcal{N}_i^{(t)}} \mu_{\phi_{ji}^{(t)}i^{(t)}}^{(l)}\left(i^{(t)}\right) b^{(l-1)}\left(i^{(t)}\right)$$

11: **end parallel**
12: **end for**
13:

$$\mu_{i^{(t)}h_i^{(t)}}\left(i^{(t)}\right) = b^{(N\text{iter})}\left(i^{(t)}\right) \ \forall i$$

The BP-based algorithm at time t is outlined in Algorithm 5, with both the prediction

step (lines 1–4) and correction step (lines 6–13). Due to cycles in the factor graph, the correction step is iterative. In Algorithm 5, we have introduced $b^{(l)}\big(i^{(t)}\big)$, the *belief* of node i at iteration l of the correction step during time slot t.

5.1.2 Numerical Results

We now evaluate the performance of the BP-based algorithm, relative to the commonly used cooperative least squares (CLS) algorithm used in [46], for location-aware networks with mobile agents without infrastructure. Variations of the CLS algorithm can be found in [47,48] and a more general overview of least squares techniques in the context of localization can be found in [49]. CLS is an iterative algorithm whereby nodes exchange point estimates of their position, and update their position estimates to minimize the least-squares cost function. While CLS is not a Bayesian algorithm, CLS can cope with limited mobility by setting initial position estimates at every time slot to be the final position estimates from the previous time slot. The BP algorithm uses the minimum-mean-square-error (MMSE) estimator to obtain a position estimate after each time step. The performance criterion to compare algorithms is *outage probability*: for a certain scenario (mobility model, number of agents, time index t), and a certain allowable error e (say, 1 meter) an agent is said to be in outage when its position error $\|i - \hat{i}\|$ exceeds e. The outage probability is then given by

$$P_{\text{out}}(e) = E\left[\mathcal{I}\{\|i - \hat{i}\| > e\}\right] \tag{5.1}$$

The expectation in (5.1) is taken with respect to the locations of the agents. Results have been obtained via Monte Carlo simulations with a network of 100 mobile agents located in a 100m by 100m area. All results are averaged over 30 random network topology instantiations. Every agent i moves from time slot $t-1$ to time slot t in a direction $\theta_i^{(t)}$ that is uniformly drawn from $[0, 2\pi)$ and crosses a distance $d_i^{(t)}$ that is drawn from a Gaussian distribution with mean zero and standard deviation $\sigma_{\text{mob}} \in \{1, 10\}$ meters. Agents move independently both with respect to one another and between time slots. At time $t = 0$, every node is

assumed to have perfect location knowledge, i.e., the a priori distribution $b\left(i^{(0)}\right)$ is a Dirac delta function $\forall i$. Agents do not know in which direction they move, but they do know the distance they travel, i.e., $\hat{d}^{(t)}_{i,\,\text{self}} = d^{(t)}_i$ and

$$p\left(i^{(t)}|i^{(t-1)}, \hat{d}^{(t)}_{i,\,\text{self}}\right) = \delta\left(\|i^{(t)} - i^{(t-1)}\| - d^{(t)}_i\right). \tag{5.2}$$

The distribution $p\left(\hat{d}^{(t)}_{ji} \,|i^{(t)}, j^{(t)}\right)$ is based on the experimentally derived ranging model described in Chapter 4.1. Messages are represented using samples: 500 samples for internode messages and 1000 samples for intranode messages. Finally, the BP algorithm was simulated with only $N_{\text{iter}} = 2$ whereas CLS was simulated with $N_{\text{iter}} = 50$.

In Fig. 5-2 we show the outage probability at $t = 20$ for both the CLS and BP. Observe that in general, the lack of infrastructure allows fewer nodes to localize in the CLS setting. CLS is comparable to BP in this case because CLS is highly sensitive to initial location estimates, and when nodes move slowly, these are fairly accurate. Hence when $\sigma_{\text{mob}} = 1$, BP provides only marginal improvement compared to CLS. However when nodes are highly mobile ($\sigma_{\text{mob}} = 10$) then initial location estimates at each time step may be poor and BP provides considerable improvements in terms of outage probability. For example, with an allowable error of 2m and when $\sigma_{\text{mob}} = 10$, CLS provides outage in approximately 25% of cases, whereas the BP-based algorithm provides outage in only 5% of cases (i.e. in this scenario BP provides a factor of 5 improvement in terms of outage probability).

5.2 Degrees of Cooperation

This section proposes a set of heuristics that reduce the computational complexity of BP-based algorithms by systematically reducing the quantity of messages passed in the network. Equivalently, these heuristics systematically reduce the degree of inter-node cooperation. When a set of network nodes pass messages cooperatively in order to localize, it is intuitive that not all messages are equally important. Additionally, if each message is made up of a high number of samples, based on techniques described in Chapter 4, then passing a high

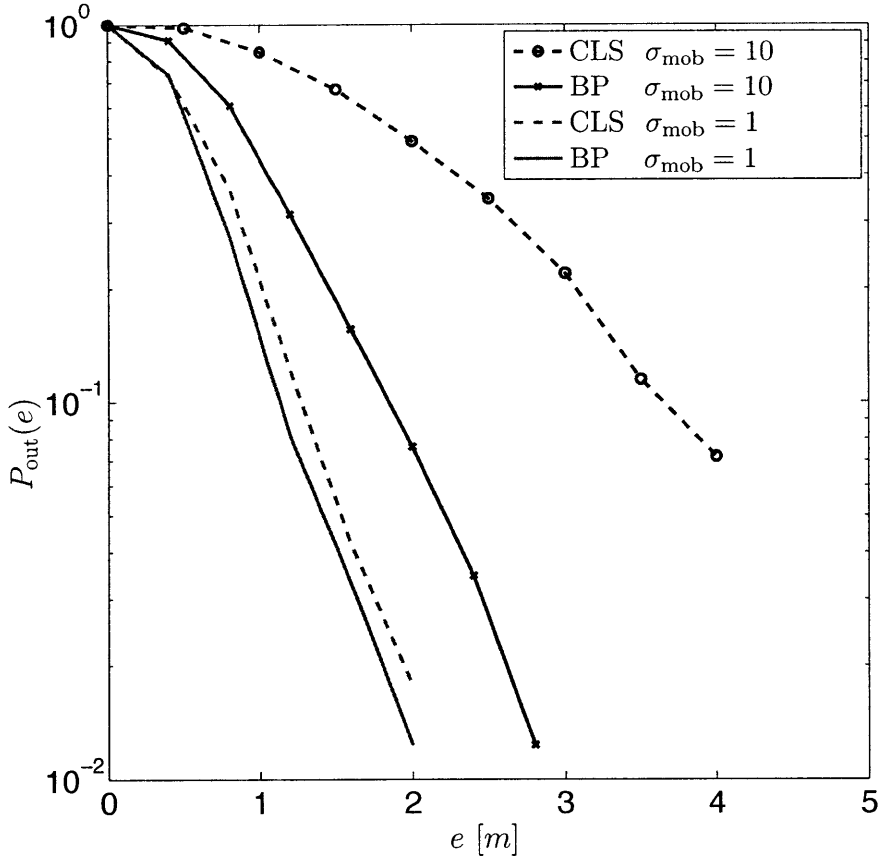


Figure 5-2: Comparison of outage probability at $t = 20$ for BP and CLS self-tracking in a network with 100 mobile agents, with $\sigma_{\text{mob}} = 1$ and $\sigma_{\text{mob}} = 10$. For CLS $N_{\text{iter}} = 50$ whereas for BP $N_{\text{iter}} = 2$. Results are averaged over 30 randomly deployed networks.

number of messages may have prohibitive computational complexity. For each network node, the algorithm complexity scales linearly in the number of neighbors. On the other hand, information from neighbors may not always be useful: (i) when the receiving node's belief is already very informative (e.g., tight around the mean); or (ii) when the transmitting node's belief is very uninformative.

To understand the degree of cooperation that is most beneficial, we will consider different levels of cooperation. Let us first introduce the following terminology: a distribution is said to be "sufficiently informative" when 95% of probability mass is located within 2m of the mean; a node becomes a *virtual anchor* when its belief is sufficiently informative; a *virtual*

bi-anchor is a node with a bimodal belief, with each mode being sufficiently informative; a node that is neither a virtual anchor nor a virtual bi-anchor will be called a *blind agent.* We are now ready to introduce four levels of cooperation at every iteration:

- **Level 1 (L1):** Virtual anchors broadcast their beliefs, while all other nodes censor their beliefs. Virtual anchors don't update their beliefs;
- **Level 2 (L2):** Virtual anchors and virtual bi-anchors broadcast their beliefs, while blind nodes censor their beliefs. Virtual anchors don't update their beliefs;
- **Level 3 (L3):** All nodes broadcast their beliefs. Virtual anchors don't update their beliefs; and
- **Level 4 (L4):** All nodes broadcast their beliefs. All nodes update their beliefs.

In terms of cooperation, note that L4 is more cooperative than L3, L3 is more cooperative than L2, and L2 is more cooperative than L1. In this sense, the different levels of cooperation are strict subsets. Our results will demonstrate how localization performance varies the level of cooperation (L1, L2, L3, or L4).

All simulations were performed in a 100m $\times$ 100m homogeneous environment, with 100 uniformly distributed agents that do not know their position and 13 fixed anchors with known positions in a grid configuration. Every node can range to other nodes within 20 meters. To decouple the effect of mobility with the message representation, we consider a single time slot, where every agent has a uniform a priori distribution over the environment $\mathcal{E}$. The algorithm was run for $N_{\text{it}} = 20$ iterations, though convergence was generally achieved well before 10 iterations. We also quantify the localization performance using outage probability. To estimate outage probability, we consider 50 random network topologies.

First of all, to establish comparison, we consider the computation time and outage performance of the BP-based algorithm as a function of the number of samples S, and the level of cooperation. Fig. 5-3 shows the comparative running time (all running times have been normalized for comparison purposes) averaged over 20 iterations and 50 random networks.

The figure demonstrates that as S increases, the running time of the algorithm significantly increases. In particular, when L4 is used with $S = 4096$, the running time per network is over 100 times that of the L2 parametric version.

Fig. 5-4 shows the comparative outage probability at $e = 1$m after 10 iterations. L1, L2, and L4 cooperation are not very sensitive to S, and generally outperform L3, especially when S is small. This is because the more complex distributions that are broadcast by L3, as compared to L2, cannot be well represented by few samples. In L4, this effect is offset by the fact that nodes keep updating their beliefs at every iteration, thus avoiding early convergence. We will provide intuitive reasoning as to why L3 is outperformed by L1 and L2 later in this section. In the remainder of this chapter we set $S = 2048$.

Fig. 5-5 shows the effect of the level of cooperation on the outage probability using the BP-based algorithm after $N_{\text{it}} = 20$ iterations. In general, L4 has the best performance in terms of accuracy and outage probability floor. Intuitively, one might expect L3 to have the next best performance, followed by L2, and then L1. However, Fig. 5-5 demonstrates that L3 has poorer accuracy than L2 and a similar outage floor. This effect can be explained as follows. The outage probability curve is a function of the number of agents in each network that have become virtual anchors, and the accuracy of each virtual anchor's belief. The fraction of agents that do not become virtual anchors within $N_{\text{it}} = 20$ is 1.7% for L1, and 0.3% for both L2 and L3.[2] This creates an outage probability floor, as those agents tend to have large localization errors. Since L2 and L3 have a similar fraction of agents that do not become virtual anchors, they have similar outage floors. The accuracy of beliefs belonging to agents that have become virtual anchors turns out to be highest for L1, followed by L2, and then L3. This is because L3 uses less reliable information than L2, which in turn is less reliable than L1. The final performance depends both on the fraction of virtual anchors (lowest for L1), and the accuracy of those virtual anchors (highest for L1). Note that we cannot compare L4 to L3 in this context, since there is no concept of a virtual anchor in L4.

In Fig. 5-6 we evaluate the outage probability for the BP algorithm as a function of the

[2]Note that for L4 there is no concept of virtual anchor.

allowable error and the iteration index, respectively, for different levels of cooperation. L3 is outperformed by L2 and L4 (as we expect from Fig. 5-4), and even by L1 (for $e > 2\,\mathrm{m}$).

In summary, our results show that more cooperation leads to faster improvement in terms of accuracy. The lowest level of cooperation, L1, is consistently slower to converge and achieves less accuracy. However, higher levels of cooperation also require the computation and representation of more complicated distributions. As a possible consequence, convergence issues may occur for L3 and L4 cooperation.

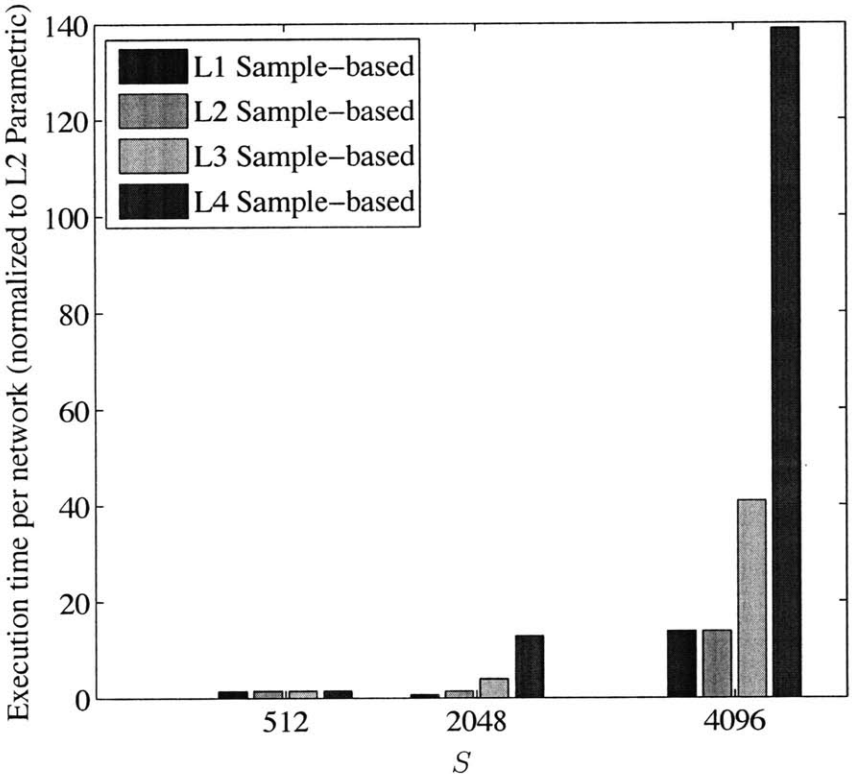


Figure 5-3: The effect of the number of samples S and the degree of cooperation on the running time per network in the BP-based algorithm. All running times are normalized for comparison purposes.

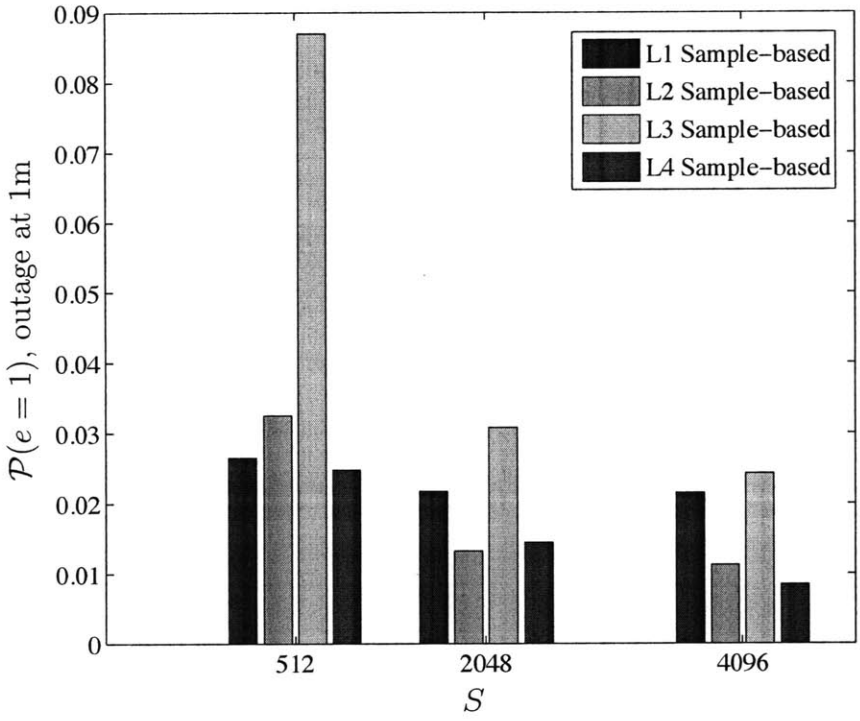


Figure 5-4: The effect of S and the level of cooperation in the centralized algorithm, on the outage probability at $e = 1$ meter. All results are after $N_{\text{it}} = 10$ iterations.

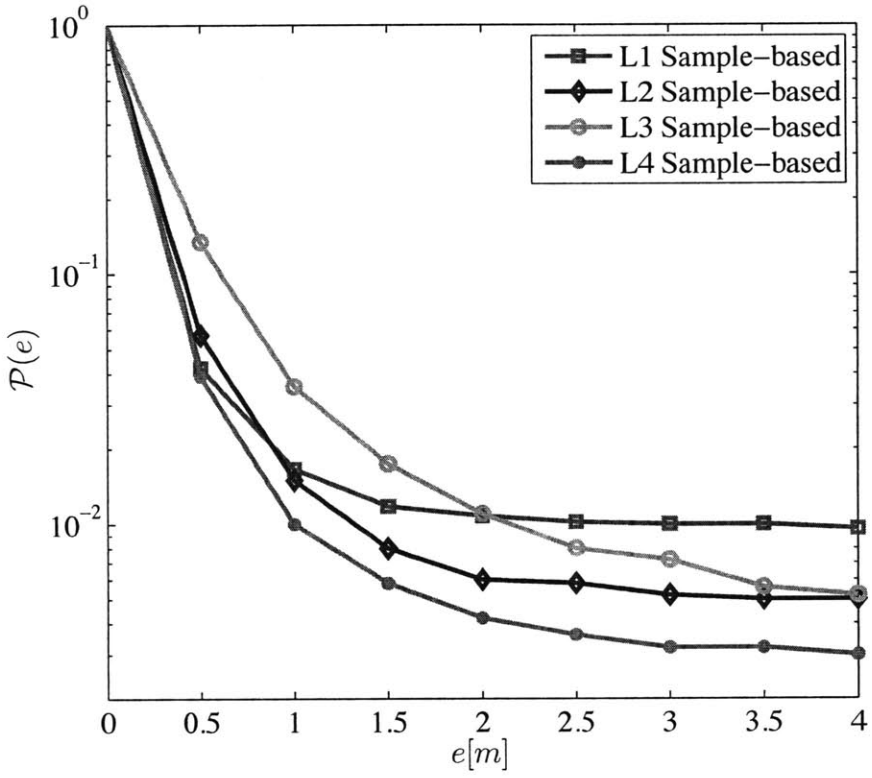


Figure 5-5: Effect of the level of cooperation on outage probability for the distributed algorithm at $N_{it} = 20$. Results are averaged over 50 networks.

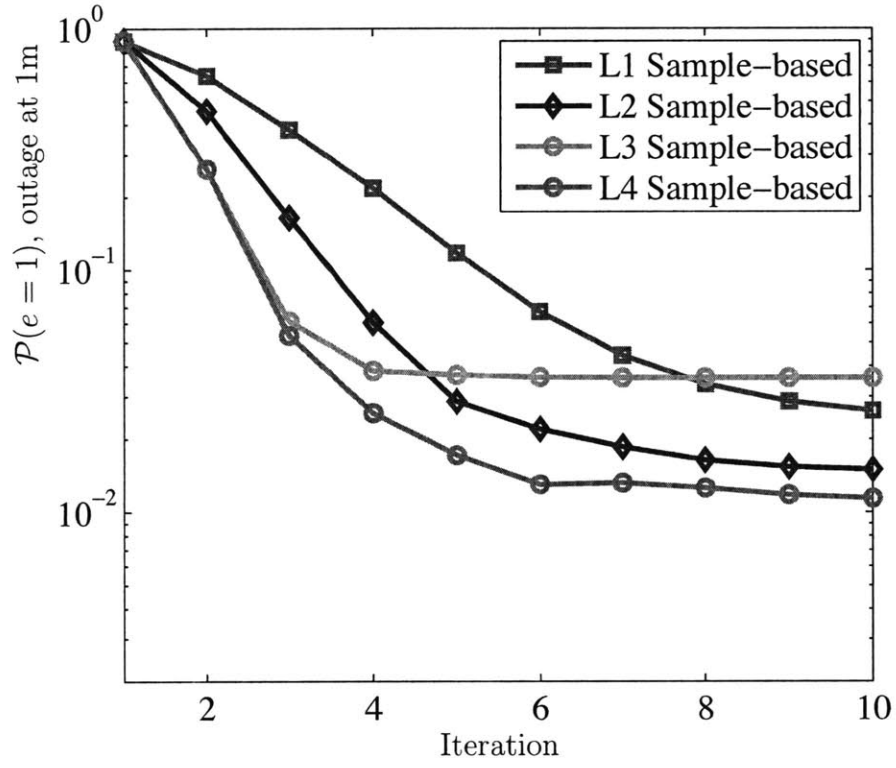


Figure 5-6: Effect of the level of cooperation on outage probability at $e = 1$, as a function of iteration. Results are averaged over 50 networks.

Chapter 6

Conclusions and Future Research

Network region graphs (NRGs) and GBP algorithms hold promise as a solution to self-inference problems. This paper introduced NRGs which allow GBP algorithms to be distributed across physical networks. A physical "backbone" or subgraph of any connected network graph is formed across which the computation for GBP is distributed. Clustered region graphs were developed as a method of generating RGs from network graphs. It was shown that all clustered region graphs that form trees and single cycles are maxent-normal. We then developed a distributed algorithm that generates a clustered region graph and passes messages for GBP. As a case-study, a method for representing GBP messages non-parametrically for location-aware networks was derived. Finally, as a special case, we quantified the performance of BP algorithms in mobile networks, as well as in static networks with varying degrees of cooperation.

Future work includes completing the performance quantification of the presented network region graph algorithms in networks with multiple hundred nodes. The maxent-normality of our generated region graphs ensures superior performance when compared to BP, but there may exist other region graph structural forms that can guarantee maxent-normality. The identification of such structures could lead to the design of new NRG algorithms. Finally, the ability to generalize the maxent-normal property may simplify the design of both GBP as well as BP algorithms.

Appendix A

Proofs of Chapter 3 Lemmas

Proof of Lemma 1. Let $i, j \in V(\mathcal{G})$ be unmarked nodes where $(i,j) \in E(\mathcal{G})$. Let $\mathcal{R}$ be an MN C^1-RG, and $r_1, r_2 \in \mathcal{R}$ include either b_i or b_j. Due to the inter-cluster property, the smallest common zone included within both r_1 and r_2 must include $\{b_i, b_j, \phi_{ij}, \phi_{ji}\}$. Thus every child region that has both r_1 and r_2 as parents also contains this common zone.

Removing (i,j) from $E(\mathcal{G})$, only removes ϕ_{ij} and ϕ_{ji} from the common zone. This implies that $|V(\mathcal{R})|$ and $E(\mathcal{R})$ remain invariant through this edge removal, and thus $\{c_r\}$ also remains invariant. Therefore, a modified C^1-RG $\mathcal{R}'$, generated from a modified $\mathcal{G}$ by removing (i,j), is also MN.

Assume an MN C^1-RG $\mathcal{R}$ is generated with $(i,j) \notin E(\mathcal{G})$. Suppose we add $(i,j) \in E(\mathcal{G})$ without violating the inter-cluster property, and then generate a new C^1-RG $\mathcal{R}'$ from this newly modified $\mathcal{G}$. Comparing $\mathcal{R}$ with $\mathcal{R}'$, we see that $V(\mathcal{R}) = V(\mathcal{R}')$ and $E(\mathcal{R}) = E(\mathcal{R}')$, thus $\{c_r\}$ also remains invariant though this edge addition. Therefore, a modified C^1-RG $\mathcal{R}'$ is also MN. □

Proof of Lemma 2. We first consider the addition of any number of unmarked nodes to the CDG such that the connectivity of each of these new nodes is one. Note that each of these unmarked nodes is singly-connected to only one DN. Factors and variables associated with this single edge will be added only to one $r \in \mathcal{R}_0$. The remaining regions and edges in $\mathcal{R}$ will not be affected by these additions, and thus the MN property will not be affected.

Suppose we add an unmarked node i with connectivity greater than one to the CDG, without violating the inter-cluster property. Without loss of generality, we only need to analyze the connections to DNs since, by Lemma 1, connections to other unmarked nodes will not affect the MN property. By the inter-cluster property, all these neighboring DNs must form a clique. Only the labels in $\mathcal{R}$ will be modified by the addition of this i, so the modified C^1-RG remains MN. □

Proof of Lemma 3. An example of a CDGs that forms a chain of length k and its associated C^1-RG is shown in Fig. A-1(a). The region-based entropy for the C^1-RG Fig. A-1(b) can be written as

$$\begin{aligned} H_{\mathcal{R}}(\{b_r\}) &= \sum_{r\in\mathcal{R}} c_r H_r(b_r) \\ &= H(1,2,\cdot,\cdot) + H(1,2,3,\cdot,\cdot) + H(2,3,4,\cdot,\cdot) + \cdots + H(k,k-1,\cdot,\cdot) \\ &\quad - H(1,2) - H(2,3) - \cdots - H(k-1,k) \end{aligned} \tag{A.1}$$

The mutual information between random variables is given by [50]

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{A.2}$$

so using this we can rewrite the $H_r(b_r)$ for each $r \in \mathcal{R}_0$ in terms of mutual information. For example, we simplify ((A.1)) by grouping the variables $[1,2,\cdot,\cdot]$ into $X = [1,2]$, $Y = [1,2,\cdot,\cdot,\backslash 1,2]$, and write

$$I(1,2,\cdot,\cdot) = H(1,2) + H(1,2,\cdot,\cdot\backslash 1,2) - H(1,2,\cdot,\cdot)\,. \tag{A.3}$$

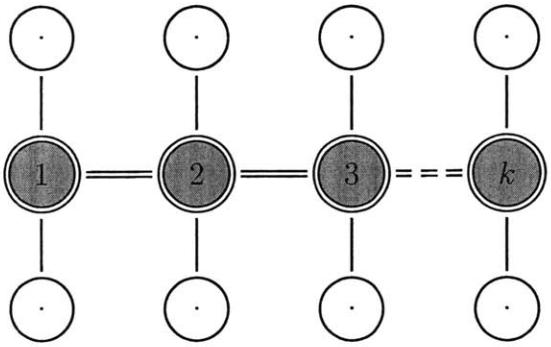
Writing the mutual information in the same fashion for each region $r \in \mathcal{R}_0$, where each r has counting number $c_r = 1$, and then substituting these terms into (A.1) gives

$$H_{\mathcal{R}}(\{b_r\}) = -I(1,2,\cdot,\cdot) - I(2,3,4,\cdot,\cdot) - \cdots - I(k,k-1,\cdot,\cdot)$$

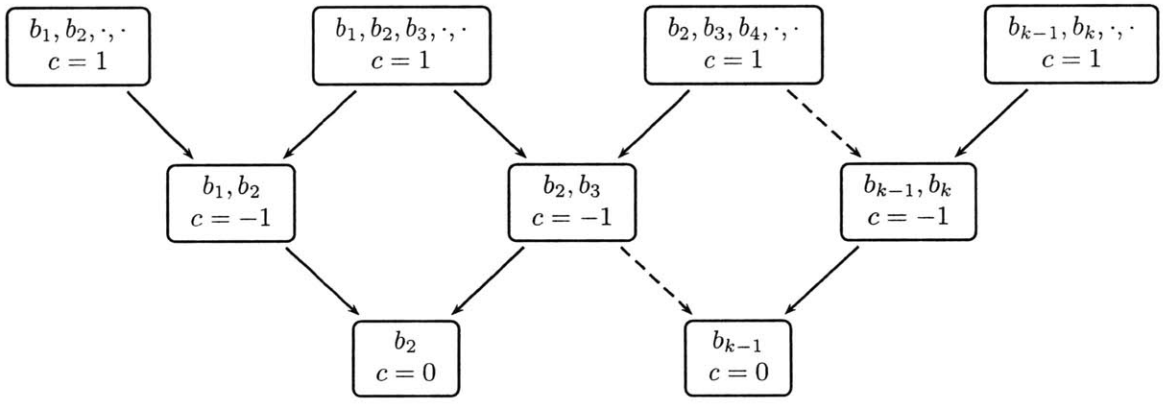
$$+ H(1,2,\cdot,\cdot\backslash 1,2) + H(1,2,3,\cdot,\cdot\backslash 2,3) + \cdots + H(k,k-1,\cdot,\cdot\backslash k,k-1)\,. \quad \text{(A.4)}$$

Note that in general the entropy of $H(X_j)$ is maximized when the distribution $b(x_j)$ is uniform. Additionally, the mutual information $I(X_j)$ is minimized when $b(x_j)$ is uniform [51]. Since $H_{\mathcal{R}}$ is a linear combination of negative mutual information terms and positive entropies, each of which are maximized when their beliefs are uniform, $H_{\mathcal{R}}$ is maxent normal.

□



(a) A small network graph, in which dominating nodes are shaded. The subgraph $\mathcal{D}$ forms a chain of length k. Unmarked nodes are unlabeled.



(b) The C^1-RG generated from the the network-graph depicted in Fig. A-1(a). Dotted arrows denote breaks in the graph.

Figure A-1: A network graph that forms a chain of length k and its associated C^1-RG. For the sack of discussion, when analyzing chain structures, when suppress the ϕ_{ij} functions in our RG as they are not relevant to our analysis.

Bibliography

[1] H. Wymeersch, U. Ferner, and M. Z. Win, "Cooperative Bayesian self-tracking for wireless networks," *IEEE Commun. Lett.*, vol. 12, no. 7, pp. 505–507, Jul. 2008.

[2] C. Chang and A. Sahai, "Estimation bounds for localization," in *in Proc. IEEE SECON*, 2004, pp. 415–424.

[3] M. I. Jordan, "Graphical models," *Stat. Sci.*, vol. 19, no. 1, pp. 140–155, 2004.

[4] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999. [Online]. Available: citeseer.ist.psu.edu/jordan98introduction.html

[5] J. D. G. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[6] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[7] R. E. Kalman, "A new approach for linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, pp. 34–45, 1960.

[8] R. G. Gallager, *Low-Density Parity Check Codes.* Cambridge, MA: MIT Press, 1963.

[9] M. Wainwright, T. Jaakkola, and A. Willsky, "Tree-based reparameterization framework for analysis of belief propagation and related algorithms," *IEEE Trans. Inf. Theory*, vol. 49, no. 5, pp. 1120–1146, May 2003.

[10] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[11] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[12] A. L. Yuille, "A double-loop algorithm to minimize the bethe free energy," in *Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition.* London, UK: Springer-Verlag, 2001, pp. 3–18.

[13] D. Mirza and C. Schurgers, "Motion-aware self-localization for underwater networks," in *Proc. Intl. Workshop. Wireless Net. Testbeds, Exp. Eval.* New York, NY, USA: ACM, 2008, pp. 51–58.

[14] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, no. 10, pp. 2173–2200, 2001.

[15] P. Pakzad and V. Anantharam, "Belief propagation and statistical physics," in *Conference on Information Sciences and Systems*, Princeton University, Mar. 2002.

[16] M. Castillo-Effen, W. A. Moreno, M. A. Labrador, and K. Valavanis, "Adapting sequential monte-carlo estimation to cooperative localization in wireless sensor networks," in *Proc. IEEE MASS*, Vancouver, Canada, oct 2006, pp. 656–661.

[17] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, Apr. 2005.

[18] U. Ferner, H. Wymeersch, and M. Z. Win, "Cooperative anchor-less localization for large dynamic networks," in *Proc. of IEEE Int. Conf. on Ultra-Wideband (ICUWB)*, vol. 2, Hannover, GERMANY, Sep. 2008, pp. 181–185.

[19] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009, special issue on *Ultra-Wide Bandwidth (UWB) Technology & Emerging Applications.*

[20] W. Suwansantisuk and M. Z. Win, "Multipath aided rapid acquisition: Optimal search strategies," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 174–193, Jan. 2007.

[21] A. Bletsas, H. Shin, and M. Z. Win, "Cooperative communications with outage-optimal opportunistic relaying," *IEEE Trans. Wireless Commun.*, vol. 6, no. 9, pp. 3450–3460, Sep. 2007.

[22] P. Berlin and D. Tuninetti, "LDPC codes for fading gaussian broadcast channels," *IEEE Trans. Inf. Theory*, vol. 51, pp. 2173–2182, Jun. 2005.

[23] D. Dardari, A. Conti, U. J. Ferner, A. Giorgetti, and M. Z. Win, "Ranging with ultrawide bandwidth signals in multipath environments," *Proc. IEEE*, vol. 97, no. 2, pp. 404–426, Feb. 2009, special issue on *Ultra-Wide Bandwidth (UWB) Technology & Emerging Applications.*

[24] M. Z. Win and R. A. Scholtz, "On the robustness of ultra-wide bandwidth signals in dense multipath environments," *IEEE Commun. Lett.*, vol. 2, no. 2, pp. 51–53, Feb. 1998.

[25] ——, "Characterization of ultra-wide bandwidth wireless indoor communications channel: A communication theoretic view," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 9, pp. 1613–1627, Dec. 2002.

[26] B. J. Frey and N. Jojic, "A comparison of algorithms for inference and learning in probabilistic graphical models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 9, pp. 1392–1416, Sep. 2005.

[27] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, 2004.

[28] M. Welling, T. Minka, and Y. W. Teh, "Structured region graphs: Morphing EP into GBP," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, vol. 21, 2005.

[29] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," MERL, Cambridge, MA, Tech. Rep. TR-2000-26, Jun. 2000, considers, briefly, a comparison between bp and gbp in terms of accuracy.

[30] D. Baker and A. Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Trans. Commun.*, vol. 29, no. 11, pp. 1694–1701, Nov. 1981.

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[32] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 10, pp. 908–920, Oct. 2004.

[33] H. Wymeersch, *Iterative Receiver Design.* Cambridge University Press, 2007.

[34] Z. I. Botev, "Nonparametric density estimation via diffusion mixing," Department of Mathematics, The University of Queensland, postgraduate series, 2007, nov.

[35] R. Kikuchi, "A theory of cooperative phenomena," *Phys. Rev.*, vol. 81, no. 6, pp. 988–1003, 1951.

[36] T. Morita, "Cluster variation method for nonuniform ising and heisenberg models and spin-pair correlation function," *Progr. Theor. Phys.*, vol. 85, no. 2, pp. 243–255, 1991.

[37] T. Morita, M. Suzuki, K. Wada, and M. Kaburagi, "Foundations and applications of culster variation method and path probability method," *Progr. Theor. Phys. Suppl.*, vol. 115, 1994.

[38] R. Diestel, *Graph theory.* Springer, 2006.

[39] Y. Shen and M. Z. Win, "Fundamental limits of wideband localization – part I: A general framework," *IEEE Trans. Inf. Theory*, vol. 55, 2009, to appear.

[40] M. Welling, "On the choice of regions for generalized belief propagation," in *Proc. UAI.* Arlington, Virginia: AUAI Press, 2004, pp. 585–593.

[41] J. Lien, U. J. Ferner, W. Srichavengsup, H. Wymeersch, and M. Z. Win, "Practical cooperative localization and tracking," *IEEE Trans. Wireless Commun., Submitted* 2010, submitted for publication.

[42] J. Lien, "*A Framework for Cooperative Localization in Ultra-Wideband Wireless Networks*," Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, Jun. 2007, thesis advisor: Professor Moe Z. Win.

[43] H. Wymeersch, *Iterative Receiver Design.* Cambridge University Press, 2007.

[44] R. Lidl and H. Niederreiter, *Finite Fields.* Addison Wesley, 1983.

[45] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Telecommunications (ETT)*, vol. 8, no. 2, pp. 119–125, Mar. 1997.

[46] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. MobiCom.* Rome, Italy: ACM, 2001, pp. 166–179.

[47] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *Proc. of the general track: USENIX technical conf.*, Berkeley, CA, 2002, pp. 317–327.

[48] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. WSNA.* Atlanta, Georgia: ACM Press, sep 2002.

[49] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[50] S. Watanabe, "Information theoretical analysis of multivariate correlation," *IBM Journal of Research and Development*, vol. 4, pp. 66–82, 1960.

[51] T. A. Cover and J. A. Thomas, *Elements of Information Theory*, 1st ed. New York, NY, 10158: John Wiley & Sons, Inc., 1991.