IMPROVING THE SPEED AND ACCURACY OF INDOOR LOCALIZATION

BY KONSTANTINOS KLEISOURIS

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Richard P. Martin

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2009

ABSTRACT OF THE DISSERTATION

Improving the Speed and Accuracy of Indoor Localization

by Konstantinos Kleisouris

Dissertation Director: Richard P. Martin

Advances in technology have enabled a large number of computing devices to communicate wirelessly. In addition, radio waves, which are the primary means of transmitting data in wireless communication, can be used to localize devices in the 2D and 3D space. As a result there has been an increasing number of applications that rely on the availability of device location. Many systems have been developed to provide location estimates indoors, where Global Positioning System (GPS) devices do not work. However, localization indoors faces many challenges. First, a localization system should use as little extra hardware as possible, should work on any wireless device with very little or no modification, and localization latency should be small. Also, wireless signals indoors suffer from environmental effects like reflection, diffraction and scattering, making signal characterization with respect to location difficult. Moreover, many algorithms require detailed profiling of the environment, making the systems hard to deploy.

This thesis addresses some of the aforementioned issues for localization systems that rely on radio properties like Received Signal Strength (RSS). The advantage of these systems is that they reuse the existing communication infrastructure, rather than necessitating the deployment of specialized hardware. Specifically, we improved the latency of a particular localization method that relies on Bayesian Networks (BNs). This method has the advantage of requiring a small size of training data, can localize

many devices simultaneously, and some versions of BNs can localize without requiring the knowledge of the locations where signal strength properties are collected. We proposed Markov Chain Monte Carlo (MCMC) algorithms and evaluated their performance by introducing a metric which we call relative accuracy. We reduced latency by identifying MCMC methods that improve the relative accuracy to solutions returned by existing statistical packages in as little time as possible. In addition, we parallelized the MCMC process to improve latency when localizing devices whose number is on the order of hundreds. Finally, since wireless transmission is heavily affected by the physical environment indoors, we investigated the impact of using multiple antennas on the performance of various localization algorithms. We showed that deploying low-cost antennas at fixed locations can improve the accuracy and stability of localization algorithms indoors.

Acknowledgements

Foremost, I would like to thank my advisor, Professor Richard P. Martin. This thesis would not have been possible without his help. I hope to make him proud of my current and future endeavors.

I would also like to thank Kathleen Goelz for her invaluable support throughout my graduate studies. She encouraged me to keep going, one step at a time, constantly moving towards higher and greater goals.

I am also grateful to Professors Michael Littman, Ahmed Elgammal and Dr. Giovanni Vannucci for being on my committee and providing valuable comments and insight.

I would like to thank my family, to whom I owe the most. My parents, Aristotelis and Maria, and my sister Panagiota, all stayed close to me despite the thousands of miles separating us. They offered me encouragement and confidence during my studies.

Finally, I would like to express my appreciation to staff and other faculty and students of the Department of Computer Science at Rutgers University who helped me get through this experience.

Dedication

To my dear parents, Aristotelis and Maria Kleisouris.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent advances in technology have embedded wireless transceivers in many computing devices, such as laptops, personal digital assistants (PDAs), cellular phones. As a result people nowadays have the flexibility to connect to various networks, like Wireless Local Area Networks (Wireless LANs) and cellular networks, from many different places, like an office building, cafeteria, vehicle. Also, sensors deployed in different areas can measure environmental properties such as temperature, humidity, and transmit their readings wirelessly. Radio waves are the primary way of transmitting data in wireless communication. Undoubtedly, wireless technology has made communication much easier and convenient, since it moves away from the physical constraints of cables.

Recent years have seen tremendous efforts [9, 41, 43, 60, 65, 74] at building systems that reuse the existing wireless communication infrastructure to localize devices; that is to provide the coordinates of a device in the 2-dimensional (2D) or 3-dimensional (3D) space. This is a new capability, since traditionally networks have been used for communication. The ability to localize has become very important nowadays. Typical applications include: (a) tracking of equipment and personnel in factories and hospitals, (b) providing location-specific information in museums and libraries, (c) controlling access to information and utilities based on users' location, (d) monitoring and management of wireless networks, (e) localizing sensors used for environmental monitoring.

A lot of localization systems have focused on providing location estimates indoors, where Global Positioning System (GPS) [27] devices do not work. However, building such systems faces a lot of challenges. First of all, these systems should be general purpose, which means they should work on any wireless device with little/no modification, and at the same time they should leverage as much of the existing communication

infrastructure of a wireless network. This is very significant, since the less extra hardware needed the easier the deployment and use of the system, and also the smaller the cost. Second, the process of localization should be done really fast, so that higher level applications can track devices and people in real time. Third, a lot of these systems require extensive profiling of the buildings where they are deployed. The profiling might require detailed maps of a particular site (e.g. wall/floor material) and also collecting radio properties (like signal strength) at known locations, which is labor-intensive and time-consuming. Environmental changes necessitate recollection of such properties to maintain localization accuracy. At the same time radio signal propagation suffers from reflection, diffraction and scattering indoors, making harder to infer location estimates from its properties. Thus, a big challenge for localization systems is to minimize the information needed to adequately profile a site and also they should be robust to environmental impacts on radio properties.

We believe such challenges must be addressed in order to reach a point where any wireless device can "know where it is" and to better service higher level applications. This thesis thus tackles some of the aforementioned issues that we hope will make indoor localization a more tractable problem. Particularly, we first focus on improving the speed of providing location estimates of a specific method that uses Bayesian Networks (BNs) [25,51]. Unlike other approaches, BNs require smaller number of radio properties to be collected at some particular site, certain versions of them can localize without the need to know the locations where the properties are collected and at the same time they can localize many devices simultaneously. Since our BNs do not have closed-form solutions we implemented several Markov Chain Monte Carlo (MCMC) [47, 73] methods to provide location estimates. We evaluated the performance of an MCMC method by introducing a metric which we call *relative accuracy*. The metric estimates the Euclidean distance of the localization result returned by an MCMC method to the result returned by a well-tested statistical package called WinBugs [49] after a long run. Hence, in this work, we define the problem of reducing localization latency as identifying MCMC methods that improve the relative accuracy in as small amount of time as possible. Also, in order to minimize the localization latency when locating a

large number of devices (on the order of hundreds), we proposed schemes to parallelize the MCMC process, achieving good speedups.

Having improved the relative accuracy of BNs indoors, we tried to improve the absolute accuracy of different algorithms that use received signal strength (RSS) to localize. Absolute accuracy is the Euclidean distance between the result returned by an algorithm and the actual location of a mobile device. Since radio waves suffer indoors from environmental effects like reflection, diffraction and scattering, inevitably absolute accuracy is affected by them. Thus, we investigated the impact of using multiple antennas on the absolute accuracy of different localization algorithms. Conclusively, in this thesis we focused on improving relative and absolute accuracy for indoor localization.

In the remainder of this chapter, we first briefly introduce some basics of communication and localization in wireless networks in Section 1.1. We then provide a general description of the methods we proposed to improve localization speed and accuracy in Section 1.2, which defines an outline for the thesis. Finally, contributions of our work are summarized in Section 1.3.

1.1 Communication and Localization in Wireless Networks

In this section, we first describe environmental effects that radio waves suffer from indoors and also summarize a range of wireless technologies that concerns us. We then give a brief background on localization algorithms and the categories they can be divided into.

1.1.1 Wireless Communication

Wireless communication relies on radio signals to transmit data. Radio signals are electromagnetic waves, which are usually characterized by both wavelength and frequency. Radio signal propagation in space is generally affected by the environment in three ways: reflection, diffraction, and scattering [45,61]. *Reflection* refers to the bouncing of radio signals from objects with larger dimensions than the signal wavelength. It may occur on ground surfaces, buildings, and furniture. *Diffraction* refers to the bending

of radio waves around objects. It usually happens when the object's surface has sharp edges, for example, around buildings, hills, and trees. *Scattering* refers to the dispersion of radio waves due to collisions with objects of smaller dimensions than the signal wavelength. In practice, it may happen around foliage, street signs or stairs within buildings. Due to these complicated propagation mechanisms, the radio signals may reach the destination through many different paths, and the final received signal is a combination over all such traversals. This is commonly referred to as the *multipath* effect.

Three wireless communication standards that are most commonly used to form networks indoors or in a relatively small area are *Wi-Fi* [2], *Bluetooth* [3], and *ZigBee* [4]. *Wi-Fi* [2] networks function according to the IEEE 802.11 standards, and are mostly used to provide Internet access at home or in office buildings. When people refer to Wireless LANs, most of the time they refer to networks based on Wi-Fi technology. The normal infrastructure for a Wi-Fi network consists of one or more Access Points (APs) or landmarks, which have the ability to communicate over the wireless medium. Wi-Fi devices can thus connect to the Internet or talk to each other through the APs. Wi-Fi devices can also connect to each other directly. *Bluetooth* [3] refers to the IEEE 802.15.1 communication standard. It is designed for lower power consumption than Wi-Fi, and thus has a relatively shorter range (1, 10, or 100 meters). Hence, it is mostly used for communication between devices located close to one another. Currently many devices support Bluetooth, including cell phones, laptops, digital cameras, printers, mice, and headsets. *ZigBee* [4] refers to the IEEE 802.15.4 communication standard. The main target for the ZigBee protocol is embedded applications such as environmental monitoring, intruder detection and building automation. This standard is widely used for communication within *sensor networks*. Since ZigBee applications are mostly embedded, the corresponding devices are required to be small. The currently available ones have already shrunk to be comparable to the size of a quarter [1, 5].

Although in this thesis we focus on the Wi-Fi protocol, the proposed solutions and conclusions drawn here can be similarly extended to the other two (Bluetooth, ZigBee) wireless communication standards.

1.1.2 Background on Localization

Over the past few years, many localization algorithms have been proposed to localize wireless devices and sensors, and provide location information to new classes of location-oriented applications. In general, localization algorithms can be categorized as: range-based vs. range-free, scene matching (fingerprint matching), and aggregate or singular.

The range-based algorithms involve distance estimation to landmarks using the measurement of various physical properties like Received Signal Strength (RSS) [35], Time Of Arrival (TOA) [27] and Time Difference Of Arrival (TDOA) [59]. Rather than use precise physical property measurements, range-free algorithms use coarser metrics like connectivity [64] or hop-counts [55] to landmarks to place bounds on candidate positions.

In scene matching approaches, a radio map of the environment is constructed by measuring actual samples, or by using signal propagation models, or some combination of the two. A node then measures a set of radio properties (often just the RSS of a set of landmarks), the *fingerprint*, and attempts to match these to known location(s) on the radio map. These approaches are almost always used in indoor environments because signal propagation is extensively affected by reflection, diffraction and scattering, and thus ranging or simple distance bounds cannot be effectively employed. Matching fingerprints to locations can be cast in statistical terms [60, 74], as a machine-learning classifier problem [12], or as a clustering problem [9]. Figure 1.1 shows the setup on an office floor of a system that uses scene matching. A number of landmarks, which record signal strength readings s_i, have been deployed to assist in localization. In practice, the s_i are averaged over a sufficiently large time window to remove statistical variability.

Finally, a third dimension of classification extends to aggregate or singular algorithms. Aggregate approaches use collections of many nodes in the network in order to localize (often by flooding), while localization of a node in singular methods only requires it to communicate to a few landmarks. For example, algorithms using optimization [23] or multidimensional scaling [64] require many estimates between nodes.

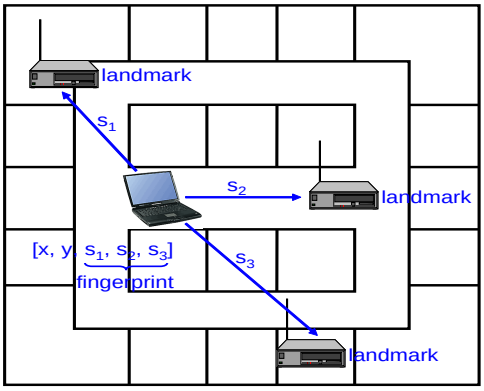


Figure 1.1. Setup of a localization system that uses landmarks (or Access Points) that record radio properties (e.g. signal strengths s_i) from a wireless device.

We can further break down localization algorithms into two main categories: point-based methods, and area-based methods. Point-based methods return an estimated point as a localization result. A primary example of a point-based method is the RADAR scheme [9]. On the other hand, area-based algorithms return a *most likely* area in which the true location resides. One of the major advantages of area-based compared to point-based methods is that they return a region, which has an increased chance of capturing the transmitter's true location. Examples of area-based algorithms are Area Based Probability (ABP) [26] and Bayesian Networks (BNs) [51].

Algorithms that use RSS as the basis of localization are very attractive options, because using RSS allows the localization system to reuse the existing communication infrastructure rather than requiring the additional cost needed to deploy specialized localization infrastructure, such as ceiling-based ultrasound, GPS, or infrared methods [33, 59, 62]. The wireless communication standards described in Section 1.1.1 (Wi-Fi, Bluetooth, ZigBee) provide RSS values associated with packet reception, and thus localization services can easily be built for such systems. Further, RSS-based localization is attractive as the techniques are technology-independent: an algorithm can be developed and applied across different platforms, whether 802.11 or Bluetooth. In addition, it provides reasonable accuracy with median errors of 1 to 5 meters [26]. Most fingerprinting approaches utilize the RSS, e.g. [9, 12], and many multilateration approaches [51] use it as well. In this thesis we thus focus on localization algorithms

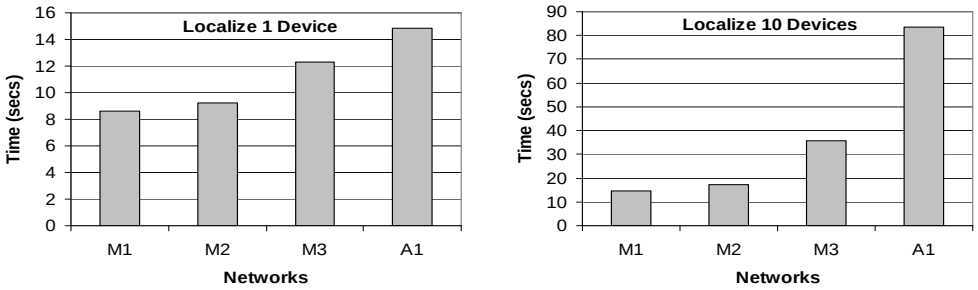


Figure 1.2. Execution time of WinBugs for localizing 1 device and 10 devices on a 2.4-GHz machine with Bayesian Networks M_1, M_2, M_3, A_1.

that employ signal strength measurements.

1.2 Thesis Structure

As we have already mentioned, there are many challenges in wireless localization. In this work, we try to minimize localization latency for a particular method and also alleviate the impact of environmental effects on radio signal strength, hoping that this will improve the localization performance of several algorithms.

In Chapter 2, we reduce the computational cost of four Bayesian Networks (BNs) [25,51], namely M_1, M_2, M_3, A_1, used for localization. These networks are graphs that represent the joint probability distribution of random variables (e.g. coordinates of a device to be localized). Inferring values for the unknowns can be done using commercial statistical packages, like WinBugs [49]. However, these packages are general-purpose solvers and, hence, incur a lot of computational cost when used. The cost increases drastically as the number of devices located simultaneously by the BNs gets large. For instance, Figure 1.2 depicts the time needed to localize 1 and 10 devices using the four BNs on a 2.4-GHz machine. We see that locating 1 device can take from 8 secs up to 15 secs, whereas locating 10 devices can take from 15 secs up to 83 secs. Clearly, this time is prohibitive for a localization system. Hence, in order for the BNs to be practical, it is imperative that they provide location estimates in as small amount of time as possible. Since the BNs under study do not have closed-form solutions, we resort to simulation methods. Specifically, we present a number of Markov Chain Monte Carlo

(MCMC) [47,73] algorithms that can solve these BNs in a smaller amount of time when compared to existing solvers. These algorithms rely on statistical sampling to explore the probability density function (PDFs) of the unknowns and build their histogram. We show that by taking advantage of the flatness of the PDFs of the unknowns of interest (e.g. coordinates of a device), we get an algorithm that has the best performance in terms of convergence to the solution provided by WinBugs. At the same time the algorithm, which we call "whole domain sampling", requires no tuning, which means it can be used as a black box for higher level applications. We also provide an analytic model that shows how flat a distribution should be so that "whole domain sampling" is more efficient than other methods.

In Chapter 3 we try to improve the localization latency when locating a large number of devices simultaneously. Although the MCMC methods proposed in Chapter 2 are computationally efficient, they still take a lot of time when localizing devices whose number is on the order of hundreds. Reducing the latency in this case is important, since, as technology advances, wireless networks will offer more benefits in the future, and hence a large number of devices will be connected to them which a system should be able to localize. Thus, in Chapter 3 we explore whether parallel computing methods can help us reduce latency in this case. Since MCMC methods generate a Markov chain, where every state of the chain corresponds to an instance of a BN with all random variables having values, we propose two schemes of parallelizing the MCMC process. The first applies inter-chain parallelism, by running multiple independent chains on different processors. The second, applies intra-chain parallelism, by dividing the formation of a single Markov state across processors. The two schemes were implemented in the Berkley Unified Parallel C (BUPC) [69] language and tested on different computing platforms. Our experimental results show that the inter-chain parallelism gives good speedups for long Markov chains, whereas the intra-chain can give good speedups for short Markov chains. Since providing good location estimates with our Bayesian Networks does not require long Markov chains, intra-chain parallelism is the scheme that can help up improve latency for localization. Also, we use the LogGP [20] model to analyze and predict the performance of the two schemes. We show that the model is

a useful tool in understanding whether the algorithms have been parallelized enough so that we get good speedups and whether there any pathological situations like load imbalance or contention.

In Chapter 4 we investigate the impact of using multiple antennas at fixed known locations on the localization performance of several algorithms that use different techniques, ranging from neighbor matching in signal space, to maximum likelihood estimation and to multilateration. These algorithms rely on the received signal strength (RSS) transmitted by a wireless device to localize it. However, indoors, the signal strength suffers from environmental effects, such as reflection, diffraction and scattering, making it hard to localize objects. Our strategy is to see first whether multiple antennas can average out environmental effects. We do so by showing that the RSS from multiple antennas can better fit a theoretical signal propagation model when compared to the RSS from a single antenna. Next, we investigate the impact of multiple antennas on the accuracy and stability of various localization algorithms. Accuracy refers to the Euclidean distance between the estimated and real location. Stability refers to how much an estimated location changes when there are small-scale movements of a wireless device around its position. Our results show that multiple antennas help improve accuracy and in some cases the improvement can be up to 70%. Similarly, we can achieve up to 100% improvement in stability over the single antenna case. Hence, localization systems can benefit from the deployment of low-cost antennas.

In summary, Chapter 2 presents MCMC methods that can solve Bayesian Networks (BNs) used for indoor localization with much smaller computational cost when compared to statistical packages like WinBugs. One of the methods, "whole domain sampling", is shown to have the best performance. Chapter 3 proposes two schemes to parallelize the MCMC process, and presents speedups for our BNs on different platforms. It also shows how the LogGP model can be used to understand and predict the performance of the two schemes. Chapter 4 shows that multiple antennas can reduce environmental effects in an indoor environment on the radio signal strength. It also presents the impact of multiple antennas on accuracy and stability on different localization algorithms. Finally, Chapter 5 concludes the thesis.

1.3 Contributions

Our contributions in this thesis include:

- We show that the probability distributions of random variables of interest (e.g. x and y coordinates) in Bayesian Networks used for localization are flat. This led us to implement an MCMC method, called "whole domain sampling", that is computationally fast and converges quickly to solutions provided by statistical packages like WinBugs. The method is shown to be at least 10 times faster than WinBugs and requires no tuning. We also present an analytic model that determines how flat a distribution should be so that "whole domain sampling" is faster than other methods.

- We propose two schemes to parallelize an MCMC method: (a) inter-chain algorithm, (b) intra-chain algorithm. The schemes were implemented in Berkeley UPC (BUPC) and tested on different computing platforms. The first algorithm gives good speedups for applications that need long Markov chains, whereas the second for applications that need short Markov chains. The intra-chain algorithm can give a speedup of 12 on 16 processors for our Bayesian Networks when localizing 200 devices simultaneously. We found BUPC an effective tool in describing the data layout needed by the two schemes. We use the LogGP model of parallel computation to understand and predict the performance of the two algorithms on different platforms.

- We show that multiple antennas can average out environmental effects on received signal strength (RSS) indoors. We do so by demonstrating that RSS from multiple antennas better fits a theoretical signal strength propagation model. We also show that multiple antennas can improve localization accuracy and stability of several algorithms.

Chapter 2

Reducing the Computational Cost of Bayesian Indoor Positioning Systems

2.1 Introduction

There have been a lot of small- and medium-scale localization systems [33, 48, 55, 59, 62, 70] for 802.11, sensor networks, custom radios, and ones that use ultrasound or infrared. In this chapter we focus on reducing the computational cost of a specific approach that uses Bayesian networks [25, 26, 51] for indoor location estimation in wireless networks. Bayesian networks can be used in a Wi-Fi (IEEE 802.11) setup to track wireless devices such as laptop computers, handheld devices, and electronic badges inside stores, hospitals and factories. The networks can also incorporate several features of the medium, such as received signal strength (RSS) and angle of arrival of the signal (AoA), to provide location estimates.

Although Bayesian networks are attractive compared to other approaches because they provide similar performance with much less training data, the computational cost of using these networks with standard statistical packages, such as WinBugs [49], is quite large as we saw in Section 1.2. Figure 1.2 shows that localizing a few points can take up to 10 seconds on a well-equipped machine. In addition, stock solvers do not scale well when localizing points with no location information in the training data; in this case localization can take well over a minute.

We are thus motivated to identify methods of solving Bayesian networks used for indoor localization that are computationally efficient and simultaneously provide quick convergence to the solution. Finding such methods not only tells us how fast we can localize, but also what results we should expect when compared to "gold standard"

solutions provided by packages like WinBugs.

Our Bayesian networks have no closed-form solutions and, thus, we turn to Markov Chain Monte Carlo (MCMC) simulation to solve these networks. This family of approaches uses statistical sampling to explore the probability density functions (PDFs) of the variables in the network. Specifically, the MCMC methods we use are Gibbs sampling and Metropolis-within-Gibbs sampling. Within these variants, there is a large diversity of approaches to sampling individual variables. Thus, in this chapter we investigate the tradeoffs of these techniques for localization.

We found that slice sampling is the method that dominates the entire execution time in the Gibbs approach as we try to localize many points simultaneously. Specifically, the number of evaluations of the full conditional is the prevailing factor that makes slice sampling computationally expensive. Second, using real data, we found that the full conditionals of the coordinates of an item we try to localize as well as the angle of the received signal strength are relatively flat.

The flatness property led us to implement a variation of slice sampling that we call *whole domain sampling*. Our method samples uniformly over the whole domain of a variable, as opposed to carefully choosing only parts of the domain to sample from. We found whole domain sampling is computationally fast and simultaneously mixes rapidly, and thus provides fast convergence. Such a method requires no tuning, making it an attractive approach since it constitutes a "black-box" sampler for our networks. For other methods, such as Metropolis, tuning is critical to get reasonable results. We also found the flatness of the full conditionals to be the key factor in determining the effectiveness of our whole domain approach.

We show that whole domain sampling can localize 1 or 10 points to within 1ft of the solution provided by WinBugs in less than half a second. Moreover, the execution time of the method is 9 to 17 times faster than the standard WinBugs solver, depending on the type of Bayesian network used and the size of the training set. Additionally, the method scales well, localizing simultaneously 51 points with no location information in the training set in 6 seconds.

In order to better understand why whole domain sampling converges faster than

other methods, we built an analytic model that estimates the number of evaluations of the full conditional under slice sampling when using: (a) a whole domain approach, and (b) a step out process. Our model can analytically determine how flat a double exponential distribution should be in order for whole domain sampling to be faster than a step out approach. Comparing the shape of this PDF to the actual PDFs in our Bayesian networks shows qualitatively that these curves clearly fall in the regime where whole domain sampling is desirable.

The rest of this chapter is organized as follows. In Section 2.2 we give a brief background on Bayesian networks and in Section 2.3 we describe how some MCMC methods work. In Section 2.4 we describe the Bayesian models used for localization, while in Section 2.5 we evaluate our MCMC samplers with respect to computational cost and accuracy vs. time. Section 2.6 presents our analytic model. In Section 2.7 we present a Monte Carlo (MC) method, Importance Sampling, and compare it to whole domain sampling. Section 2.8 gives related work, and in Section 2.9 we summarize our work.

2.2 Background

A *graphical model* is a multivariate statistical model embodying a set of conditional independence relationships. Here, we focus on acyclic digraphs (ADGs). The edges in the graph encode the relationships. Each vertex corresponds to a random variable X_v, $v \in V$, taking values in a sample space $\mathcal{X}_v$. To simplify notation, we use v in place of X_v in what follows. In an ADG, the *parents* of a vertex v, $\text{pa}(v)$, are those vertices from which edges point into v. The *descendants* of a vertex v are the vertices which are reachable from v along a directed path. A vertex w is a *child* of v if there is an edge from v to w. The parents of v are taken to be the only direct influences on v, so that v is independent of its non-descendants given its parents. This property implies a factorization of the joint density of all v, which we denote by $p(V)$, given by

$$p(V) = \prod_{v \in V} p(v|\text{pa}(v)) \tag{2.1}$$

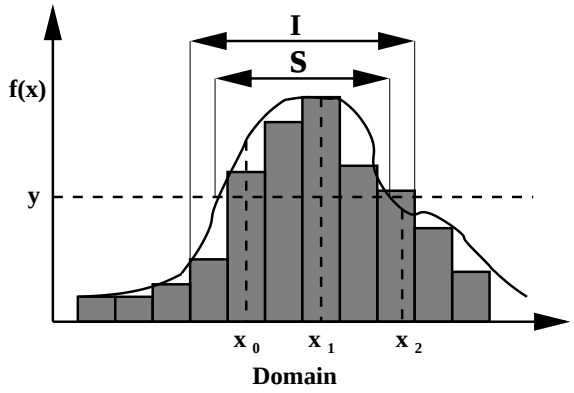


Figure 2.1. Constructs used for MCMC sampling

In the Bayesian framework, model parameters are random variables and, hence, appear as vertices in the graph. When some variables are discrete and others continuous, or when some of the variables are latent or have missing values, a closed-form Bayesian solution generally does not exist. Analysis then requires either analytic approximations of some kind or simulation methods. One such simulation method is the Monte Carlo method that has been used to compute the integral of some function $f(x)$ over some region D, by drawing independent and identically distributed (i.i.d.) random samples uniformly from D. Figure 2.1 provides some intuition in this process. The curve represents the unknown PDF of a variable (e.g. the x-coordinate of an object to be localized). Monte Carlo sampling methods approximate the PDF by building a histogram using randomized draws. If the draws are generated by evolving a Markov chain, they are no longer independent, and the process is called Markov Chain Monte Carlo (MCMC).

2.3 Markov Chain Monte Carlo

An MCMC method starts with some initial value for each stochastic variable v (e.g. x-coordinate), and then cycles through the graph replacing the old value of each v with a new value. The new value is drawn from some distribution that depends on the MCMC method used. After sufficient iterations of the procedure one assumes the Markov chain has reached its stationary distribution. Future simulated values are then monitored. The monitoring process may record the entire histogram, or only measure

the median, mean, or the 95% interval.

Once a Markov chain has reached its stationary distribution, a delicate issue is whether the chain moves fast around the space of the PDF of a stochastic variable. If it does, then we say the chain "mixes" rapidly. Intuitively, in Figure 2.1, mixing describes how much of the domain is explored as a function of time.

Below we give a brief overview of two MCMC methods that can be used for Bayesian inference. More details and other methods can be found in [47, 53, 54, 67, 73].

2.3.1 Gibbs Sampling

A single-variable or univariate (updates one variable at a time) Gibbs sampler chooses the new value of a stochastic variable v from its conditional probability distribution, given all the other quantities, denoted $V \backslash v$, are fixed at their current values (known as the "full conditional"). The crucial connection between directed graphical models and Gibbs sampling lies in expression (2.1). The full conditional distribution for any vertex v is equal to:

$$\begin{aligned} p(v|V\backslash v) &\propto p(v, V\backslash v) && (2.2)\\ &\propto \text{terms in } p(V) \text{ containing } v && (2.3)\\ &= p(v|\text{pa}(v)) \prod_{w \in \text{child}(v)} p(w|\text{pa}(w)) && (2.4) \end{aligned}$$

i.e., a prior term and a set of likelihood terms, one for each child of v. Thus, when sampling from the full conditional for v, we need only consider vertices which are parents, children, or parents of children of v, and we can perform local computations.

2.3.1.1 Conjugate Sampling

In many applications full conditional densities can be expressed in a closed form (conjugate) and thus drawing samples from it can be done using standard algorithms. For instance, the full conditional could be a normal or a gamma distribution from which sampling is straightforward.

2.3.1.2 Slice Sampling

In our networks, some full conditionals are complex and unavailable in closed form. For instance, we cannot directly compute the PDF of a variable that represents the x-coordinate of a point to be localized. In these situations, we can turn to slice sampling, which is a general process that works to estimate arbitrary distributions.

Suppose f is the full conditional density of a variable. An issue in Gibbs sampling is that each time we change the value of one variable, we have changed the underlying f for that instance of the network. Thus, we cannot compute the true joint-density of a variable by simply running through the domain in small increments and building the curve directly, because the curve will change when we change the value of another variable.

The strategy slice sampling follows is to draw randomized values of $f(x)$ for each variable, and follow a procedure to pick randomized values in the domain in a way such that the number of times these occur (or fall into specific discrete ranges) will approximate the PDF of the full conditional.

Suppose we have an initial value for the variable x, x_0. Then, the method uses an auxiliary variable $y = kf(x_0)$, where k is uniformly distributed in $(0, 1)$, to define a *slice* S, such that $S = \{x : y < f(x)\}$ (see Figure 2.1). Assuming we know S, we would like to pick a new value, x_1, uniformly across the domain defined by the slice. However, we can not always easily estimate the edges of S, and so must approximate it with an interval I.

Several schemes are possible in order to find I:

- If the range of the variable is bounded, I can be the whole range. There is thus no computational cost for I. We call this approach *Whole Domain Sampling.*

- We can start with an initial guess w of S that contains the current value of the variable, and then perhaps expand it by a "stepping out" process. The process expands w in steps of size w until both ends are outside the slice or a predetermined limit is reached. For example, in Figure 2.1, if a predetermined limit is not used and w is equal to the width of a bar in the histogram, I might by off from

S by at most one w on each side.

- Given a guess w of S, w can be expanded following a "doubling out" procedure. Doubling produces a sequence of intervals, each twice the size of the previous one, until an interval is found with both ends outside the slice or a predetermined limit is reached. The idea here is that finding the edges of S should be much faster even if we lose some precision in estimating the edges of I.

Both "step out" and "double out" start by positioning the estimate w randomly around the current value x_0. The predetermined limit they may apply to terminate the expansion of w is an interval of size mw, for some specified integer m. Once an interval I has been found, "step out" follows a shrinkage procedure that samples uniformly from an interval that is initially equal to I and which shrinks each time a point is drawn that is not in $S \cap I$ (e.g. point x_2 in Figure 2.1 where $f(x_2) \leq y$). A point picked that is outside $S \cap I$ is used to shrink I in such a way that the current point x_0 remains within it. "Double out" follows the same shrinkage process with some additional constraints (see [54]) for the point that is finally accepted. Depending on the shape of $f(x)$, and the quality of I's approximation of S, we may reject many draws of x.

In practice, to avoid possible problems with floating-point underflow, it is safer to compute $g(x) = -ln(f(x))$ rather than $f(x)$ itself, and thus $S = \{x : g(x) < -ln(k) + g(x_0)\}$. We call $g(x)$ "minus log the full conditional density". Also, there are several variations of slice sampling, like multivariate slice sampling, that updates many stochastic variables simultaneously.

2.3.2 Metropolis Algorithm

A univariate Metropolis algorithm is an MCMC method that chooses the next value of a stochastic variable v by first sampling a *candidate* point y from a *proposal* distribution q. Practically, q is used to propose a random "unbiased perturbation" of the current value of v. For example, q could be a normal distribution with mean the current value of v and variance user defined. It then computes the "gain" in an objective function resulting from this perturbation. A random number U, uniformly distributed in $(0, 1)$,

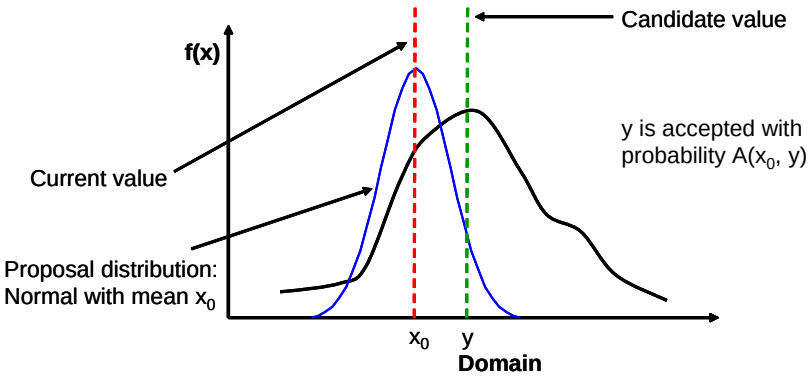


Figure 2.2. Metropolis algorithm

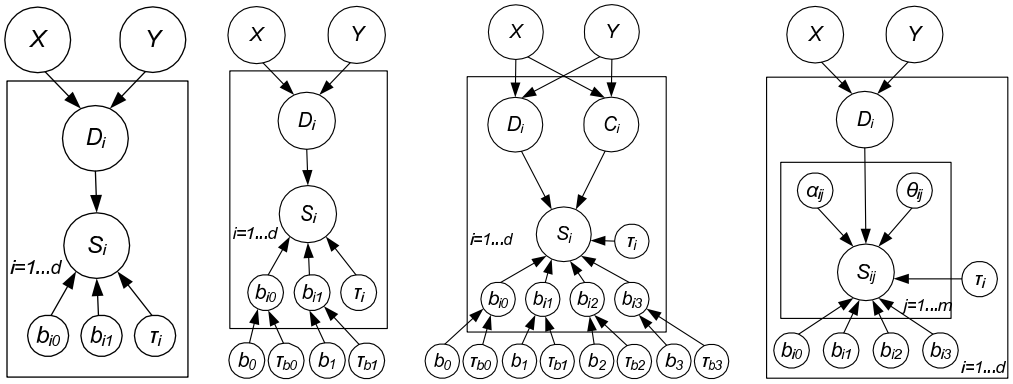
is generated and the candidate point y is accepted with probability $A(v, y)$, otherwise v retains its current value. In this work, the following Metropolis acceptance function is used:

$$A(v, y) = min\left(1, e^{-(g(y)-g(v))/T}\right) \tag{2.5}$$

where g is minus the log full conditional density of v and T is some constant. Figure 2.2 depicts how the Metropolis algorithm works for a random variable x, if the proposal distribution is normal with mean the current value of x.

Heuristically, the Metropolis algorithm is constructed based on a "trial-and-error" strategy. The choice of the proposal distribution is critical for the efficiency of the algorithm. On one hand, it could lead to a large number of candidates y being rejected, and on the other hand it could result in accepting nearly all proposed candidates, but the candidates could be close to each other in the space of the distibution of v. In both cases the algorithm is inefficient as it does not "mix" rapidly.

Gibbs sampling can be seen as a special case of the Metropolis algorithm, since the proposal function for Gibbs is the full conditional of a node and the acceptance function is always one (the candidate point y in Gibbs sampling is always accepted). Finally, there are times when we use the Metropolis algorithm for some nodes of a network and Gibbs sampling for the remaining nodes. This is called Metropolis-within-Gibbs sampling.



(a) Network M_1 (b) Network M_2 (c) Network M_3 (d) Network A_1

Figure 2.3. Bayesian graphical networks using WinBugs plate notation.

2.4 Localization Networks

Figure 2.3 presents a series of Bayesian networks of increasing complexity that embody extant knowledge about Wi-Fi signals as well as physical constraints. The networks are called M_1, M_2, M_3, A_1, and can be used for a two-dimensional location estimation problem in a building with d access points. Each rectangle is a "plate", and shows a part of the network that is replicated; in our case, the nodes on each plate are replicated for each one of the access points.

Vertices X and Y represent location, while vertex D_i represents the Euclidean distance between the location specified by (X, Y) and the ith access point. X and Y are bounded by the length L and the breadth B of a building respectively. Vertex S_i (M_1, M_2, M_3) represents the signal strength measured at (X, Y) with respect to the ith access point. All networks reflect the fact that the signal strength decays approximately linearly with log distance. Specifically, the value of the signal strength, S_i, with respect to the ith access point follows a signal propagation model $S_i = b_{0i} + b_{1i} \log D_i$, where b_{0i}, b_{1i} are parameters specific to the ith access point. The networks capture noise and outliers by modeling the S_i as a Gaussian distribution around the above propagation model with variance τ_i, as shown in expression 2.6:

$$S_i \sim N(b_{i0} + b_{i1} \log D_i, \tau_i) \tag{2.6}$$

Variable(s)	Description
X, Y	x- and y-coordinate of a location.
d	Number of access points.
D_i	Euclidean distance between the location specified by (X, Y) and the ith access point.
C_i	A 0/1 variable that shows whether location (X, Y) shares a corridor with the ith access point.
S_i	Signal strength measured at (X, Y) with respect to the ith access point.
S_{ij}, θ_{ij}, a_{ij}	S_{ij} is the jth signal strength measured at (X, Y) with respect to the ith access point, when the antenna of the access point is at an angle θ_{ij} and the signal is received by the mobile at an angle a_{ij}.
m	Number of signal strength readings that a mobile receives from an access point in one rotation of the antenna (each reading corresponds to a different angle).
b_{ij}	Coefficients of the linear regression model that describes how a signal degrades linearly with log distance.
b_i	Parents of the coefficients of the linear regression model.
τ_i, τ_{bj}	Precision in the Gaussian distribution that describes the variables S_i (S_{ij}) and b_{ij} respectively.

Table 2.1. Variables of the networks M_1, M_2, M_3, A_1 depicted in Figure 2.3.

The hierarchical portion of M_2 (vertices b_0, b_1, τ_{b0}, τ_{b1}) reflects prior knowledge that the different access points behave similarly. This similarity is expressed in the network by making the coefficients of the linear regression model have common parents. As was shown in [51], the model can provide accurate location estimates without any location information in the training data, leading to a truly adaptive, zero-profiling technique for location estimation. Practically this is really significant, since the location measurement process is slow and human-intensive.

Network M_3 models the corridor effect. That is, when an access point is located in a corridor, the signal strength tends to be substantially stronger along the entire corridor. Variable C_i in M_3 takes the value 1 if location (X, Y) shares a corridor with access point i and 0 otherwise. We define "sharing a corridor" as having an x- or y-coordinate within three feet of the corresponding access point coordinate. Since corridor width varies from building to building, this definition should vary accordingly, although we do not pursue this here.

Network A_1 incorporates both the knowledge of angle-of-arrival of the signal (AoA) and the knowledge of received signal strength (RSS). In A_1 there are m signal strength readings at a particular location (X, Y) with respect to the ith access point; each is measured when the rotational directional antenna of the access point is at an angle

θ_{ij} and the signal is received by the mobile at an angle a_{ij}. The ratio $360/m$ is called granularity G and determines the angle intervals at which the signal strengths are measured in a rotation of the antenna.

Table 2.1 summarizes the description of the variables used in our localization networks. More details about the networks can be found in [25, 51].

2.5 Experimental Results

In this section we present our experimental results that were all performed on a Pentium 4 PC with a 2.8-GHz CPU, 1 GB of RAM and running Microsoft Windows XP. Our software was implemented in ANSI C. All of our networks use training data in the learning process that maps signals to locations (M_1, M_2, M_3, A_1) and also to angles (A_1). For M_1, M_2, M_3 we used the BR dataset from [51] that contains 253 training points, was collected in a building that measures 255ft $\times$ 144ft and has 5 access points. For A_1 we used a dataset from [25] consisting of 20 training points collected in a building that measures 200ft $\times$ 80ft and has 4 access points. We follow the leave-n-out method, where n denotes the number of points to localize.

2.5.1 Profiling a Gibbs Sampler

We first implemented a Gibbs sampler for all our networks. The sampler uses slice sampling for variables X, Y (M_1, M_2, M_3, A_1) and α_{ij} (A_1). All the other stochastic quantities are sampled using either a conjugate normal or a conjugate gamma method. The solvers were implemented in such a way so that the values of deterministic nodes (nodes that are a logical function of other nodes in the network) that do not change in every iteration are calculated only once in the whole sampling process. Examples of such cases are nodes D_i and C_i for points in the training set with location information.

Figures 2.9(a)-2.9(d) depict the average execution time breakdown (over 30 runs) of Gibbs sampling for our four networks, when the slice sampling method applies step out with w=1ft and m=10 (mw is used as a limit to terminate the step out process; see Section 2.3.1.2). For A_1 the value of granularity is $G = 120$. We see that, as the

Algorithm	Description
met wd	Univariate Metropolis with proposal uniform over the whole domain of X, Y (uniform over the whole domain for angle).
met sd=k (or sd=k, l)	Univariate Metropolis with proposal Gaussian whose standard deviation is k and mean the current value (the standard deviation is l for angle).
slice wd	Univariate slice sampling over the whole domain of X, Y (univariate slice sampling over the whole domain for angle).
slice so=k (or so=k,l)	Univariate slice sampling for X, Y by doing step out with w=k and m=10 ($w = l$ for angle).
slice do=k (or do=k,l)	Univariate slice sampling for X, Y by doing double out with w=k and m=10 ($w = l$ for angle).
slice2d wd	Two-dimensional (X and Y are updated together) slice sampling over the whole domain of X, Y (univariate slice sampling over the whole domain for angle).

Table 2.2. All MCMC algorithms. The text in the parentheses refers to A_1.

number of points we localize increases from 1 (Figures 2.9(a), (b)) to 10 (Figures 2.9(c), (d)), slice sampling dominates the total time of the sampler. There is also an increase on the time of the conjugate methods and this is because when we localize 10 points, there are more deterministic nodes whose values need to be estimated in every iteration. Slice sampling takes considerably more time in A_1 when compared to the other three networks, because it is used not only for the X and Y coordinates, but also for the angle α_{ij}.

2.5.2 MCMC Algorithms

To speed slice sampling we experimented with several variations as well as we tried Metropolis-within-Gibbs sampling. Table 2.2 summarizes the MCMC methods we used for Bayesian inference on our networks and can be categorized into Metropolis-within-Gibbs sampling (met algorithms) and Gibbs sampling (slice algorithms). The Metropolis-within-Gibbs samplers apply the Metropolis algorithm for X, Y, a_{ij} and conjugate sampling for the remaining stochastic quantities. For our experiments we used two forms for the proposal distribution of the Metropolis algorithm. A uniform distribution over the whole domain of the variables X, Y, a_{ij}, since these variables are bounded with domain $(0 \dots L)$, $(0 \dots B)$ and $(0 \dots 2\pi)$ respectively, and also a Gaussian centered on the current value and standard deviation k for X, Y and l for a_{ij}. On the other hand, Gibbs samplers apply slice sampling for X, Y, a_{ij} and conjugate sampling

for the other variables. We have implemented four types of slice sampling. Specifically, we did univariate slice sampling by: (a) sampling uniformly over the whole domain of X, Y, a_{ij}, (b) doing step out, (c) doing double out. For the latter two cases we used w=$k$ft for X and Y, w=l radians for a_{ij} and m=10 to terminate the expanding process. The fourth type we implemented was a two-dimensional slice sampling (multivariate approach) that updates X and Y simultaneously and samples uniformly over the domain of X and Y, while for a_{ij} it follows univariate slice sampling over the whole domain.

2.5.3 Comparing Algorithms

Figure 2.10 presents the average performance achieved by our algorithms expressed as relative accuracy and standard deviation vs. time. We call *relative accuracy* the Euclidean distance of the results of our solver compared to the ones from WinBugs after running WinBugs for 10000 iterations as burnin, 100000 additional and having the over relax option set (see [66]). Specifically, the Euclidean distance is estimated using the mean of the variables X and Y. As was shown in [25, 51], this number of iterations provides adequate convergence for the networks that we consider here, according to standard WinBugs diagnostics. The samples of the burnin iterations are discarded and only the samples of the burnin count in the estimation of analytic summaries for the stochastic quantities. The idea here is that the per-variable statistics of long runs of a well-tested, widely-used solver should converge to the true distribution as defined by the combination of the model and data. All our results are thus compared against this "gold standard", as opposed to "ground truth" accuracy of the true location of the object.

We ran our solvers with 100 iterations as burnin and the additional ranged from 1000 to 10000 with increments of 1000. In each case, the results are the average of 30 runs; in every run a different set of point(s) is chosen to be localized. We observe that univariate slice sampling over the whole domain ("slice wd" in the graphs) has the best ratio of relative accuracy vs. time for all networks; it can localize 1 or 10 points with relative accuracy less than 1ft in less than half a second. Moreover, "met sd=1"

and "slice so=1" have the worst performance, since they converge very slowly to the WinBugs solution as can be seen from Figure 2.10(e). Hence, they fail to mix rapidly. Among the remaining algorithms, "slice do=1", "met sd=20" and "slice so=10" are not stable in providing a solution, as indicated by their standard deviation in Figures 2.10(f), 2.10(g), 2.10(h), which show that they need more than 10000 iterations so that the standard deviation becomes small.

Furthermore, as can be seen in the graphs of Figure 2.10, some lines are shorter than others. The reason is that the computational cost per iteration is different for our algorithms and as a result some algorithms take less time to execute than others. The factor that differentiates the computational cost among all these algorithms is the number of evaluations of minus log the full conditional g of a stochastic variable (see Section 2.3.1.2). Figure 2.11 shows some more results for different sizes of the training set N. As can be seen, even for smaller sizes of N (e.g. 51, 101), "slice wd" has the best performance.

Figure 2.4 depicts the average number of evaluations per variable X and Y for network M_1. We observe that the Metropolis algorithms perform fewer number of evaluations when compared to the slice sampling algorithms. The reason is that slice sampling algorithms that follow the step out and double out process ("slice so" and "slice do" in the graphs) evaluate g several times until they get an estimate I of the slice S (see Section 2.3.1.2). In addition, all slice sampling methods follow a shrinkage procedure during which g could potentially be evaluated several times until the next value to be accepted is found. On the other hand, Metropolis algorithms evaluate g once before a candidate point y is proposed and once after. The number of evaluations per variable X, Y are similar for the other Bayesian networks.

Among the slice sampling algorithms, we see that two-dimensional slice sampling ("slice2d wd") and univariate slice sampling ("slice wd") over the whole domain of X and Y have the fewest evaluations. The first of the two takes advantage of the fact that X and Y have the same full conditional and as a result g is estimated once when X and Y need to be updated, whereas the latter estimates g once for X and once for Y. Since "slice wd" and "slice2d wd" take the whole domain as an estimate of the slice,

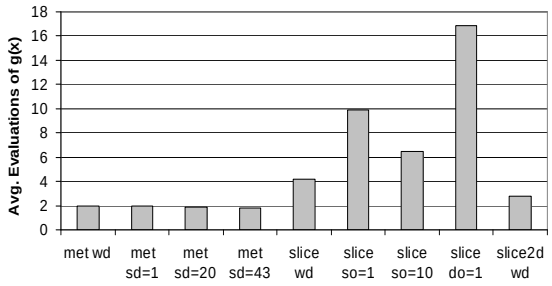


Figure 2.4. Average number of evaluations per variable X and Y after 10000 iterations of minus log the full conditional $g(x)$ for M_1 when we use 253 training points to localize 1.

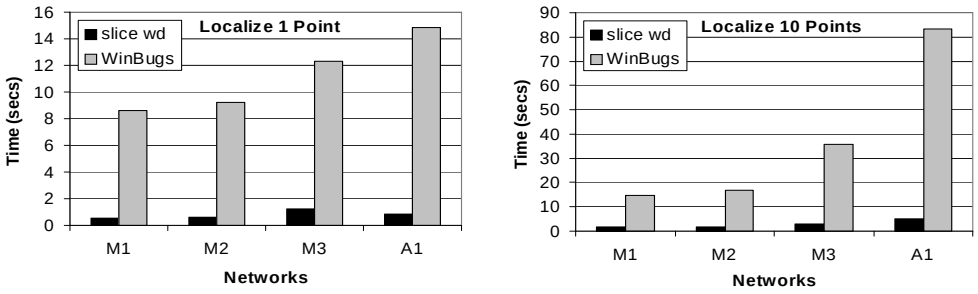


Figure 2.5. Execution time comparison of "slice wd" against WinBugs for localizing 1 point and 10 points. The total number of iterations are 10000 and the number of training points are 253 for M_1, M_2, M_3, and 20 for A_1.

the only evaluations of g they perform is in the shrinkage process. Specifically, "slice wd" performs 4.13 evaluations per variable on average in the shrinkage process, out of which 2.13 are rejections. This is a clear indication that g is relatively flat, because the method can find a point within the slice with only a few rejections.

Figures 2.9(e)-2.9(h) depict the average execution time breakdown of Gibbs sampling, when slice sampling uses the "slice wd" method. We see that "slice wd" takes less time when compared to "slice so=1" that is shown in Figures 2.9(a)-2.9(d). Particularly, slice sampling is 2.06 faster for M_3 to 2.57 faster for A_1 when we localize 1 point, and is 2.13 times faster for M_3 to 2.51 faster for A_1 when we localize 10 points. Finally, Figure 2.5 compares the average execution time (over 30 runs) of Gibbs sampling when using "slice wd" to WinBugs (over relax option set). Our solver is faster than WinBugs by a factor that ranges from 9.8 (M_3) to 17.9 (A_1) for localizing 1 point, and from 9.1

(M_1) to 16.1 (A_1) for 10 points.

2.5.4 No Location Information

Work [51] showed that M_2 can localize devices with no location information in the training set. However, when we ran our solver for 51 signal vectors and with 51 unknown positions, the solver occasionally returned a solution different from WinBugs. Our solver found an alternate, but incorrect, solution for the values of the coefficients of the linear regression model that describes how a signal degrades linearly with log distance (Equation 2.6). Specifically, although the parameters b_{i0} and b_{i1} are supposed to be negative and positive respectively, our solver found a solution with inverted signs for these parameters. When we have location information in the training set we never get alternate solutions, because the location information restricts the sign of b_{i0} to be negative. So, we bounded b_{i0} to be negative when there is no location information. Figure 2.6 shows relative accuracy vs. time after bounding b_{i0}. Specifically, Figures 2.6(a), 2.6(b) show the performance of seven out of nine MCMC algorithms that behave similarly. We see that these algorithms can localize a device with relative accuracy less than 3ft in six seconds with the exception of "slice do" that converges more slowly to the WinBugs solution. On the other hand, Figures 2.6(c), 2.6(d) show that the remaining two algorithms, "slice so=1" and "met sd=1", perform poorly.

2.6 Analytic Model

In this section we describe an analytic model that gives us insight into when whole domain sampling will be computational more efficient than other methods. To keep the analysis tractable, we use a double exponential distribution and build its histrogram by comparing whole domain sampling to a fix-sized step out process.

To quantify computational costs, our analysis compares the number of times univariate slice sampling evaluates $g(x)$ when using: (a) the whole domain as an initial guess of the slice S, (b) the "step out" process. We assume that x follows a double

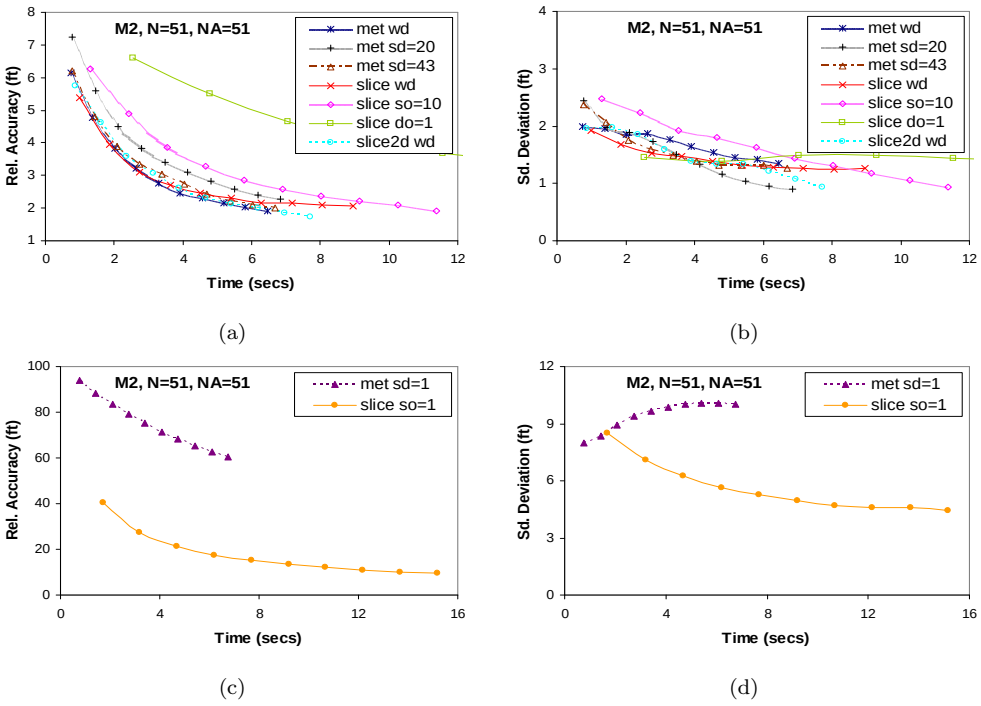


Figure 2.6. Relative accuracy and standard deviation vs. time for N=51 training points with no location information after bounding the coefficients b_{i0} of the linear regression model.

exponential distribution with probability density:

$$H(x;\lambda) = \frac{\lambda e^{-\lambda|x|}}{2} \qquad -\infty < x < \infty,\ \lambda > 0 \tag{2.7}$$

Although x is not bounded, we assume that, for a specific λ, there exists an interval (a_d, b_d) that contains 99% of the distribution and so consider x bounded within this interval. Since the distribution is symmetric with respect to the y axis and has mean zero, $a_d < 0$ and $b_d > 0$. For the variable x then:

$$g(x) = -ln(H(x;\lambda)) = -ln(\lambda/2) + \lambda|x| \tag{2.8}$$

Recall slice sampling picks as a new value of x a point within $S \cap I$. In our case g is unimodal (i.e., has a single peak), and hence S consists of a single interval centered around the point $(0, 0)$. Using the whole domain as an initial guess of S, $I = (a_d, b_d)$, whereas if we use step out, $I = (a_s, b_s)$, for some interval (a_s, b_s) returned by starting

with an initial guess w of the slice and then perhaps expanding it. However, since S is a single interval (as mentioned earlier), I will contain the whole interval S in both cases. The total number of evaluations E of $g(x)$ for univariate slice sampling is:

$$\begin{aligned} E &= \textit{Evaluations to define } S + \\ &\quad \textit{Evaluations to estimate } I + \\ &\quad \textit{Evaluations in the shrinkage process} \end{aligned}$$

Defining slice S requires one evaluation of $g(x)$ so as to determine the value of the auxiliary variable y (see Section 2.3.1.2). Estimating I entails zero evaluations of $g(x)$ using the whole domain, whereas using step out the number of evaluations will be $(b_s - a_s)/w + 1$. The size of (a_s, b_s) is always a multiple of w and, since step out starts by positioning w randomly around the current value of x, it is equal to:

i) $2w$, if $w > S$ and only one endpoint of w is outside S

ii) w, if $w > S$ and both endpoints of w are outside S

iii) $2w$ or $3w$, if $w = S$

iv) $\lceil S/w \rceil w$ or $(\lceil S/w \rceil + 1)w$, if $w < S$ and $\lceil S/w \rceil \neq S/w$

v) $(S/w + 1)w$ or $(S/w + 2)w$, if $w < S$ and $\lfloor S/w \rfloor = S/w$

In order to approximate the average number of evaluations of $g(x)$ required to estimate I using step out, we simplify as follows. First, when $w > S$ (cases i, ii) we assume that the size of (a_s, b_s) is w (case ii). Choosing (i) or (ii) depends on how w is positioned around the current value of x which is determined the moment of actual sampling. The number of evaluations $(b_s - a_s)/w + 1$ will differ only by one when using case (i) vs. (ii) and hence the choice should not introduce a lot of error. Moreover, for large values of w, we expect that choice (ii) will most likely occur in a real MCMC simulation. For the same reasons, in case (iii) we assume that the size will be $2w$ and in cases (iv), (v), the size will be $\lceil S/w \rceil w$ and $(S/w+1)w$ respectively. Finally, in all cases above, we use the mean value, $\bar{S}$, of S which we estimate as follows. As explained in

Section 2.3.1.2, S is defined by using a random variable k that is uniformly distributed in $(0, 1)$ with probability density $P(k)$. The size of S for function g in equation (2.8), given a specific x and k, is $S(x, k) = 2(|x| - ln(k)/\lambda)$. Thus, the mean size of S will be

$$\begin{aligned} \bar{S} &= \int_{a_d}^{b_d} H(x;\lambda) \int_0^1 P(k) S(x,k)\, dk\, dx \\ &= \frac{4 + (\lambda a_d - 2)e^{\lambda a_d} - (\lambda b_d + 2)e^{-\lambda b_d}}{\lambda} \end{aligned}$$

After estimating I, slice sampling follows a shrinkage procedure until it identifies a new x. The number of evaluations of $g(x)$ in the shrinkage process, either using the whole domain or step out, will be equal to the number of rejections (for the proposed points outside $S \cap I$) plus one for the point that is finally accepted. As shown in the discussion in [54], the interval I will shrink exponentially with rate 0.5. So, the size of I at the nth trial will be $I_n = Ie^{-0.5n}$. The probability of having n rejections in the shrinkage procedure before finding a new x is:

$$p_n = \prod_{i=0}^{n-1} \left(1 - \frac{\bar{S}}{Ie^{-0.5i}}\right) \frac{\bar{S}}{Ie^{-0.5n}}$$

Since S consists of a single interval, the size of I at the nth trial is greater than $\bar{S}$. So,

$$\frac{\bar{S}}{Ie^{-0.5n}} \leq 1 \Rightarrow n \leq 2ln\left(\frac{I}{\bar{S}}\right) \tag{2.9}$$

In our model we follow a conservative approach and assume that the number of rejections in the shrinkage process is equal to the upper bound of n in equation (2.9). Thus, the total number of evaluations for the whole domain E_{wd} and step out E_{so} is:

$$\begin{aligned} E_{wd} &= 2ln\left(\frac{b_d - a_d}{\bar{S}}\right) + 2 \\ E_{so} &= \frac{b_s - a_s}{w} + 2ln\left(\frac{b_s - a_s}{\bar{S}}\right) + 3 \end{aligned}$$

Figure 2.7 compares the number of evaluations of $g(x)$ from a slice sampler and our analytic model. We consider the domain of x to be the range $(-20, 20)$ because

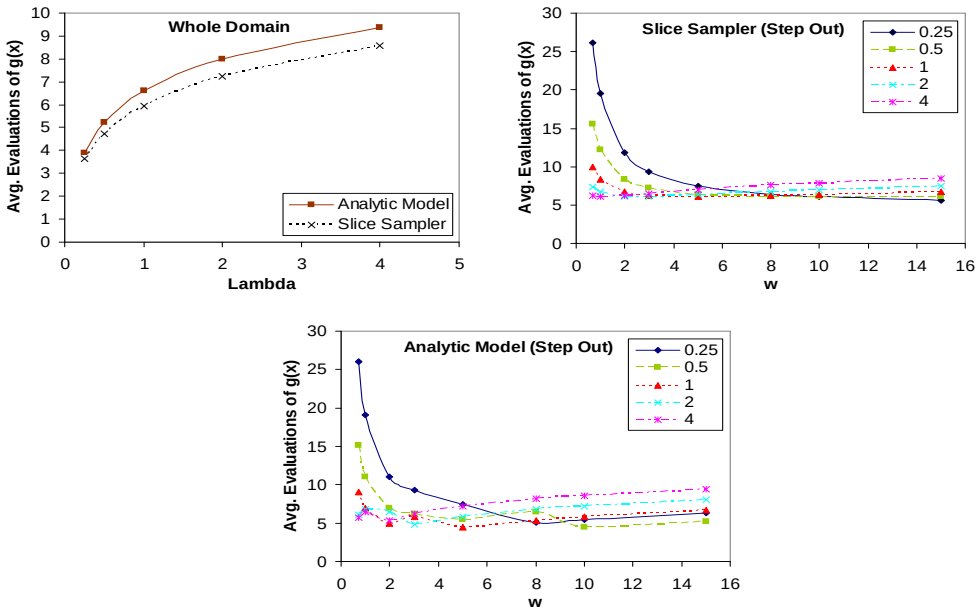


Figure 2.7. Comparison of the number of evaluations of minus log the full conditional $g(x)$ for the double exponential distribution from a slice sampler (1000000 iterations) and the analytic model.

the values of the probability density outside this range are close to zero. The graphs show the analysis predictions closely follow the results of the sampler for step out and whole domain sampling, although due to the simplifications that we made the analytic model sometimes underestimates the number of evaluations. The results also show that as the width, w, increases, the number of evaluations of $g(x)$ approaches the number of evaluations of the whole domain (as expected). Most importantly, for λ=2 there are sizes of w ($\geq$2) that step out has fewer evaluations of $g(x)$ than using the whole domain, whereas for $\lambda \leq 1$, using the whole domain gives fewer evaluations. Hence, a distribution has to be at least as peaky as a double exponential distribution with λ=2, in order to use step out, whereas for distributions with higher variance, using the whole domain is more computationally efficient.

Figure 2.8 compares the shape of the full conditionals f of the double exponential distribution for three values of λ, the x-coordinate of a point to be localized by M_1 and one of the angles α_{ij} in A_1. Comparing the shapes of the two distributions this way is "fair", because the double exponential distribution is the same as a single-node Bayesian network whose prior term is given by equation (2.7) but has no likelihood

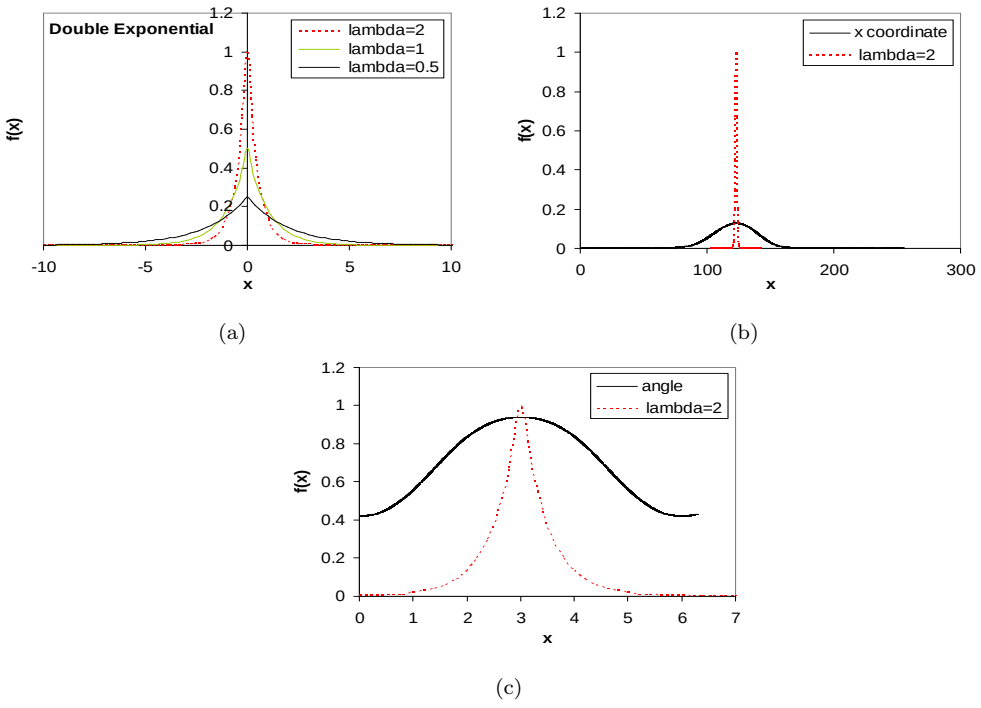


Figure 2.8. Full conditionals of the (a) double exponential, (b) x-coordinate of a point to be localized by M_1, (c) angle a_{ij} in A_1. (b), (c) also depict the double exponential with λ=2 whose mean has been shifted to match the mean of the latter two full conditionals.

terms since it has no children. Our solvers for M_1 and A_1 generated values of the full conditionals of the x-coordinate and the angle α_{ij} uniformly over their domain at some specific iteration, once the Markov chain had reached its equilibrium. We see that the latter two full conditionals are much more spread out than the double exponential with λ=2. This is the reason that "slice wd" has fewer evaluations of g (hence shorter execution time) than "slice so" in all four networks, when used to sample for the X, Y and α_{ij} stochastic variables.

2.7 Importance Sampling

One major drawback of the MCMC methods is that some of the samples are discarded. However, for every discarded sample, there is some computational cost that we pay. These kind of samples are the ones in the burnin iterations, the ones that are rejected in the shrinkage process of the slice sampling algorithm, and also the ones that do not satisfy the Metropolis criterion (Equation 2.5) in the Metropolis algorithm. In an effort

to reduce the time we provide location estimates with our Bayesian networks, we tried to identify methods that use every drawn sample, and hence no value is discarded.

One such method we experimented with was Importance Sampling (IS) [47]. The method draws a value for a random variable from its prior distribution, regardless of its current value. Thus, IS is a Monte Carlo (MC) method rather than an MCMC; the new state does not depend on the previous. Also, IS does not have burnin iterations, but all drawn values contribute to the estimation of the value of a random variable. For every value that is drawn, the method estimates a weight w, which corresponds to the likelihood of the variable the moment the value was drawn. So, for a variable x, and for a number of iterations $iter$, IS estimates the mean of the variable as follows:

$$mean(x) = \frac{\sum_{i=1}^{iter} value_i(x) * w_i(x)}{iter} \tag{2.10}$$

We applied IS to infer values for the x, y-coordinates in our Bayesian networks. So, instead of applying slice sampling and the Metropolis algorithm for these two types of variables, we used IS, whereas for the other unknowns in the networks we still used conjugate sampling. Equation 2.4 shows that the full conditional f of a variable is equal to prior $\times$ likelihood. As mentioned in Section 2.3.1.2, instead of f, we use $g(x) = -ln(f(x))$ to avoid possible problems with floating-point underflow. Therefore:

$$g(x) = -ln(f(x)) = -ln(prior) - ln(likelihood) \tag{2.11}$$

Since the prior of the x, y-coordinates in our Bayesian networks are uniform over the length and breadth of a building, in Equation 2.11, $-ln(prior) = 0$ and hence $g(x) = -ln(likelihood)$. We estimate the value of w in Equation 2.10 as follows:

$$w_i(x) = \frac{e^{g_i(x) - \max_i(g_i(x))}}{\frac{\sum_{i=1}^{iter} e^{g_i(x) - \max_i(g_i(x))}}{iter}} \tag{2.12}$$

Equation 2.12 reveals that we have only one evaluation of g per iteration for a variable,

unlike slice sampling which has 4.13 and Metropolis which has 2 (Figure 2.4). Thus, IS is computationally more efficient than these two MCMC methods. Moreover, the closer the value of $g_i(x)$ to $max(g_i(x))$, the higher the weight w_i that is assigned to the corresponding $value_i(x)$ in Equation 2.10. This means that IS assigns a lot of weight to drawn values with very high likelihood.

Figure 2.12 compares Importance Sampling to whole domain sampling with respect to relative accuracy and standard deviation vs. time for BN M_1, for different sizes of the training set N and points to localize NA. The experiments were done on a 550-MHz CPU, and the number of iterations ranged from 2000 to 10000 with increments of 1000 (out of these iterations, 1000 were burnin for whole domain sampling). The graphs show that whole domain sampling achieves better relative accuracy and standard deviation in all cases. When localizing one point, the difference in relative accuracy between the two algorithm ranges from 6.1ft to 8.3ft for 2000 iterations, and from 2.4ft to 3.2ft for 10000 iteration. When localizing 10 points, the difference ranges from 6.6ft to 13.8ft for 2000 iterations, and from 4.1ft to 13.7ft for 10000 iterations.

The graphs also reveal that IS is faster, since the IS lines are shorter than the lines of whole domain sampling. More specifically Figures 2.13, 2.14 compare the absolute execution time of the two algorithms, as well as the percentage of time reduction that we achieve with IS with respect to whole domain sampling. For 10000 iterations, when localizing one point the reduction can range from 10% to 20%, whereas for 10 points the reduction ranges from 39% to 47%.

Overall, we conclude that, although IS can not outperform whole domain sampling, it is a very simple and computationally efficient algorithm that can give decent localization results for our Bayesian Networks.

2.8 Related Work

There are many active research efforts developing localization systems for wireless and sensor networks. We cannot cover the entire body of work in this section. Rather, we first give the reader a general sense of the approaches used and then cover the related

work on probabilistic inference using MCMC methods.

In general, RSS-based localization systems have been shown to have average accuracies of 6-15ft depending on the level of training data used in a specific environment, for instance, a specific building floor [9, 12, 26]. The key advantage of these approaches is that they can use the existing packet traffic to localize. The absolute accuracies of the Bayesian networks explored here are on the higher end of these systems, with average accuracies of 15ft [51], but require much less training data, often 10 points or less, compared with the 100 or more points needed for other approaches. Other positioning strategies that use ultrasound [62] or generate specific radio waveforms can have higher accuracies, often less than 1m, but require either additional infrastructure or custom radios.

[26,51] are the first to propose Bayesian networks as a location estimation technique. They exploit signal strength information from a collection of access points to localize simultaneously a set of terminals. Additionally, one of their key findings is a model that provides accurate location estimates without any location information in the training data, leading to a truly adaptive, zero-profiling technique. The networks are shown to be robust and competitive to the state of the art approaches for position estimation. [25] extends the previous work, by incorporating the angle-of-arrival (AoA) of the signal along with the received signal strength (RSS) for better position estimation. They show that such a solution reduces the size of the training examples needed to reach the same performance of Bayesian networks that rely solely on RSS. Moreover, [29] proposes the use of Bayes filters in real-world location estimation tasks common in pervasive computing. Specifically, they illustrate how particle filters, a variant of Bayes filters, can be used to estimate a person's location using multiple inaccurate ID sensors such as MIT's Cricket ultrasound tags and VersusTech infrared badge system.

[53,54] provide an extensive study on methods used for probabilistic inference using MCMC methods, such as Gibbs sampling, slice sampling and the Metropolis algorithm. Also, [6] describes a slice Gibbs sampler which is essentially off-the-shelf (i.e., requires no tuning). Unlike slice sampling that slices the prior $\times$ likelihood, the algorithm in [6] slices only the likelihood. In our work, we emphasize more on the computational cost

of MCMC methods as well as how easy it is to use them in an automatic way.

2.9 Summary

In this chapter we show how to reduce the computational cost of solving Bayesian networks used for indoor localization. We introduced a novel approach to sampling these networks that we call *whole domain sampling*. Our results show that the shape of the full conditionals of the coordinates to be estimated as well as the angle of the received signal strength are ideally suited for whole domain sampling. It achieved the best convergence time when compared to other algorithms. Specifically, we can converge to a value of less than 1ft of the position estimated by the WinBugs general-purpose solver (given a very long run) within half a second if we try to localize 1 or 10 points. Also, our method results in execution times that are 9 to 17 times faster than WinBugs. Moreover, we can localize 51 points with no location information to within 3ft of the WinBugs solution in six seconds. Such a method is very appealing because it constitutes a "black-box" sampler (i.e., requires no tuning) for our networks.

Finally, to demonstrate a theoretical foundation on why our approach works, we present an analytic model that tells us how flat a distribution of a stochastic variable should be in order for whole domain sampling to be computationally more efficient than other methods. We show that the actual distributions in our networks easily fall into the regime where whole domain sampling is the best choice.

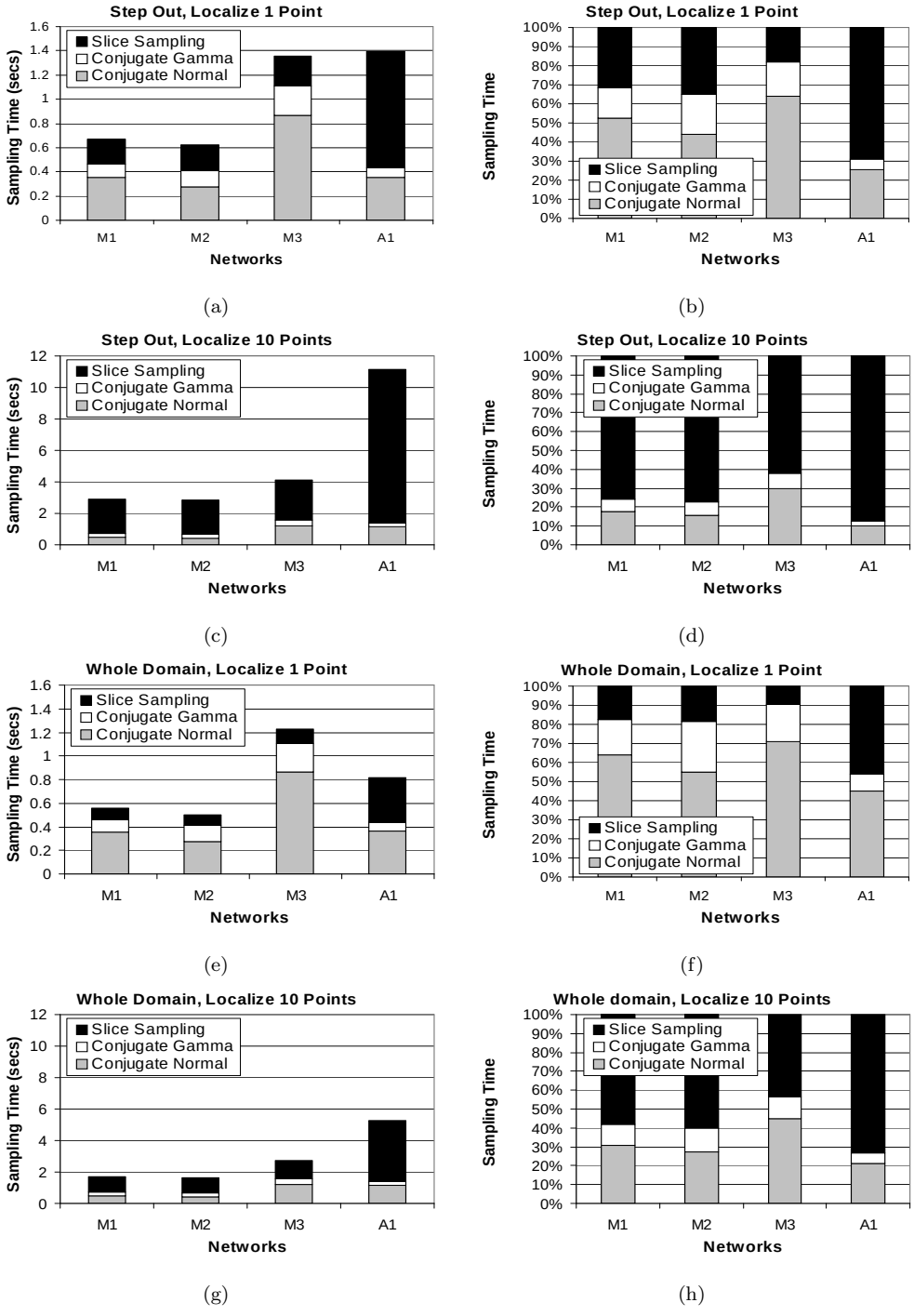


Figure 2.9. Breakdown of the average execution time of Gibbs sampling when slice sampling uses step out (a)-(d) and the whole domain (e)-(h). Graphs (b), (d), (f), (h) depict phases as a percentage of the absolute whole time shown in graphs (a), (c), (e), (g). The total number of iterations are 10000, the number of training points are 253 for M_1, M_2, M_3, and 20 for A_1.

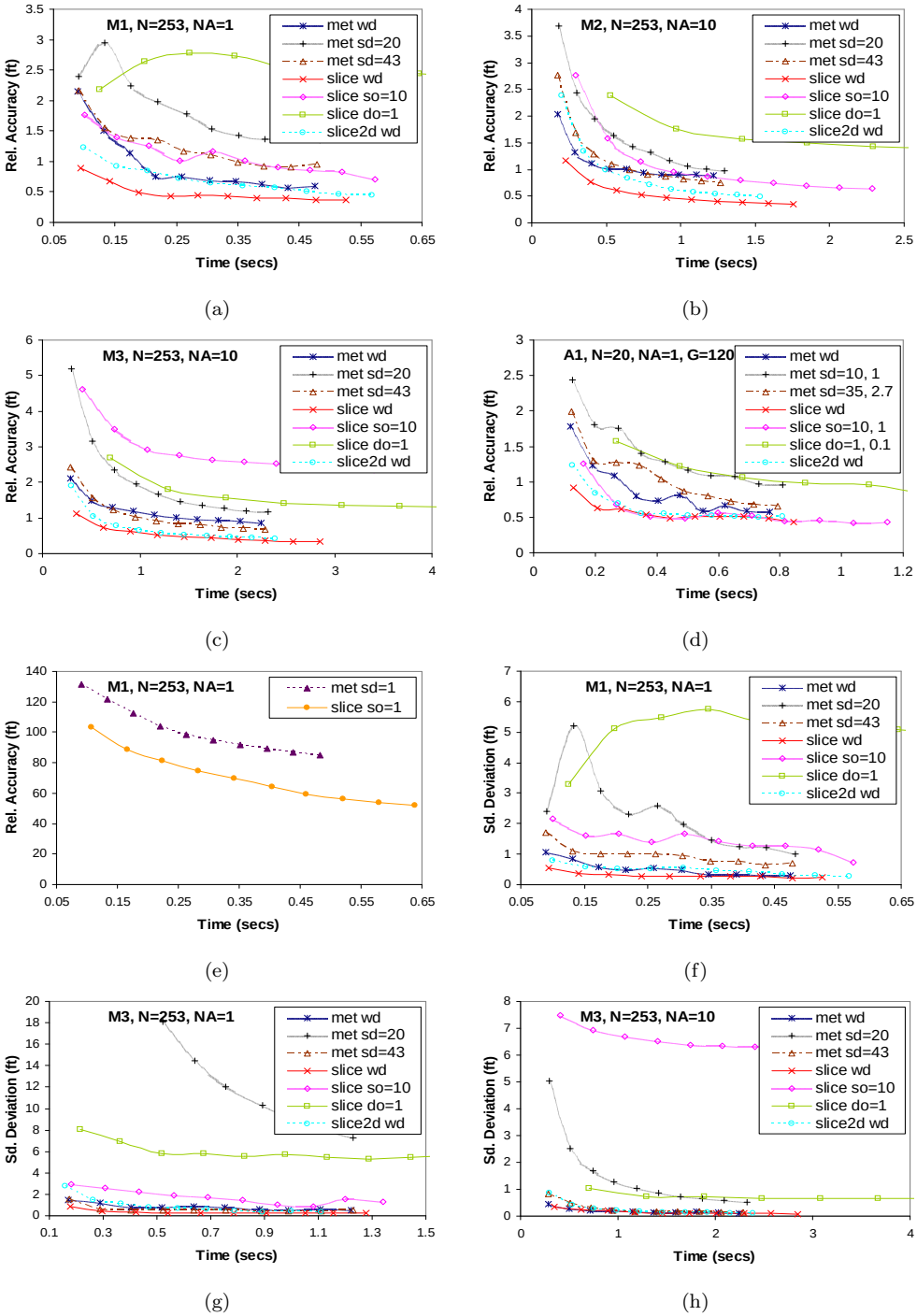


Figure 2.10. Relative accuracy and standard deviation vs. time for different MCMC algorithms (see Table 2.2). N is the number of training points out of which we localize NA points. The size of w is in feet for X, Y, and radians for a_{ij}.

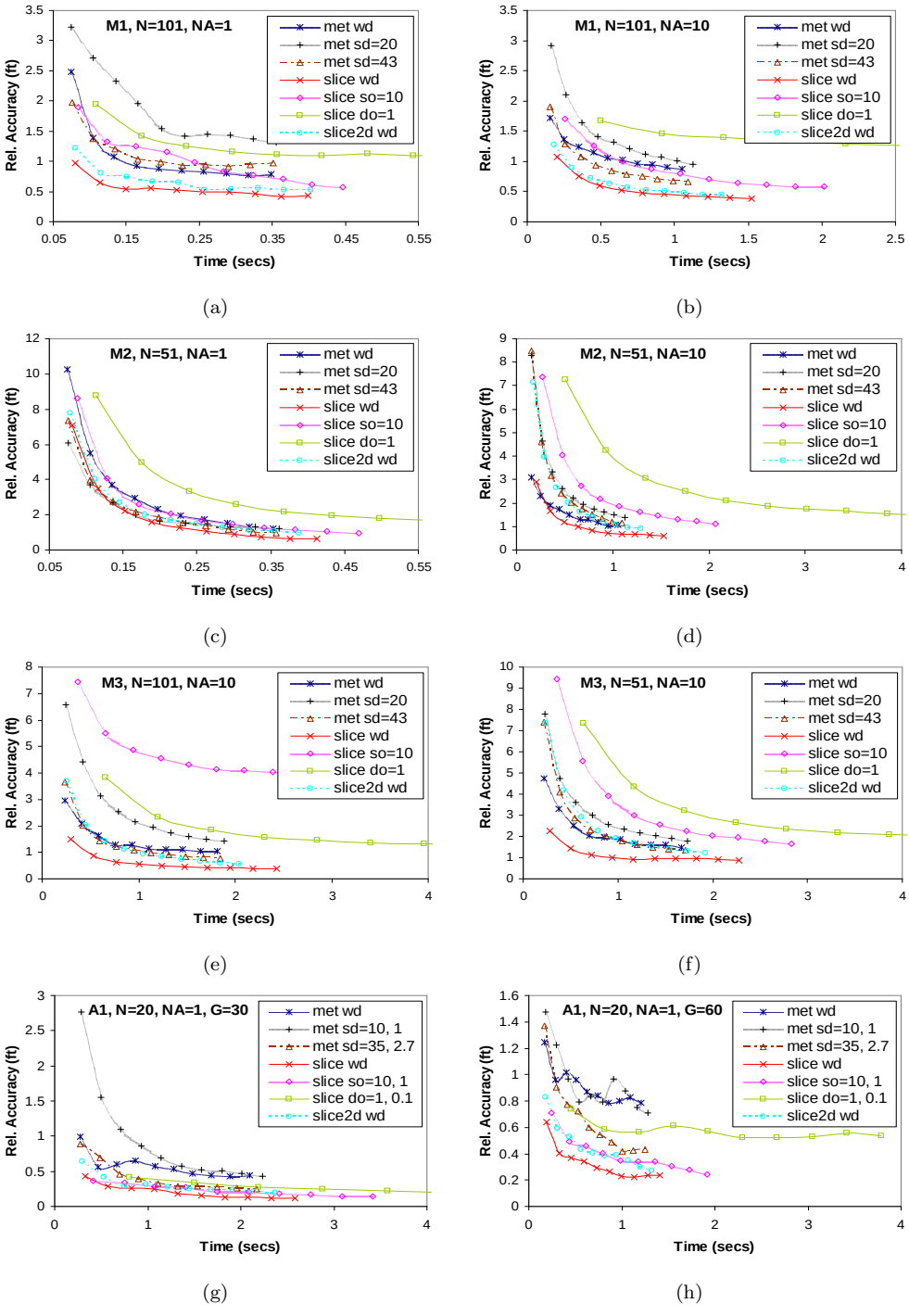


Figure 2.11. Relative accuracy vs. time for different algorithms (see Table 2.2). N is the number of training points out of which we localize NA points. The size of w is in feet for X, Y, and radians for a_{ij}.

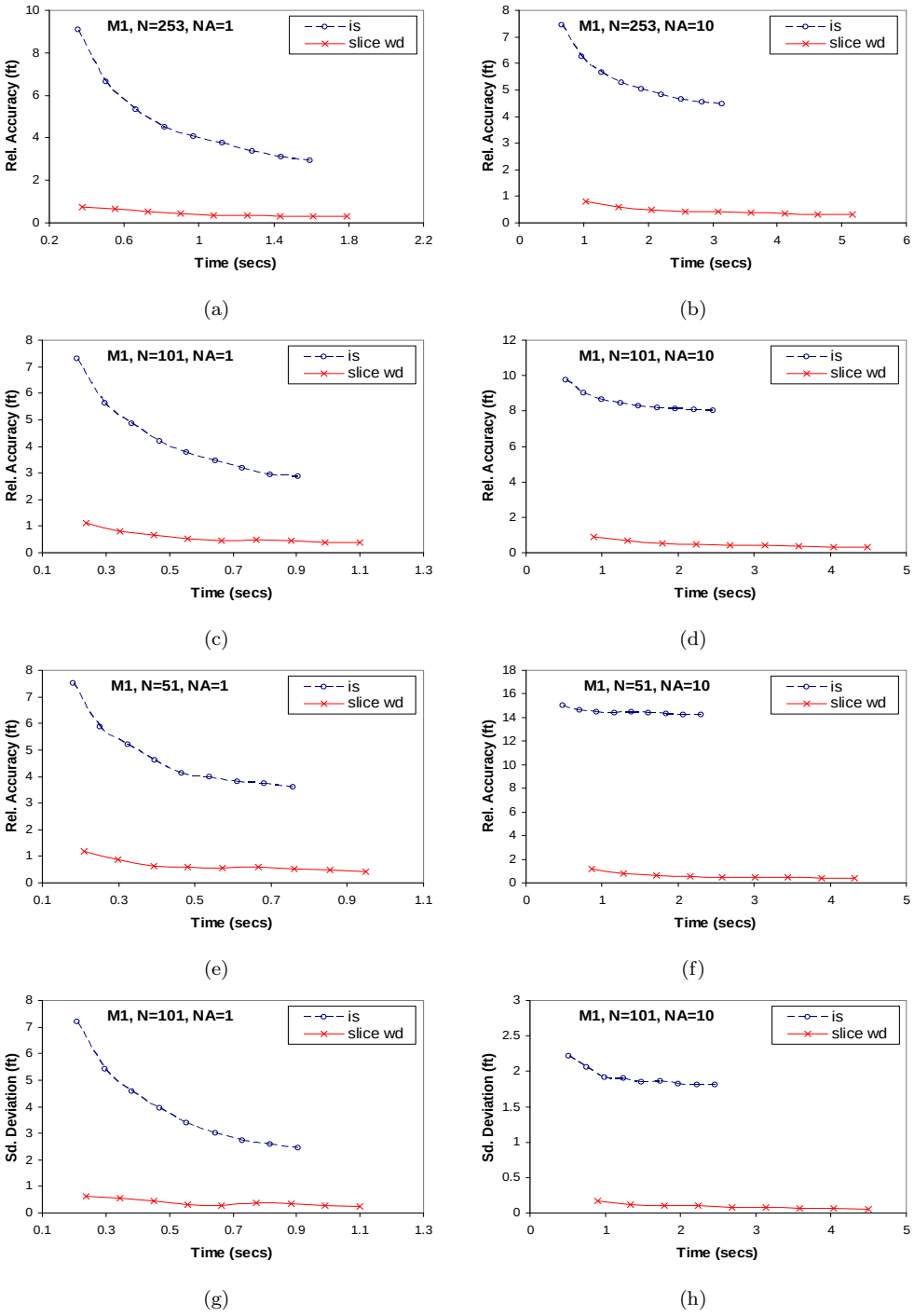


Figure 2.12. Relative accuracy and standard deviation vs. time for importance sampling (is) and whole domain sampling (slice wd). The results are for Bayesian network M_1 when localizaing 1 and 10 points on a 550-MHz CPU.

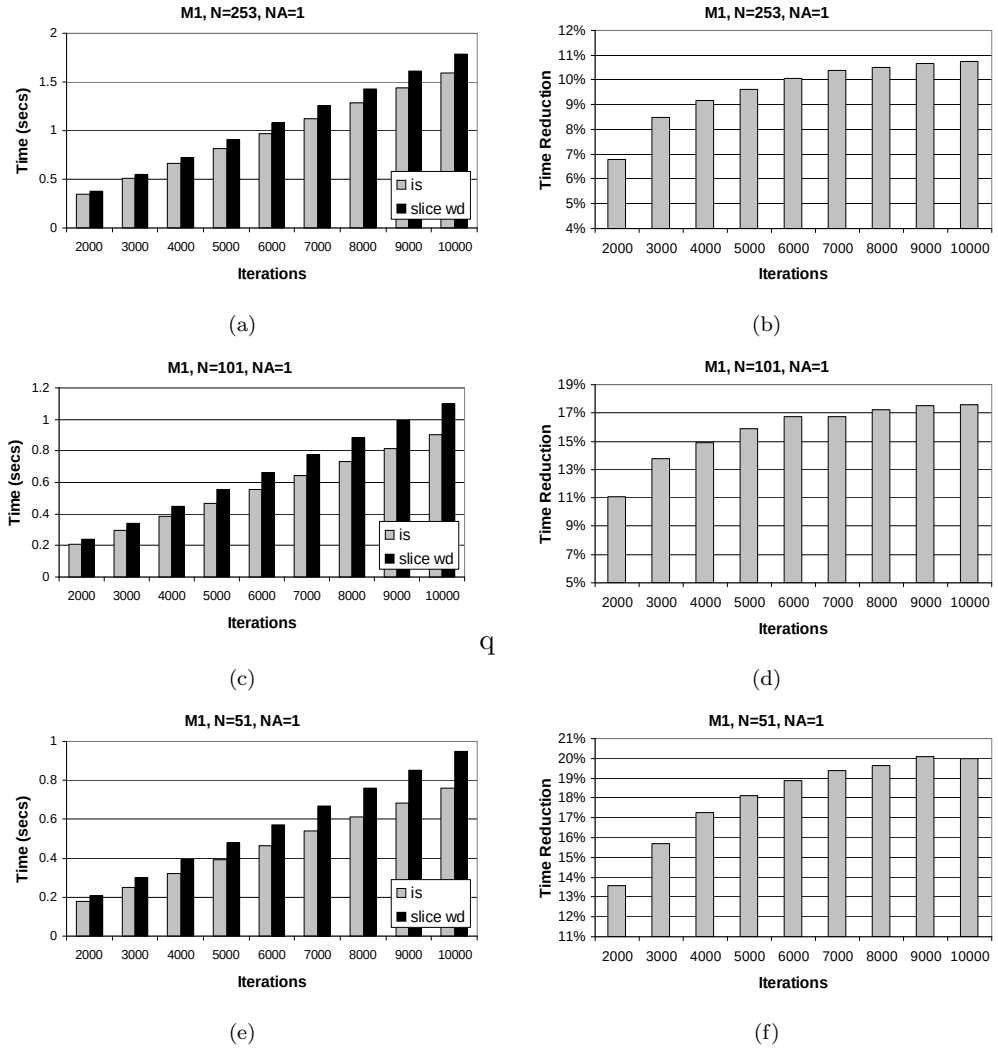


Figure 2.13. Absolute time (a), (c), (e) of importance sampling (is) and whole domain sampling (slice wd) and percentage of time reduction (b), (d), (f) of "is" over "slice wd" on a 550-MHz CPU when localizing 1 point with M_1.

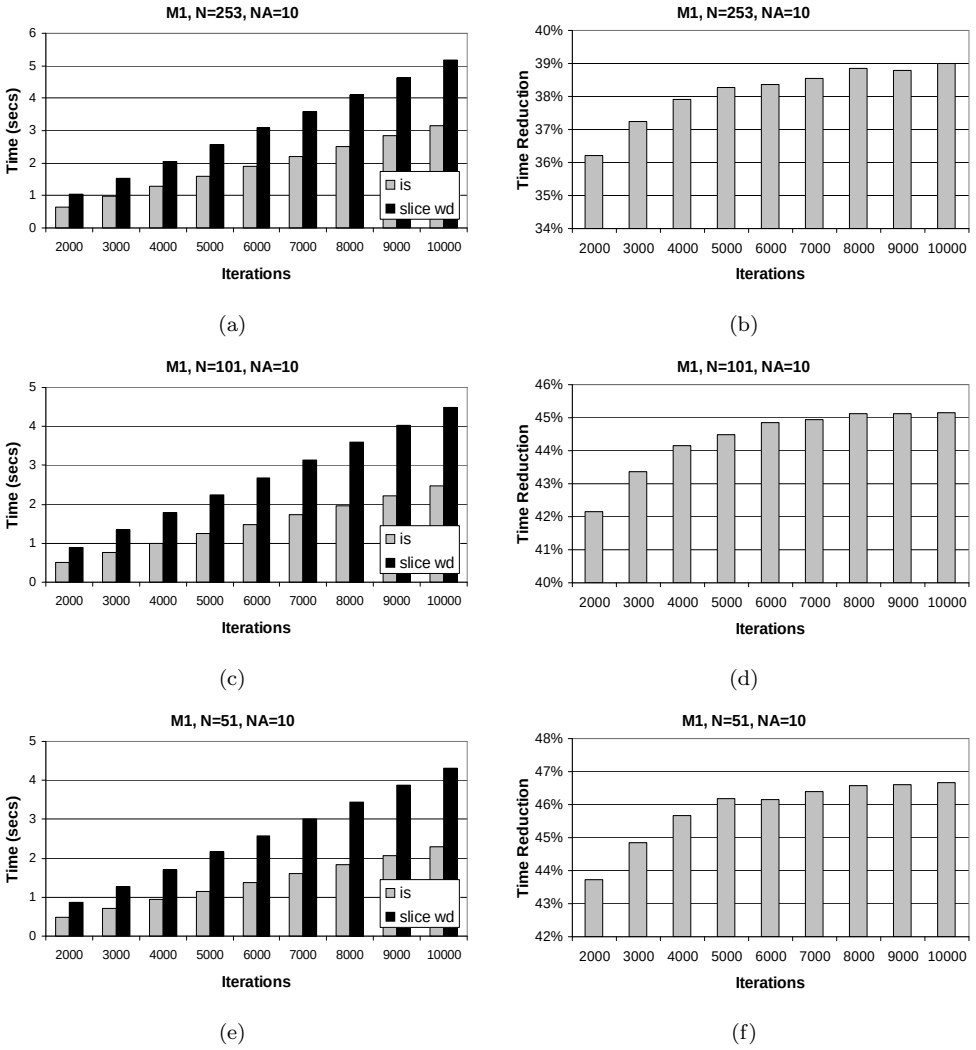


Figure 2.14. Absolute time (a), (c), (e) of importance sampling (is) and whole domain sampling (slice wd) and percentage of time reduction (b), (d), (f) of "is" over "slice wd" on a 550-MHz CPU when localizing 10 points with M_1.

Chapter 3

Parallel Algorithms for Bayesian Indoor Positioning Systems

3.1 Introduction

In the previous chapter we implemented several Bayesian inference methods for Bayesian networks M_1, M_2, M_3, A_1 using Markov Chain Monte Carlo (MCMC) walks [47,53,54, 67,73]. In MCMC methods, each instantiation of the network (i.e. with the variables having values) forms a state in a Markov chain. As we saw, the MCMC is a sampling procedure which generates a successive state, with new values for the variables. This procedure is, in effect, generating a new node in a Markov chain, where each node is an instance of the network. We call the process of drawing a random sample for all the variables in the network an *iteration*; this corresponds to generating one state in the Markov chain.

Although the MCMC methods proposed in the previous chapter are both computationally efficient and provide quick convergence, they can still take a lot of time when many devices are localized simultaneously. For instance, they can take more than half a minute on a 2.4-GHz machine to simultaneously localize 200 devices. We are thus motivated to explore parallel computing methods for this problem.

In this chapter we describe two parallelization strategies. The first, inter-chain parallelism, runs multiple independent chains on different processors. The observed values are then aggregated to form the probability distributions of the variables. The second approach, intra-chain parallelism, divides the work of a single chain between processors. The division in effect partitions the formation of a single Markov state (i.e. an iteration) across processors.

We implemented our two approaches using Berkeley Unified Parallel C (BUPC) [69], which is a parallel language that adopts a Single Program Multiple Data (SPMD) model using a global address space (GAS). Specifically, we applied these two approaches to the most efficient MCMC inference method described in the previous chapter, which was called "whole domain sampling". We found UPC an effective language for describing the data layout needed by our algorithms.

We evaluated our implementation on three platforms: a 16-node symmetric multiprocessor (SMP), a 4-node cluster comprising of quad processors, and a 16 single-CPU-per-node cluster. Our results show that intra-chain parallelism gives speedups of 12 on 16 processors on the first two platforms, when the MCMC method has performed a small number of iterations (at most 10,000). On the other hand, inter-chain parallelism requires many more iterations (at least 40,000) on these two platforms in order to achieve speedups of 12 and higher. We found the 16-way cluster, which could only run the inter-chain algorithm, required at least 60,000 iterations to achieve a speedup of 12.

For the Bayesian networks we study here, it was shown in the previous chapter that only a small number of iterations (at most 10,000) is required in order to get good localization results. Hence, intra-chain parallelism is a good candidate for applying parallelism to them, when run on platforms such as the first two. However, load balancing is harder to achieve in the intra-chain parallelism, since it requires to split evenly the computational cost of a single iteration of the MCMC method among processors, which can be non-trivial. Additionally, it necessitates communication for every iteration. In inter-chain parallelism it is easier to achieve load balancing, since it only requires to divide evenly the number of iterations of the MCMC method among processors. For Bayesian networks that need many iterations in order to give good results, inter-chain parallelism is the algorithm to choose, as for large number of iterations it can outperform intra-chain parallelism.

In order to analyze and predict the performance of our two algorithms we use the LogP [20] model and its extension for large messages, LogGP [7]. The predictions of the models are compared to the experimentally gathered data from the first two platforms.

The comparisons show that the predictions are within 5% of the observed execution time for the inter-chain parallelism, whereas for the intra-chain they are 7%-25% less than the actual time. The reason for the latter discrepancy is that there is load imbalance in the intra-chain parallelism that the models fail to capture. Nevertheless, the models can give us a good indication of the performance of our algorithms on different platforms.

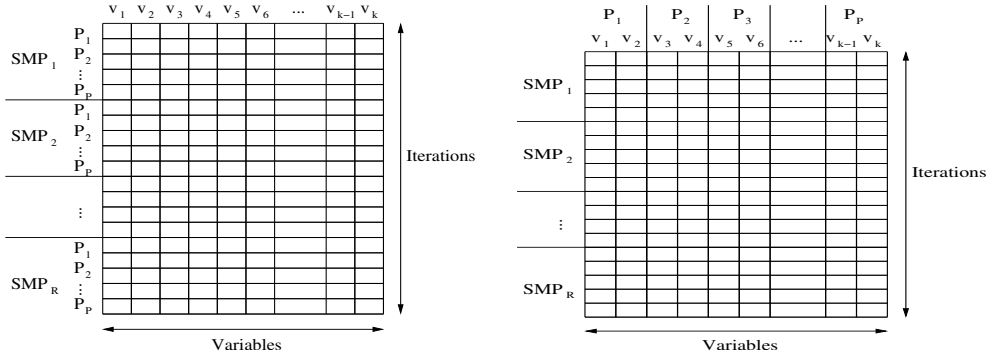
The rest of this chapter is organized as follows. In Section 3.2 we describe the two parallel algorithms applied to the Bayesian inference of our networks. In Section 3.3 we present our speedup results on different platforms, and in Section 3.4 we analyze the performance of our algorithms using the LogP/LogGP models. Section 3.5 presents related work. Finally, Section 3.6 summarizes our results.

3.2 Parallel Algorithms

Below, we describe two parallel algorithms we apply to the MCMC process that infers values for the unknowns of the networks presented in Section 2.4. In both cases we assume we apply parallelism to the generation of a single Markov chain that requires B burnin iterations and A additional, and hence, the length of the chain is $B + A$.

3.2.1 Inter-Chain Parallelism

Inter-chain parallelism divides equally the additional iterations of the chain among all Q processors that are available. In the case of a P-way SMP, $Q = P$, whereas in a cluster of R P-way machines, $Q = P * R$. Thus, it runs Q chains in parallel, each one using a different starting seed for the Markov walk. Different seeds ensure that each Markov walk will follow a different trajectory. Moreover, every chain needs to have B burnin iterations in order to ensure that the values chosen in the additional iterations are from the stationary distribution. Hence, the length of each chain is $B + A/Q$. Figure 3.1(a) shows a pictorial representation of how the algorithm divides the computational cost. Each row corresponds to a single iteration during which an MCMC method updates variables v_1, v_2, ... ,v_k, whereas each column corresponds to the values generated by the method for some variable after a number of iterations have been performed. Essentially,



(a) Inter-chain parallelism (b) Intra-chain parallelism

Figure 3.1. Sampling load distribution by our two parallel algorithms.

the algorithm "slices" horizontally the number of additional iterations, forcing though each chain to consist of B burnin iterations.

In order to generate statistics such as the median or the 95% interval, the samples of all variables generated in the additional iterations need to be sorted. There are several options that this can be done; through the use of some parallel sort (e.g. radix sort [24]) or processors can exchange the samples of the variables they control so that all samples of a given variable are collected by a single processor that can in turn sort them locally. After testing the performance of the two options, we decided that the latter is faster, as the number of samples to be sorted per processor for our networks do not justify the use of a parallel sort. So, if there are k variables in a Bayesian network, processors divide them equally among them and each one sorts k/Q variables. Once sorting is done, statistics are gathered by a single processor that outputs the results to a file.

The algorithm manages to easily balance the sampling load on all processors, as each processor updates the same number of variables and generates chains of the same length. Also, running the algorithm on a cluster of SMPs is trivial, since all processors on the cluster are treated equally. The disadvantage of the algorithm though is that all processors need to pay an overhead of B burnin iterations.

3.2.2 Intra-Chain Parallelism

Intra-chain parallelism ensures that each SMP generates only one Markov chain regardless of the number of processors in it. Within an SMP the algorithm distributes the variables to be updated to the processors of the SMP, and, therefore, each processor updates only a subset of the variables of the Bayesian network. At the end of each iteration, every processor gathers the new values of the variables generated by the other processors of the SMP. The reason is that a processor requires the values of other variables in the network during the update process, and in order for the Markov chain to evolve, it is important that each processor has the latest value of the other variables.

For a single SMP, the algorithm generates only one chain of length $B + A$. For a cluster of R SMPs, the additional iterations are divided equally among the SMPs, so that there are R Markov chains that run in parallel, each of length $B + A/R$. Every SMP uses a different seed to evolve the Markov walk. The assignment of variables to processors is identical on all SMPs. Figure 3.1(b) depicts how the algorithm divides the computational cost among the SMPs. In particular, it "slices" the cost vertically within an SMP and horizontally across the SMPs of a cluster. It is interesting to note that in the case of a cluster of single-processor SMPs the intra-chain algorithm becomes inter-chain. There is no vertical "slicing" of the computational cost within an SMP; the single processor of an SMP updates all variables of the Bayesian network.

To generate statistics, the samples of all processors need to be sorted. In the case of only one SMP, each processor can sort locally the samples of the variables it has been assigned to, as it has the samples of all the additional iterations of its variables. In the case of a cluster of SMPs, processors that have been assigned the same variables in the cluster can split the variables they have been assigned to amongst them, exchange samples, and sort them locally. Applying a parallel sort (e.g. radix sort) is harder when compared to inter-chain parallelism, as parallel sorts assume that all processors have samples for all variables. For intra-chain parallelism this would require extra cost to distribute the samples so that all processors have samples of all variables. Moreover, as in the inter-chain case, the algorithm does not generate enough samples that would

justify the use of a parallel sort. Consequently, we chose to have processors sort samples locally on both SMP and cluster. Once statistics are estimated, they are collected by a single processor that outputs them to a file.

Unlike inter-chain parallelism, the cost of the B burnin iterations is paid only once within each SMP. On the other hand, the algorithm necessitates an all-to-all exchange of values within an SMP at the end of each iteration. Furthermore, balancing the sampling load on the processors of a given SMP can be a non-trivial task, as the computational cost of updating a variable varies among the variables of a Bayesian network. The computational cost depends on the sampling method (e.g. slice sampling, conjugate sampling [47, 53, 54, 73]) used to update a variable, as well as on the size of the training data given as input to the Bayesian network. A load imbalance will affect processors when they communicate at the end of each iteration, as some processors might have to wait for others to finish updating their variables.

Special consideration had to be taken when we implemented the intra-chain parallelism for our Bayesian networks. Assigning random subsets of the variables to different processors resulted in localization results that deviated from the results of a chain generated by a single processor that updates all variables together. The reason is that the MCMC method we use for inference and we call "whole domain sampling" (see 2.3.1.2) is a Gibbs sampling method. Applying parallelism to such a method by distributing variables to different processors requires an algorithm like the one described in [56]. As explained in Section 3.5, these kind of algorithms do not map efficiently onto different interconnection structures, and thus we follow the vertical "slicing" described earlier. Nevertheless, we realized that the intra-chain algorithm can give results similar to a single chain, when applied to our Bayesian networks, by ensuring that certain groups of variables are assigned to the same processor. Specifically, variables b_{i0}, b_{i1} for the same i in networks M_1, M_2, and b_{i0}, b_{i1}, b_{i2}, b_{i3} for the same i in M_3, A_1, and X, Y of a specific location have to be on the same processor.

3.3 Experimental Results

We have implemented the two algorithms described in Section 3.2 using the Berkeley UPC (BUPC) [69] parallel language, which is an extension of C and provides a Global Address Space (GAS) model. Programmers have full control over how their data is laid out across processors, and can access this data via standard mechanisms such as pointer dereferences, array indexing, or memcpy-style bulk copy calls. In our work we used the 2.4.0 version of the BUPC compiler.

The algorithms were tested on three platforms. The first is an SMP with 16 processors running Linux. Each processor has a 2.4-GHz clock speed and 2 GBs of memory. The second is a cluster of 4 Pentium 3 machines, each one having a quad processor and running Linux. Each processor has a 550-MHz clock speed and 250 MBs of memory. The nodes of the cluster are connected by a 100-Mbps switch. The third is a cluster of 16 Pentium 4 machines with a dual processor, running Linux and connected by a 100-Mbps switch. Each processor has a 3.2-GHz clock speed and 500 MBs of memory. In all our experiments we used only one of the two processors of every node in the latter cluster. The training data sets we used for our Bayesian networks are the ones presented in Section 2.5. Also, we followed the leave-n-out method, meaning that n points were chosen from the training set to be localized.

The results shown next were generated by applying both algorithms to a specific MCMC inference method, "whole domain sampling", that was shown in Section 2.5.3 to be the fastest in terms of time and convergence for the networks that we study here. However, they can be applied to other MCMC methods too. For the training data and the MCMC method we use, we figured out that we needed 800 burnin iterations. All our results are the average of 30 runs; in every run a different set of point(s) is chosen to be localized. In the graphs we describe next, N is the number of training points out of which we localize NA points. Also, $Gran$ is the granularity (see Section 2.4) used to take signal strength measurements in the A_1 network.

The metric we have chosen to measure the performance of the algorithms is speedup,

which is defined as follows:

$$Speedup = \frac{T_{serial}}{T_{parallel}(P)} \tag{3.1}$$

where T_{serial} is the running time of the serial algorithm on one processor, and $T_{parallel}(P)$ is the running time of the parallel version of the algorithm on P processors. We use the time needed by a single-threaded BUPC program as T_{serial}, since we have managed to make it run as fast as the single-threaded C solvers proposed in Section 2.5. In BUPC there are two ways to run a single-threaded program by using the following options at compile time: (a) -nopthreads, (b) -pthreads=1. However, in the latter case the compiler adds some overhead to the program rendering it a little slower than in case (a). So, since strictly speaking T_{serial} is supposed to be as small as possible in equation 3.1, we measure T_{serial} by compiling our algorithms with the -nopthreads option. There are different possibilities for measuring time in these cases, such as user time, wall clock time. We have chosen to measure wall clock time, that is the elapsed time between the start and the end of a run. Wall clock time includes the cost of negative effects like communication overhead, idle time caused by imbalance and synchronization. Finally, since speedups depend on the amount of computational cost that is distributed to processors, we present results with increasing values of NA and decreasing of $Gran$. By increasing the value of NA, we are able to see the benefits from the two types of parallelism when we localize many devices at the same time.

3.3.1 Inter-Chain Results

Figure 3.2 shows the speedups we get for some of our Bayesian networks using 16 threads on the 16-node SMP and the 4-machine cluster. The number of iterations in the graphs is the total number of iterations run by all threads together. As can be seen, both platforms give approximately the same speedup with the SMP offering slightly higher. The reason for the small improvement is that, in the cluster, threads need to use the Ethernet network to exchange the variable samples in order to sort them and produce statistics. Our results show that the inter-chain parallelism does not offer good

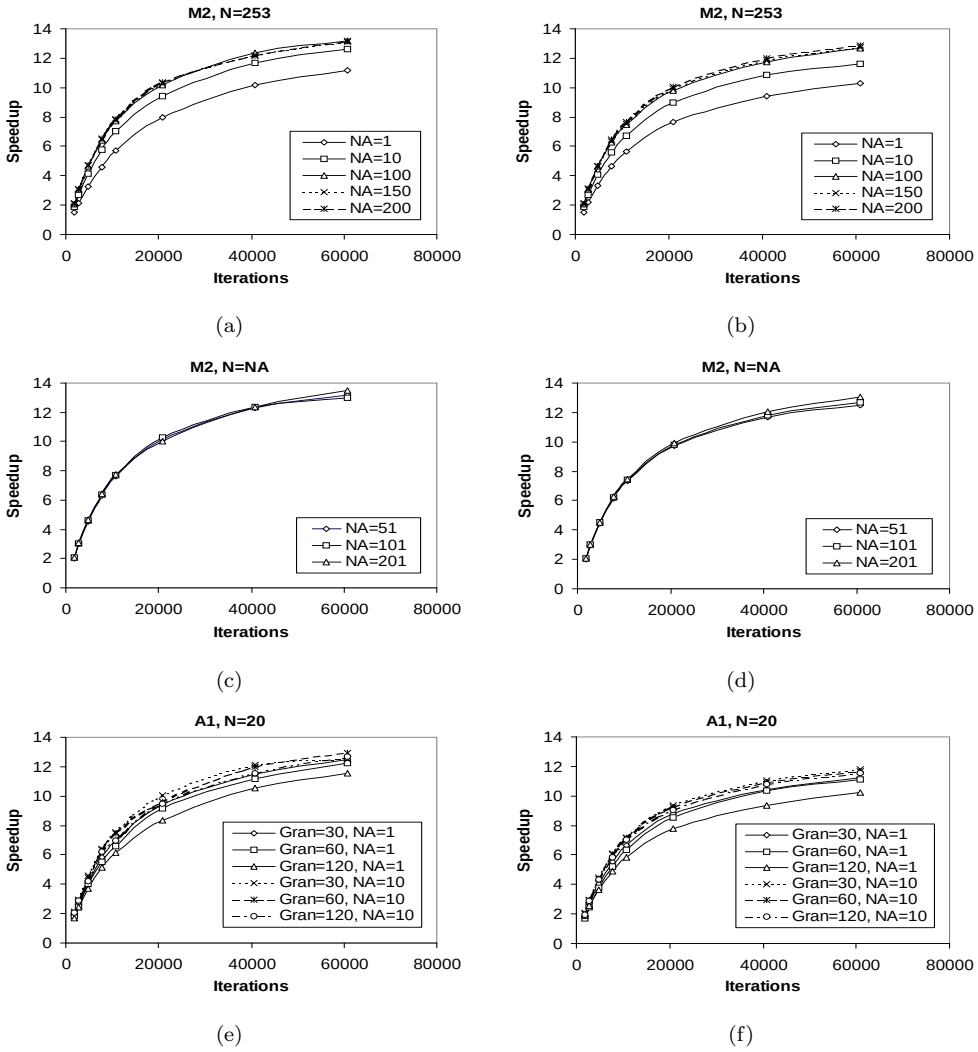


Figure 3.2. Speedups of the inter-chain parallelism using 16 threads (one per processor) on a 16-node SMP (a), (c), (e) and on a cluster of 4 quad-processor machines (b), (d), (f).

speedups as the number of iterations scale down to 2,000. The reason is that, although the computational cost of the additional iterations is divided among all threads, the cost of the burnin is not; it is a fixed overhead that all threads need to pay. We see the algorithm needs at least 40,000 iterations in order to pay off the burnin computational cost of 800 iterations and give speedups of 12. However, as was shown in Section 2.5.3, for our networks we do not need more than 10,000 iterations to get good localization results. Figure 3.3 presents the speedups for the other networks (M_1, M_3) as well as when we use 8 threads (2 quad-processor SMPs in 3.3(f)). The figure shows that, when

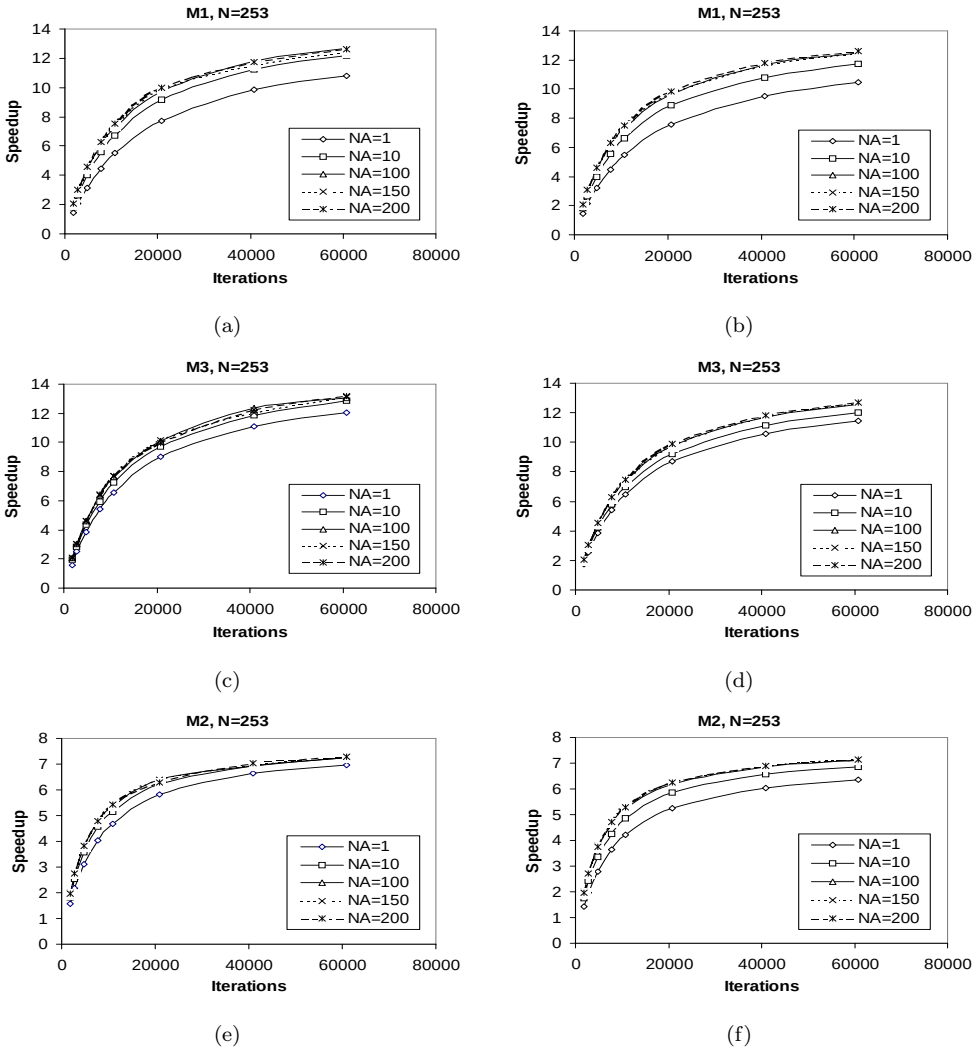


Figure 3.3. Speedups of the inter-chain parallelism using 16 threads (a, b, c, d) and 8 threads (e, f) (one per processor) on a 16-node SMP (a), (c), (e) and on a cluster of 4 quad-processor machines (b), (d), (f).

using 16 threads, networks M_1, M_3 have performance similar to the ones shown in graphs 3.2(a), 3.2(b). With 8 threads the algorithm gives a speedup of 7 after running 40,000 iterations on M_2 (we get similar results for the other networks).

Finally, Figure 3.4 depicts the relative accuracy offered by a single-threaded solver and the parallel version of the solver when the inter-chain scheme is used. As was explained in Section 2.5.3, we call *relative accuracy* the Euclidean distance of the results of our solver compared to the ones from WinBugs after running WinBugs for 10,000 iterations as burnin, 100,000 additional and having the over relax option set (see [66]).

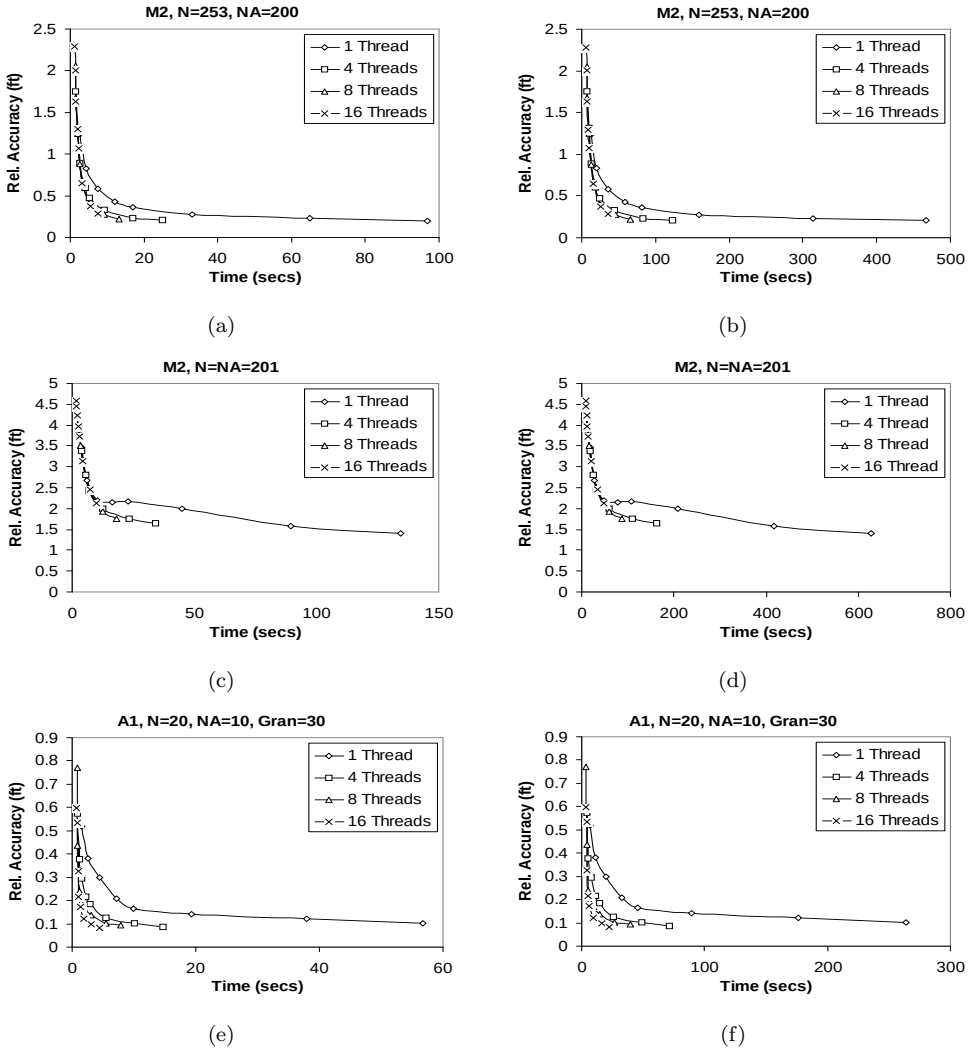


Figure 3.4. Relative accuracy vs. time of the inter-chain parallelism on a 16-node SMP (a), (c), (e) and on a cluster of 4 quad-processor machines (b), (d), (f).

In all graphs of the figure, the last point in every line corresponds to 60,000 iterations. We see that the accuracy of the inter-chain parallelism is qualitatively similar to the one of a single-threaded solver. Therefore, we do not loose accuracy with respect to a single-threaded solver and at the same time we get the benefits of parallelism.

3.3.2 Intra-Chain Results

Figure 3.5 depicts the speedups of the intra-chain parallelism when running 16 threads on the 16-node SMP and the 4-machine cluster. We see that on the 16-node SMP

speedups are very good (close to 12) even when the number of iterations scale down to 2,000. On the cluster however, speedups are low for a small number of iterations (2,000 to 5,000), but after that they reach the level of speedups offered by the 16-node SMP. The reason is that on the 16-node SMP there is only one Markov chain and hence the cost of the burnin iterations is paid only once. In the case of the cluster, since it consists of 4 machines, the algorithm ran 4 chains in parallel and each one had to pay the overhead of the burnin. The graphs show that after 5,000 iterations the algorithm pays off the burnin cost giving higher speedups. The number of iterations shown in graphs 3.5(b), 3.5(d), 3.5(f) is the total number of iterations run by all 4 chains together.

An important issue in this algorithm is balancing the load within an SMP. This requires assigning variables to the processors in such a way so that in every iteration the total time spent on each processor for updating variables is roughly equal. In the current study, we used the computational cost required to update each variable of the network to assign variables to processors at compile time under the restrictions explained in Section 3.2.2. Since load balancing can be achieved more easily on 4 processors (each machine in the cluster is a quad Pentium 3) rather than on 16, when the number of iterations increase, the cluster can achieve better speedups than the 16-node SMP. This can be clearly seen for M_2 when $N = NA = 201$ (graphs 3.5(c), 3.5(d)), and for A_1 when $Gran = 30$, $NA = 10$ (graphs 3.5(e), 3.5(f)) for 10,000 iterations, where the cluster starts outperforming the SMP.

Graph 3.5(c) shows that when localizing all the points in the training set ($N = NA$) we get speedups of up to 10, whereas in 3.5(a) the same network (M_2) can give better speedups (up to 12) when localizing $NA < N$ points. As was shown in Section 2.5.4, in the $NA = N$ case we need to bound the b_{i0} parameters for all i, resulting in a high computational cost when updating these variables. As a consequence, we were not able to achieve good load balancing in this case on the 16 processors of the SMP. Also, graphs 3.5(e), 3.5(f) show that we do not achieve as high speedups for A_1 as for the other networks. This is due to the fact that the training data for this network consists of only 20 points and hence there is not enough workload to distribute to the

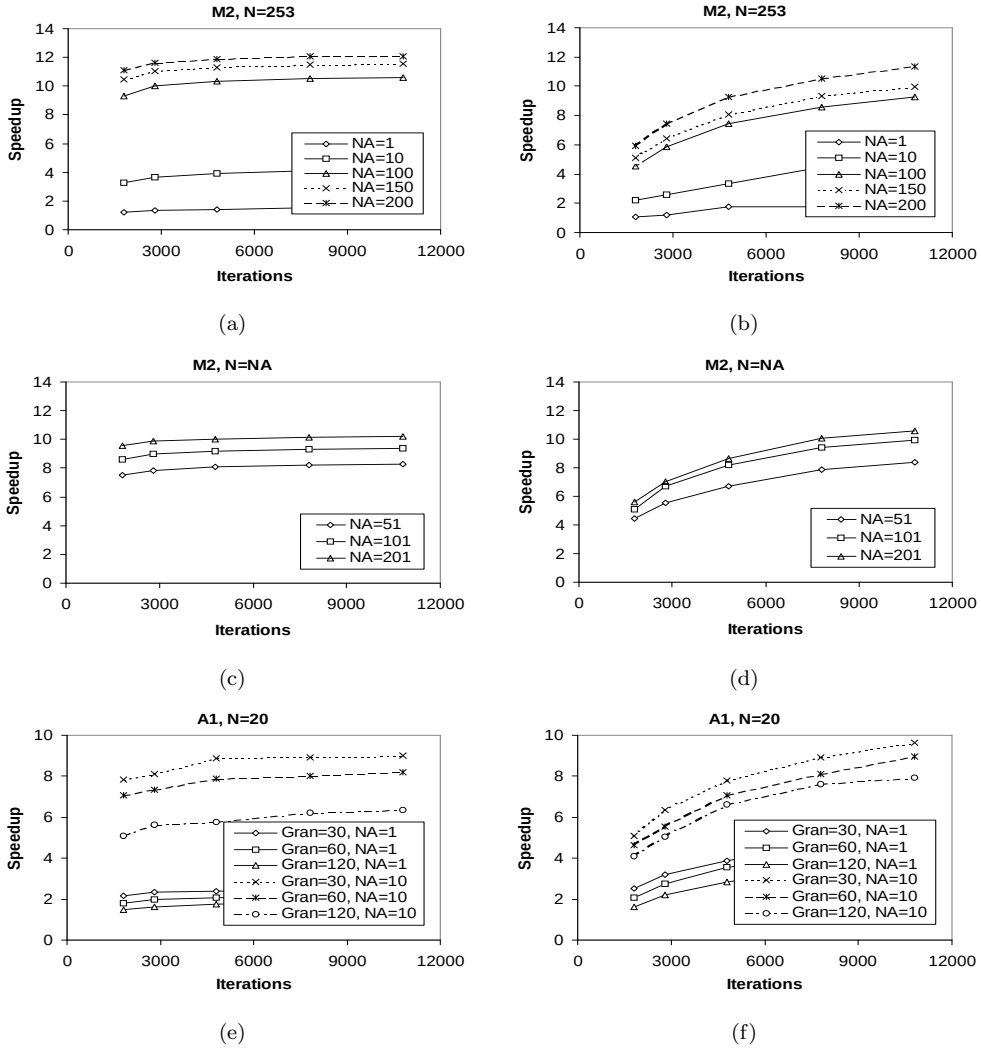


Figure 3.5. Speedups of the intra-chain parallelism using 16 threads (one per processor) on a 16-node SMP (a), (c), (e) and on a cluster of 4 quad-processor machines (b), (d), (f).

processors that will offset the communication required at the end of each iteration and thus achieve speedups comparable to the other networks. Moreover, the difference in the speedup achieved between small NAs (1, 10) and large NAs (>=100) in graph 3.5(a) is larger when compared to the corresponding in 3.2(a). The discrepancy is attributed again to the communication that intra-chain necessitates at the end of each iteration. The speedups of networks M_1, M_3 are presented in Figure 3.6, which show that their perfromance is similar to the ones shown in graphs 3.5(a), 3.5(b). Also, the latter figure shows the speedups of M_2 when using 8 threads (2 quad-processor SMPs

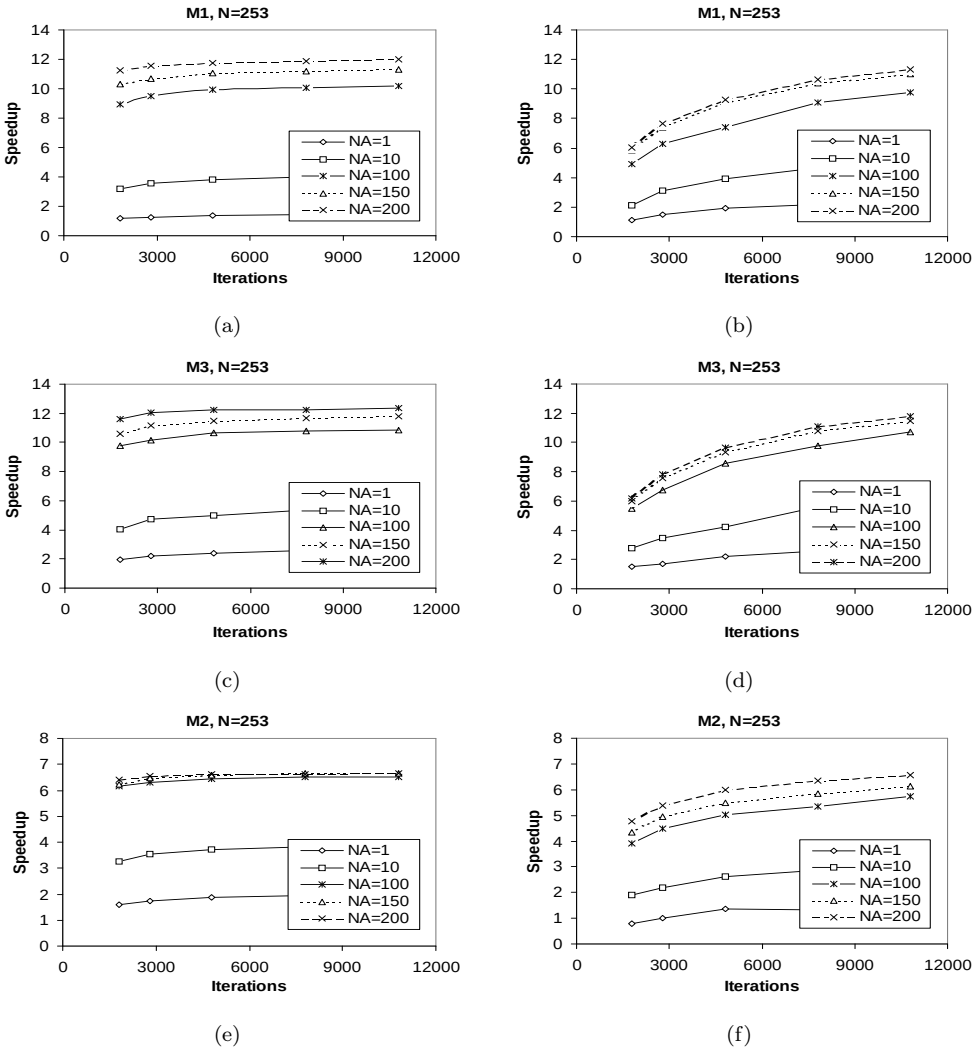


Figure 3.6. Speedups of the intra-chain parallelism using 16 threads (a, b, c, d) and 8 threads (e, f) (one per processor) on a 16-node SMP (a), (c), (e) and on a cluster of 4 quad-processor machines (b), (d), (f).

in 3.6(f)).

Figures 3.7, 3.8 present the relative accuracy of single-threaded solvers and the intra-chain solvers. In these figures, the last point in every line corresponds to 10,000 iterations. We see that the intra-chain scheme provides us with accuracy that is similar to the accuracy of a single-threaded solver, which proves that the way the algorithm assigns variables to processors indeed helps so that the offered accuracy is not compromised.

As was mentioned in Section 3.2.2, the algorithm behaves like the inter-chain when

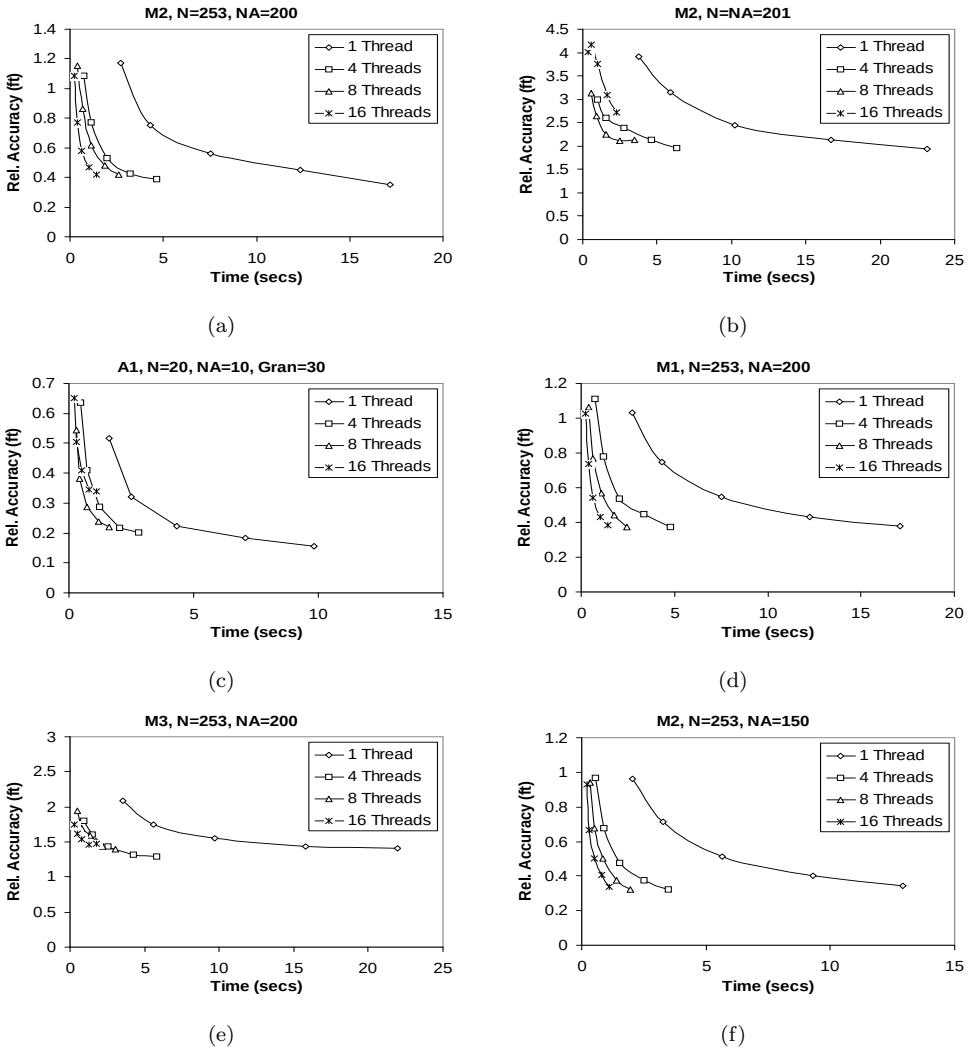


Figure 3.7. Relative accuracy vs. time of the intra-chain parallelism on a 16-node SMP.

run on a cluster with 1 thread per SMP. Figure 3.9 presents its performance when run on our 16-node cluster. The curves are similar to the ones depicted in Figure 3.2, but the speedups are smaller when compared to that figure. The reason for the decrease is that each thread on the 16-node cluster uses the Ethernet network to exchange samples with other threads, whereas in Figure 3.2 there is communication within an SMP on the 4-machine cluster apart from across the Ethernet during the exchange, and on the 16-node SMP there is no usage of Ethernet at all. So, when compared to the 16-node SMP and 4-machine cluster (with 16 threads), speedups are smaller by 1.8 and 1.5 respectively for M_2 when $NA = 200$, 1.3 and 0.9 for M_2 when $N = NA = 201$, 2.0

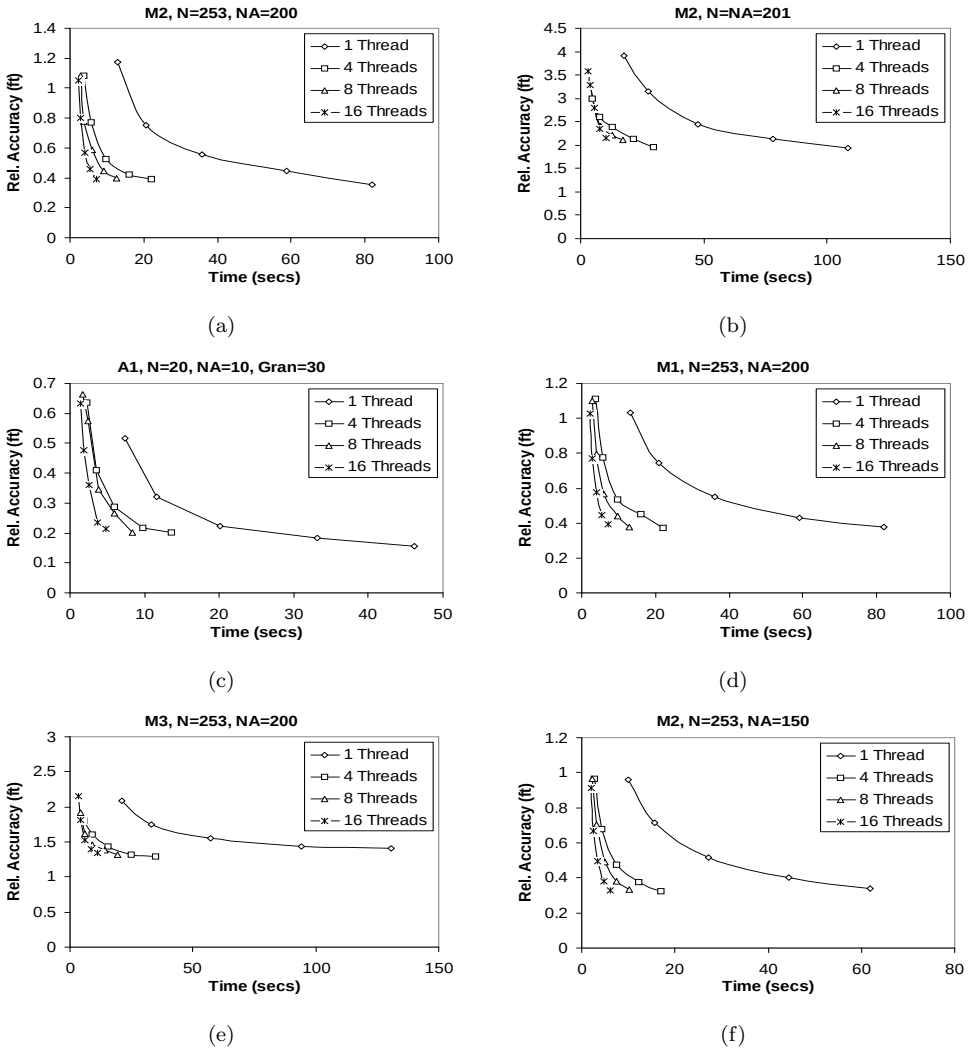


Figure 3.8. Relative accuracy vs. time of the intra-chain parallelism on a cluster of 4 quad-processor machines.

and 1.3 for A_1 when $Gran = 30$ and $NA = 10$. For smaller NAs, the decrease is higher, because there is not enough computation to offset the increased communication overhead of the threads. Graphs 3.9(a), 3.9(b), 3.9(d), 3.9(e) show that for M_1, M_2, M_3 we need at least 60,000 iterations in order to get a speedup of 12, whereas for A_1 (graph 3.9(c)) we need even more. Also, unlike graphs 3.3(e), 3.3(f) that show that on a 16-node SMP and two quad-processor machines we can get speedups of 7 with 40,000 iteration, graph 3.9(f) shows that we need more than 60,000 iterations to achieve the same speedup on a cluster of 8 single-processor machines.

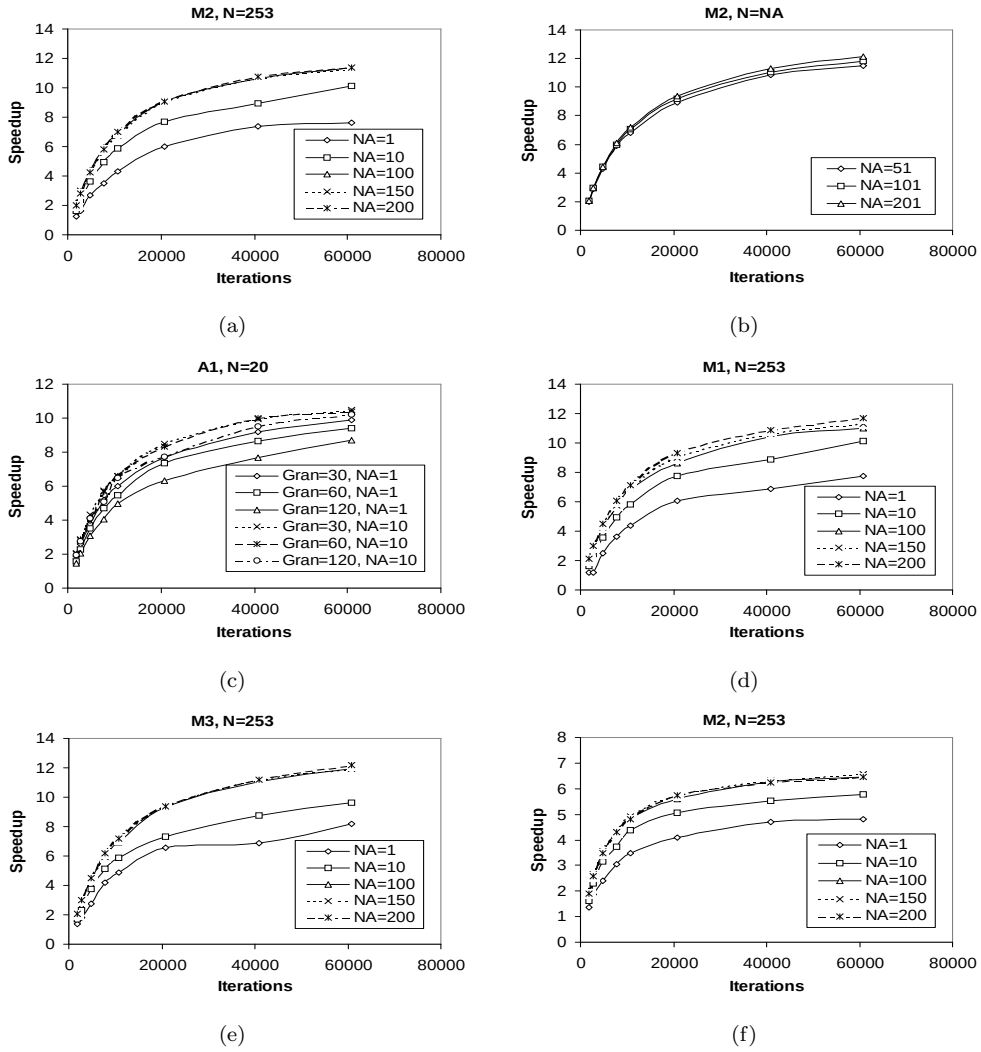


Figure 3.9. Speedups of the intra-chain parallelism using 16 threads (a, b, c, d, e) and 8 threads (f) (one per processor) on a cluster of 16 machines. The algorithm is essentially inter-chain on the cluster.

3.4 LogGP Analysis

In this section we describe a way of modeling the behavior of our algorithms so that we understand how much time each phase of the algorithms requires as well as we can predict their performance on several platforms.

LogP/LogGP		16-node SMP	Cluster SMP	Cluster Net
Latency	L	0.39870 μsec	0.51790 μsec	72.1308 μsec
Overhead	o	0.04315 μsec	0.25655 μsec	24.2436 μsec
Gap	g	0.10330 μsec	0.62310 μsec	50.2202 μsec
Gap-per-byte	G	0.00263 $\frac{\mu sec}{byte}$	0.00977 $\frac{\mu sec}{byte}$	0.33447 $\frac{\mu sec}{byte}$

Table 3.1. LogP/LogGP model parameters on a 16-node SMP and a cluster of 4 quad-processor machines.

3.4.1 Modeling Communication and Computation

In order to describe the network performance of our algorithms we use the LogP [20] model, which is a well-established approach to modeling small messages, and its extension, LogGP [7], for large messages. To analyze the performance of our algorithms on a cluster, we use two types of LogGP parameters; one for communication within an SMP of the cluster and one for communication between SMPs. To see why, remember that in the intra-chain algorithm on the cluster, all processors of the SMP need to exchange the newly-generated values of the variables they control at the end of each iteration. For this kind of communication we need the LogGP parameters of the SMP. On the other hand, in both algorithms, when the processors of the cluster need to exchange samples in order to sort them, we need to use the LogGP parameters of the network that connects the SMPs. Table 3.1 summarizes the parameter values of the LogP/LogGP model on two platforms. Specifically, the second solumn presents the parameter values for communication within the 16-node SMP, the third column, for communication within one of the quad-processor SMPs, and the fourth column, for communication using the network that connects the quad-processor machines. The parameters were measured as in [13,21], by writing BUPC benchmark programs. The value of G shown in the table is the worst possible G measured from the benchmarks.

Local computation is captured by measuring time per variable per processor. Table 3.2 shows the formulas that estimate the sampling time of one iteration in our Bayesian networks. In this table, d is the number of access points (see Section 2.4), and NA the number of points we try to localize. In the formula that estimates the sampling time of network M_1, $t_{b_{i01}}$ represents the time needed to update either variable b_{i0} or b_{i1}, while t_{τ_i}, t_X, t_Y the time needed for variables τ_i, X, Y respectively (see Figure 2.3).

Network	Time
M_1	$\sum_{j=1}^{2*d} t_{b_{i01}} + \sum_{j=1}^{d} t_{\tau_i} + \sum_{j=1}^{NA} t_X + \sum_{j=1}^{NA} t_Y$
M_2 $(NA < N)$	$\sum_{j=1}^{2} t_{b_{01}} + \sum_{j=1}^{2} t_{\tau_{01}} + \sum_{j=1}^{2*d} t_{b_{i01}} + \sum_{j=1}^{d} t_{\tau_i} + \sum_{j=1}^{NA} t_X + \sum_{j=1}^{NA} t_Y$
M_2 $(NA = N)$	$\sum_{j=1}^{2} t_{b_{01}} + \sum_{j=1}^{2} t_{\tau_{01}} + \sum_{j=1}^{d} t_{b_{i0}} + \sum_{j=1}^{d} t_{b_{i1}} + \sum_{j=1}^{d} t_{\tau_i} + \sum_{j=1}^{NA} t_X + \sum_{j=1}^{NA} t_Y$
M_3	$\sum_{j=1}^{4} t_{b_{03}} + \sum_{j=1}^{4} t_{\tau_{03}} + \sum_{j=1}^{4*d} t_{b_{i03}} + \sum_{j=1}^{d} t_{\tau_i} + \sum_{j=1}^{NA} t_X + \sum_{j=1}^{NA} t_Y$
A_1	$\sum_{j=1}^{4*d} t_{b_{i01}} + \sum_{j=1}^{d} t_{\tau_i} + \sum_{j=1}^{360/Gran*d*NA} t_\alpha + \sum_{j=1}^{NA} t_X + \sum_{j=1}^{NA} t_Y$

Table 3.2. Sampling time of one iteration.

The formulas for the other networks are explained analogously. Table 3.3 presents the local computation rates in our networks for the 16-node SMP. When possible, we use a constant value for the time per variable; however, in the case of $t_{b_{i01}}/t_{b_{i03}}$, t_{τ_i}, $t_{b_{i0}}$, $t_{b_{i1}}$, we use time per variable per training size (N) per number of points to localize (NA), because computation is dependent on these parameters. Additionally, there is dependency on granularity in A_1. Table 3.4 summarizes the values of the local computation rates for the cluster of 4 quad-processor machines.

Moreover, Table 3.5 gives the time required by the algorithms on two platforms as this is estimated by the LogGP models. The two platforms are a P-way SMP and a cluster of R P-way machines. The formulas for $T_{intra,smp}$, $T_{intra,clu}$, $T_{inter,smp}$, $T_{inter,clu}$ capture the phases of the algorithms that take most of the time in the algorithms. Specifically, these phases are the sampling process, generation of statistics (t_{gen_stats}), moving samples to other processors so that they can be sorted ($t_{move\_samples}$), and gathering values from all processors within an SMP at the end of each iteration in the intra-chain parallelism (t_{gather_values}). We did not include in the formulas time needed by the processors to read training data from input files, gather and write statistics to output files, initialization of data structures and random generators. However, these phases take very little time when compared to the total execution time of the algorithms. We implemented the gathering-values phase (t_{gather_values}) by using bulk memcpy calls and a tournament barrier [34], since BUPC does not provide us with

Variable	$\mathbf{M_1}$	$\mathbf{M_2}$ ($\mathbf{NA < N}$)	$\mathbf{M_2}$ ($\mathbf{NA = N}$)
$t_{b_{i01}}/t_{b_{i03}}$	0.00045*N*NA	0.00045*N*NA	
t_{τ_i}	0.00044*N*NA	0.00045*N*NA	0.00044*N*NA
t_X	3.14835	3.13906	3.38464
t_Y	2.61147	2.61531	2.80526
t_{gen_stats}	0.13406	0.13401	0.13370
$t_{b_{01}}/t_{b_{03}}$		0.00125	0.00157
$t_{\tau_{01}}/t_{\tau_{03}}$		0.15044	0.15048
$t_{b_{i0}}$			0.00255*N*NA
$t_{b_{i1}}$			0.00044*N*NA
t_α			
Variable	$\mathbf{M_3}$	$\mathbf{A_1}$	
$t_{b_{i01}}/t_{b_{i03}}$	0.00049*N*NA	0.00442*N*NA*360/Gran	
t_{τ_i}	0.00048*N*NA	0.00455*N*NA*360/Gran	
t_X	3.48992	20.4175	
t_Y	3.08435	15.4654	
t_{gen_stats}	0.13379	0.13468	
$t_{b_{01}}/t_{b_{03}}$	0.25061		
$t_{\tau_{01}}/t_{\tau_{03}}$	0.35645		
$t_{b_{i0}}$			
$t_{b_{i1}}$			
t_α		0.57787	

Table 3.3. Local computation rates (in μsecs) for the 16-node SMP.

collective operations, such as all-gather-all, that apply only within an SMP of a cluster when threads run on all SMPs. The value of t_{gen_stats} (Tables 3.3, 3.4) corresponds to a measured time per sample needed to generate statistics. This includes the time of local quick sort, finding median, average, 2.5% and 97.5% interval, and standard deviation.

In the formulas we assume the algorithms try to parallelize the generation of a Markov chain with length *total_iter* that consists of *burnin* burnin iterations and *additional_iter* additional. Furthermore, the number of variables assigned by the intra-chain algorithm to processors within an SMP as well as the number of variables sorted by each processor in the inter-chain algorithm are approximated by $network\_vars/P$, where *network_vars* is the total number of variables in the Bayesian network. The approximation is good for the inter-chain parallelism as processors are assigned equal number of variables to sort (see Section 3.2.1). On the other hand, it is rougher for the intra-chain, as processors are assigned at compile time different number of variables to update based on the computational cost required to generate a new sample for a variable. As was explained in Section 3.3.2 the assignment tries to keep the load on

Variable	$\mathbf{M_1}$	$\mathbf{M_2}$ **(NA < N)**	$\mathbf{M_2}$ **(NA = N)**
$t_{b_{i01}}/t_{b_{i03}}$	0.00196*N*NA	0.00196*N*NA	
t_{τ_i}	0.00194*N*NA	0.00193*N*NA	0.00192*N*NA
t_X	15.6335	15.3928	16.0084
t_Y	13.1049	12.8384	13.2591
t_{gen_stats}	0.73306	0.73324	0.73306
$t_{b_{01}}/t_{b_{03}}$		1.51180	1.51138
$t_{\tau_{01}}/t_{\tau_{03}}$		2.18680	2.17861
$t_{b_{i0}}$			0.01083*N*NA
$t_{b_{i1}}$			0.00195*N*NA
t_α			
Variable	$\mathbf{M_3}$	$\mathbf{A_1}$	
$t_{b_{i01}}/t_{b_{i03}}$	0.00286*N*NA	0.01704*N*NA*360/Gran	
t_{τ_i}	0.00276*N*NA	0.01577*N*NA*360/Gran	
t_X	21.0373	92.5862	
t_Y	18.7017	69.2865	
t_{gen_stats}	0.73370	0.73454	
$t_{b_{01}}/t_{b_{03}}$	1.51696		
$t_{\tau_{01}}/t_{\tau_{03}}$	2.29166		
$t_{b_{i0}}$			
$t_{b_{i1}}$			
t_α		2.97796	

Table 3.4. Local computation rates (in μsecs) for the cluster of 4 quad-processor machines.

all processors within an SMP equally balanced. Additionally, since variables are represented in our implementation as floating point numbers, each sample of a variable requires $sizeof(float)$ bytes. The tournament barrier [34] as well as the BUPC barrier require $\lceil log_2(P) \rceil$ phases to complete and hence their running time is estimated as $\lceil log_2(P) \rceil * (L + o + g)$.

Finally, in order to distinguish the LogGP parameters that refer to communication within an SMP and communication using the network connecting SMPs, in Table 3.5 we use the L, o, g, G notation for the first case, and L_{net}, o_{net}, g_{net}, G_{net} for the latter.

3.4.2 Measured vs. Predicted Results

Figure 3.10 displays the total measured time of the inter-chain algorithm as well as its different phases, measured and predicted, for network M_2 on the 16-node SMP and the 4-machine cluster. We present only the phases that consume most of the time in the running time of the algorithm; the rest of the measured time is captured in the "Other" phase shown in the graphs, whereas the predicted time does not have "Other"

Algorithm	Time
Intra-chain (SMP)	$T_{intra,smp} = \left(\frac{Sampl.\ time\ of\ 1\ iter}{P} + t_{gather\_values}\right) * total\_iter + t_{gen\_stats} * additional\_iter * vars\_per\_processor$
	$t_{gather\_values} = (P-1)*(L+o+(m-1)*G+g)+2*(\lceil log_2(P)\rceil * (L+o+g))$
	$vars\_per\_processor \approx network\_vars/P$
	$m \approx vars\_per\_processor * sizeof(float)$
Intra-chain (cluster)	$T_{intra,clu} = \left(\frac{Sampl.\ time\ of\ 1\ iter}{P} + t_{gather\_values}\right) * \left(burnin + \frac{additional\_iter}{R}\right) + t_{move\_samples} + t_{gen\_stats} * additional\_iter * \frac{vars\_per\_processor}{R}$
	$t_{gather\_values} = (P-1)*(L+o+(m-1)*G+g)+2*(\lceil log_2(P)\rceil * (L+o+g))$
	$t_{move\_samples} = (R-1) * (L_{net} + o_{net} + (n-1) * G_{net} + g_{net}) + 2 * (\lceil log_2(P*R)\rceil * (L_{net} + o_{net} + g_{net}))$
	$vars\_per\_processor \approx network\_vars/P$
	$m \approx vars\_per\_processor * sizeof(float)$
	$n \approx (vars\_per\_processor * additional\_iter * sizeof(float))/R^2$
Inter-chain (SMP)	$T_{inter,smp} = (Sampl.\ time\ of\ 1\ iter) * \left(burnin + \frac{additional\_iter}{P}\right) + t_{move\_samples} + t_{gen\_stats} * additional\_iter * vars\_per\_processor$
	$t_{move\_samples} = (P-1)*(L+o+(m-1)*G+g)+2*(\lceil log_2(P)\rceil * (L+o+g))$
	$vars\_per\_processor \approx network\_vars/P$
	$m \approx (network\_vars * additional\_iter * sizeof(float))/P^2$
Inter-chain (cluster)	$T_{inter,clu} = (Sampl.\ time\ of\ 1\ iter) * \left(burnin + \frac{additional\_iter}{P*R}\right) + t_{move\_samples} + t_{gen\_stats} * additional\_iter * vars\_per\_processor$
	$t_{move\_samples} = (P-1) * (L+o+(m-1)*G+g) + ((R-1)*P) * (L_{net} + o_{net} + (m-1) * G_{net} + g_{net}) + 2 * (\lceil log_2(P*R)\rceil * (L_{net} + o_{net} + g_{net}))$
	$vars\_per\_processor \approx network\_vars/(P*R)$
	$m \approx (network\_vars * additional\_iter * sizeof(float))/(P*R)^2$

Table 3.5. Time of the inter-chain and intra-chain algorithms on two platforms.

phase. Formulas $T_{inter,smp}$ and $T_{inter,clu}$ (Table 3.5) were used for the predicted time in graphs 3.10(b) and 3.10(d) respectively. We see that, on both platforms, the overall predicted time by our models closely match the measured time, as it is within 5% of it. From the breakup of the total time into phases, it is obvious that sampling dominates execution time on both platforms; it accounts for at least 90% of the running time. The "move samples" phase, is the phase where processors exchange samples to be sorted. For the SMP this time is negligible, whereas for the cluster it is considerably greater as the per-byte bandwidth (G in Table 3.1) is much larger on the network than within an SMP.

Figure 3.11 displays the total measured time of the intra-chain algorithm as well as its different phases, measured and predicted, for network M_2. As in Figure 3.10, we present only the phases that consume most of the running time of the algorithm. An important phase of this algorithm is the "gather values" phase, during which processors within an SMP exchange the newly-generated values of the variables they control at

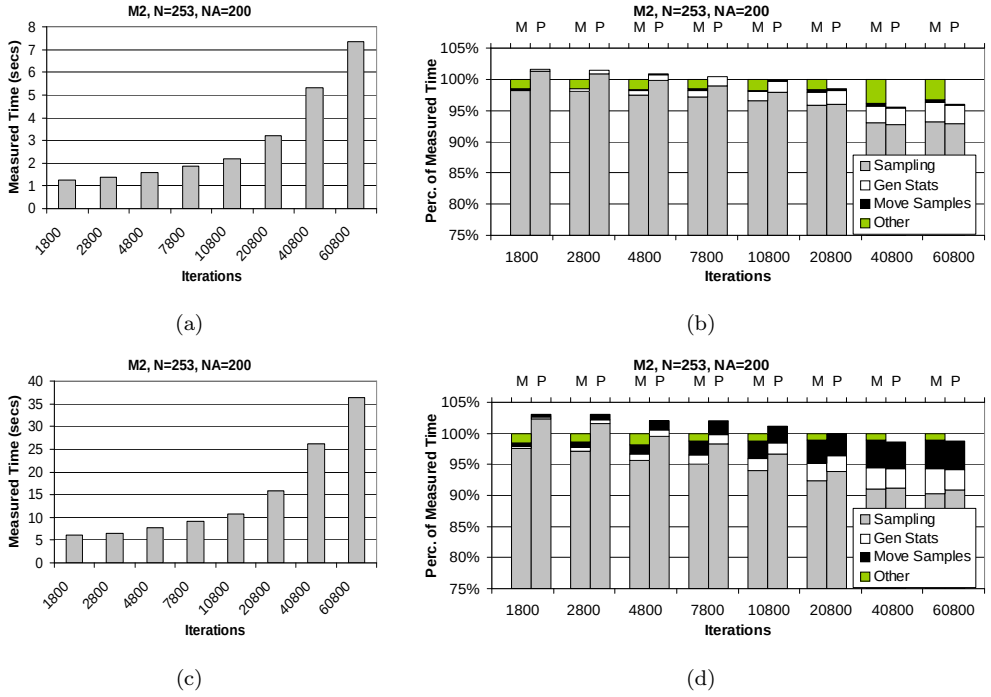


Figure 3.10. Performance of the inter-chain parallelism using 16 threads (one per processor) on a 16-node SMP (a), (b) and on a cluster of 4 quad-processor machines (c), (d). Graphs (b), (d) depict phases as a percentage of the measured time shown in (a), (c) respectively. "M" is for measured and "P" for predicted.

the end of each iteration. The actual total time needed for this phase depends on how equally-distributed the sampling load is among the processors of an SMP. If the load is not equally balanced, then some processors will have to wait for others to finish updating their variables in order to receive these new values. We see from the graphs that the time this phase takes as a percentage of the total execution time can range from 7.8% (graph 3.11(f)), to 19% (graph 3.11(b)) and up to 25% (graph 3.11(d)).

When we used formulas $T_{intra,smp}$, $T_{intra,clu}$ (Table 3.5) to predict the running time of the algorithm, we saw that the estimated time given by the models was away from the measured; the difference ranged from 7% to 25%. There could be two reasons for this: (a) load imbalance, (b) contention. Both these factors are not captured in our models. We measured the load imbalance by subtracting the minimum time spent in the "gather values" phase from the maximum time spent in the phase among all processors within an SMP. The imbalance time was added as a percentage to the predicted time in graphs 3.11(b), 3.11(d), 3.11(f). The latter graphs show that now the total predicted

time with the added imbalance is close to the actual running time, and imbalance ranges from 6.4% (graph 3.11(f)), to 10.6% (graph 3.11(b)) and up to 21.7% (graph 3.11(d)). In addition, they show that imbalance is higher on the 16-node SMP (graphs 3.11(b), 3.11(d)) than on the cluster (graph 3.11(f)). The reason is that the SMP has 16 processors whereas each SMP on the cluster has 4 and distributing the load evenly, under the restrictions explained in Section 3.2.2, can be done more easily on a 4-way machine than on a 16-way.

The effect of not dividing the load evenly among processors is more intense for the M_2 network when $N = NA$ (graph 3.11(d)). As was explained in Section 3.3.2, for this case we had to bound the b_{i0} parameters (see Figure 2.3) for all i, resulting in not good load balancing. Moreover, not having a good load balance affects the predicted sampling time, because in the $T_{intra,smp}$, $T_{intra,clu}$ formulas (Table 3.5) we divide the sampling cost of one iteration by P. In general, dividing by P underestimates the sampling cost of the intra-chain algorithm by a factor that depends on the load imbalance. For instance, when $NA < N$ in the M_2 network (graph 3.11(b), 10800 iterations), our model underestimates the sampling time by 3.1%, whereas in the $N = NA$ case (graph 3.11(d), 10800 iterations), the sampling time is underestimated by 8%.

The results for the other Bayesian networks (for both algorithms) are qualitatively similar to the ones explained above and are shown in Figures 3.12, 3.13, 3.14, 3.15.

3.5 Related Work

Several researchers have proposed parallel algorithms for Bayesian inference. Specifically, [56] describes an algorithm that assigns variables to processors which can communicate directly only with the processors for "nearby" variables, as determined by the connections present in the Bayesian network. Using such local communication, it is possible for the processors to coordinate in such a way that a number of processors can simultaneously select new values for the variables they control, while being assured that the other variables on which this selection is based are not being updated simultaneously. We believe that these kind of algorithms lack robustness, as they usually do not

map with equal efficiency onto interconnection structures different from those for which they were designed. Moreover, it requires a lot of synchronization and communication between processors. The algorithm can be used as an alternative to the intra-chain algorithm presented here, but because of its inefficiency we follow a different approach to introduce parallelism within a Markov chain.

Moreover, [39, 40] describe experimental results for a parallel version of a junction tree algorithm, implemented on a Stanford DASH multi-processor and an SGI Challenge XL. The algorithm transforms a Bayesian network into cliques and exploits parallelism across cliques (topological parallelism) and in cliques. They demonstrate speedups on random generated networks and on a medical diagnosis network. Basically, they consider parallelizing only independent operations and speedups rely on the structure of the network as well as the size of the cliques. Unlike them, we try to give good speedups without exploiting the structure of the network. Also, we apply our techniques on methods for approximate Bayesian inference (Monte Carlo simulation) whereas the algorithm in [39, 40] is for exact inference.

[57] presents a parallel algorithm for exact Bayesian inference with improved (logarithmic) worst-time complexity, when compared to other methods (e.g. [22, 39, 40]), regardless of the network topology. However, the author has no implementation results on some specific parallel architecture.

[28] describes parallel algorithms and their MPI-based implementation for Bayesian phylogenetic inference using MCMC, which are evaluated on a 32-node Beowulf cluster. In this approach, processors are arranged in a 2D grid topology so that both chain-level and subsequence-level parallelism can be used. The authors basically distribute either entire chains or parts of a chain to different processors, but they do not try to exploit intra-chain parallelism as we do. Moreover, they do not use some model of parallel computation (such as LogP/LogGP [7, 20]) to analyze the behavior of their algorithms.

Finally, [63] uses an algorithm by [58], which is a revision of [56] mentioned earlier, to map Bayesian networks onto hypercube parallel architectures. The mapping scheme maintains parent-child adjacency, is implemented and verified on a 64-node nCUBE. However, the algorithm has the same drawbacks as the one in [56].

3.6 Summary

In this chapter we describe two parallel algorithms for MCMC-based Bayesian inference in indoor positioning systems. The algorithms apply inter-chain and intra-chain parallelism on the Markov chain generation, were implemented using BUPC, and tested on three platforms: a 16-node SMP, a 4-node cluster of quad processors, and a 16 single-processor-node cluster. Our results show that the intra-chain parallelism scales well for short Markov chains achieving a speedup of 12 on the first two platforms, when the number of iterations of the MCMC method is 10,000 or less. This fact makes the algorithm particularly attractive for our positioning systems, since it was shown in the previous chapter that they do not need more than 10,000 iterations to provide good localization results. On the other hand, the inter-chain parallelism requires at least 40,000 iterations on the first two platforms and at least 60,000 iterations on the third in order to give speedups of 12 or more. Hence, the latter algorithm is suitable for applications that require long Markov chains.

We also use the LogP/LogGP model of parallel computation to analyze the behavior of these algorithms and predict their performance on different platforms. Our analysis shows that the predicted time of the model is within 5% of the measured for the inter-chain parallelism, whereas for the intra-chain it is 7%-25% less due to load imbalance that the algorithm suffers from.

As future work, we want to employ a more dynamic way to distribute the load in the intra-chain parallelism, rather than distribute it statically at compile time.

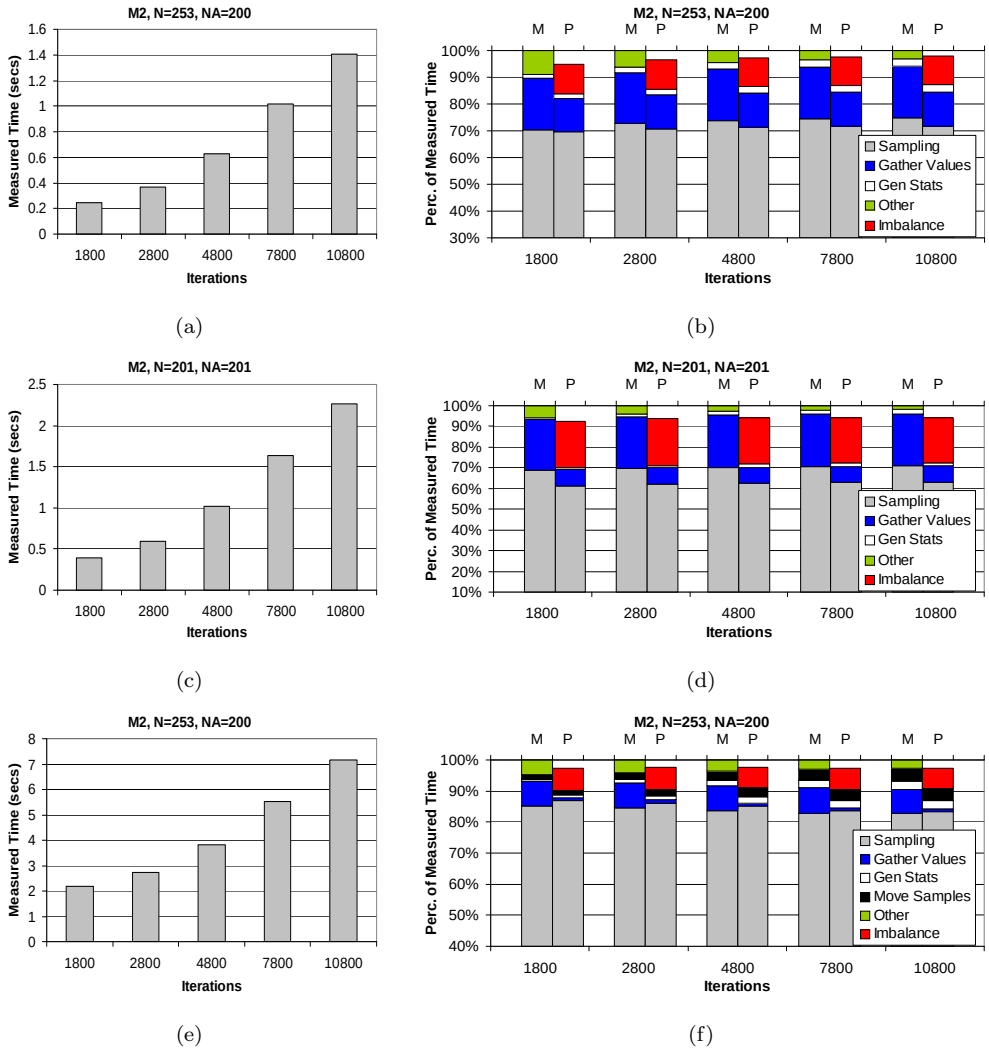


Figure 3.11. Performance of the intra-chain parallelism using 16 threads (one per processor) on a 16-node SMP (a)-(d) and on a cluster of 4 quad-processor machines (e), (f). Graphs (b), (d), (f) depict phases as a percentage of the measured time shown in graphs (a), (c), (e) respectively. "M" is for measured and "P" for predicted.

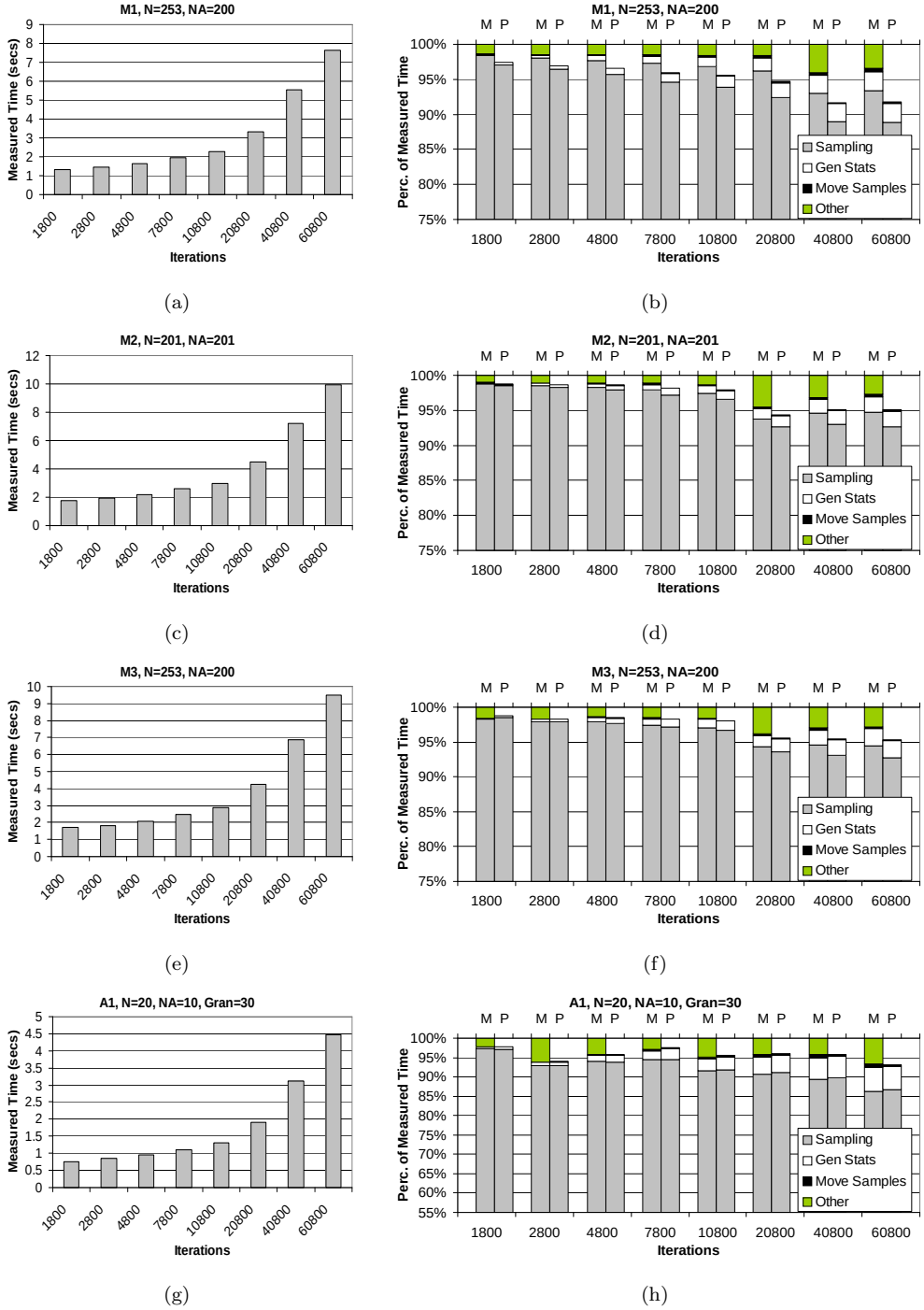


Figure 3.12. Performance of the inter-chain parallelism using 16 threads (one per processor) on a 16-node SMP. Graphs (b), (d), (f), (h) depict phases as a percentage of the measured time shown in (a), (c), (e), (g) respectively. "M" is for measured and "P" for predicted.

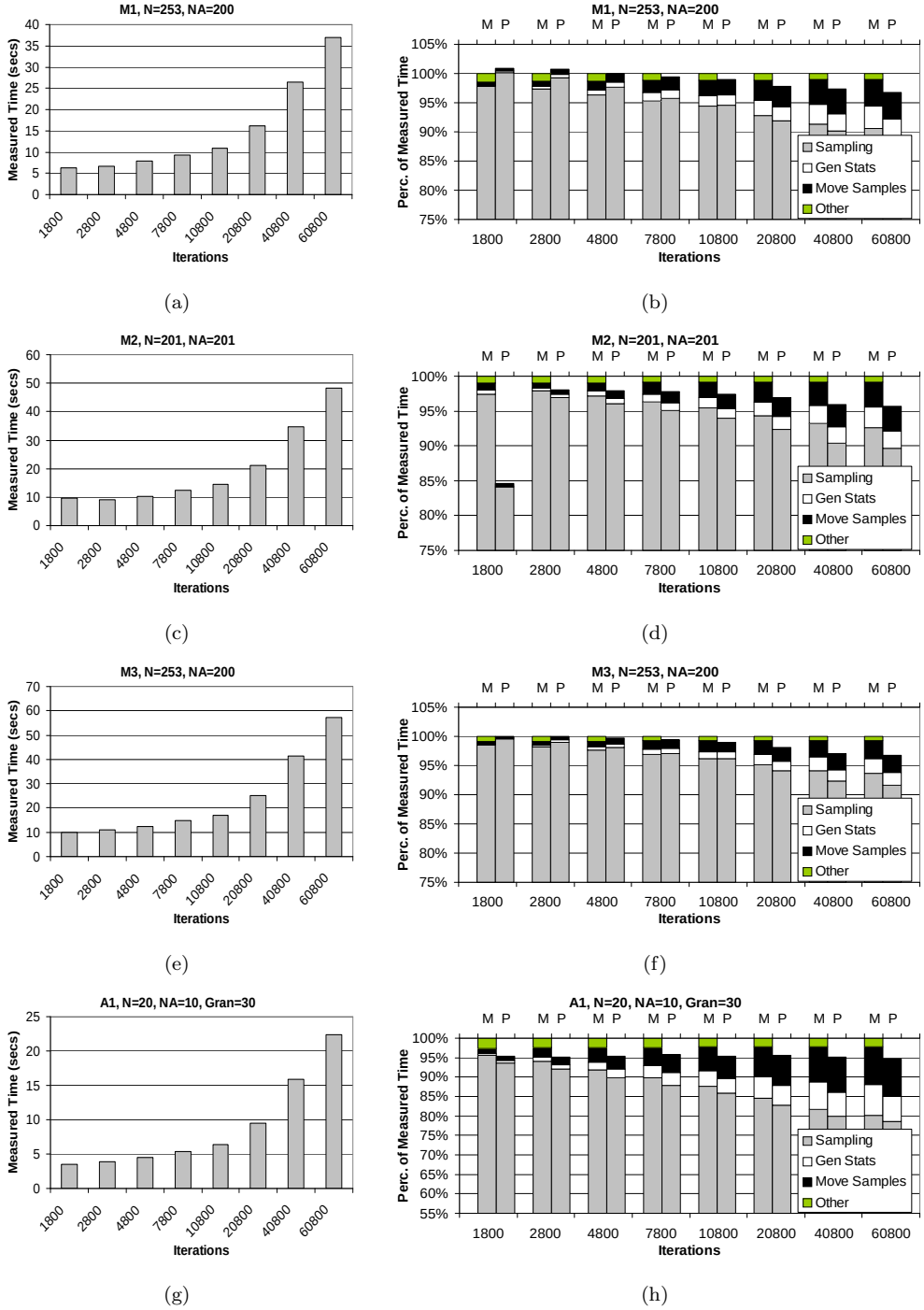


Figure 3.13. Performance of the inter-chain parallelism using 16 threads (one per processor) on a cluster of 4 quad-processor machines. Graphs (b), (d), (f), (h) depict phases as a percentage of the measured time shown in (a), (c), (e), (g) respectively. "M" is for measured and "P" for predicted.

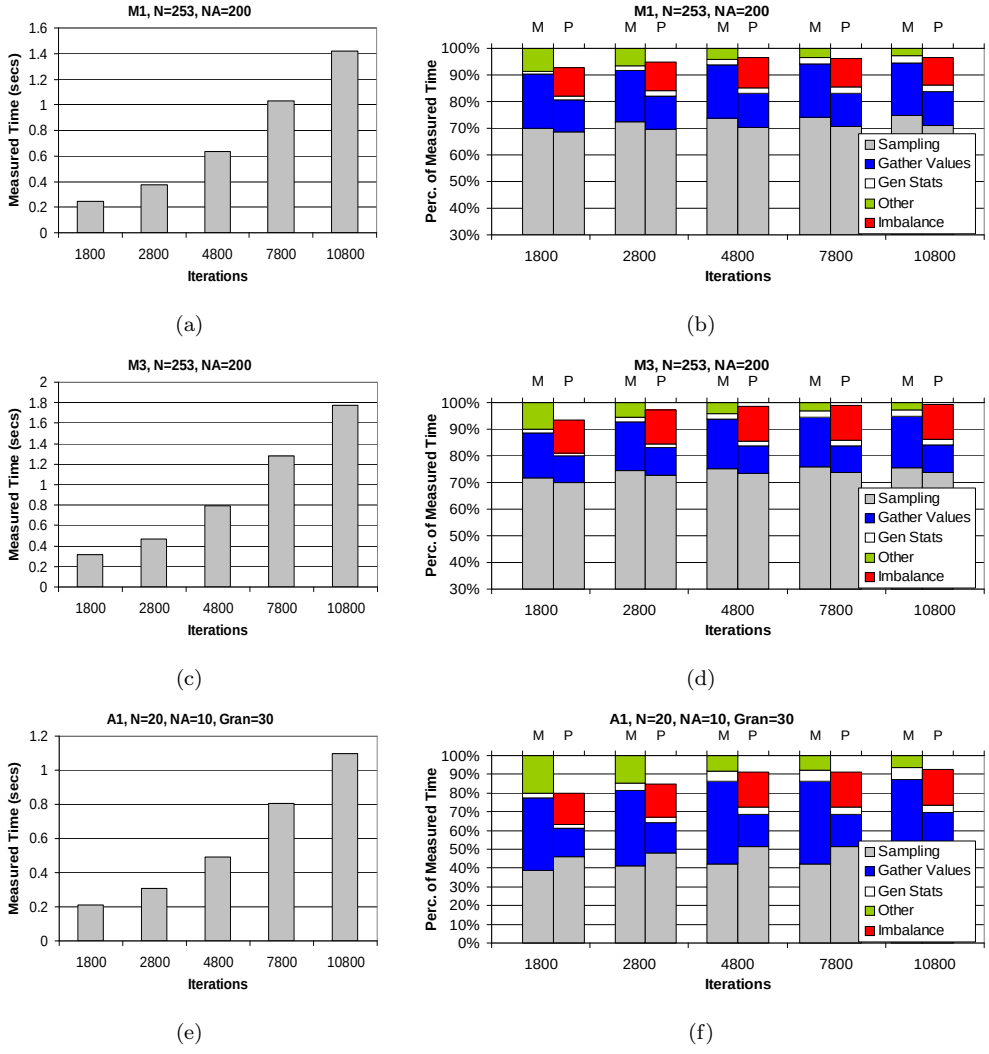


Figure 3.14. Performance of the intra-chain parallelism using 16 threads (one per processor) on a 16-node SMP. Graphs (b), (d), (f) depict phases as a percentage of the measured time shown in graphs (a), (c), (e) respectively. "M" is for measured and "P" for predicted.

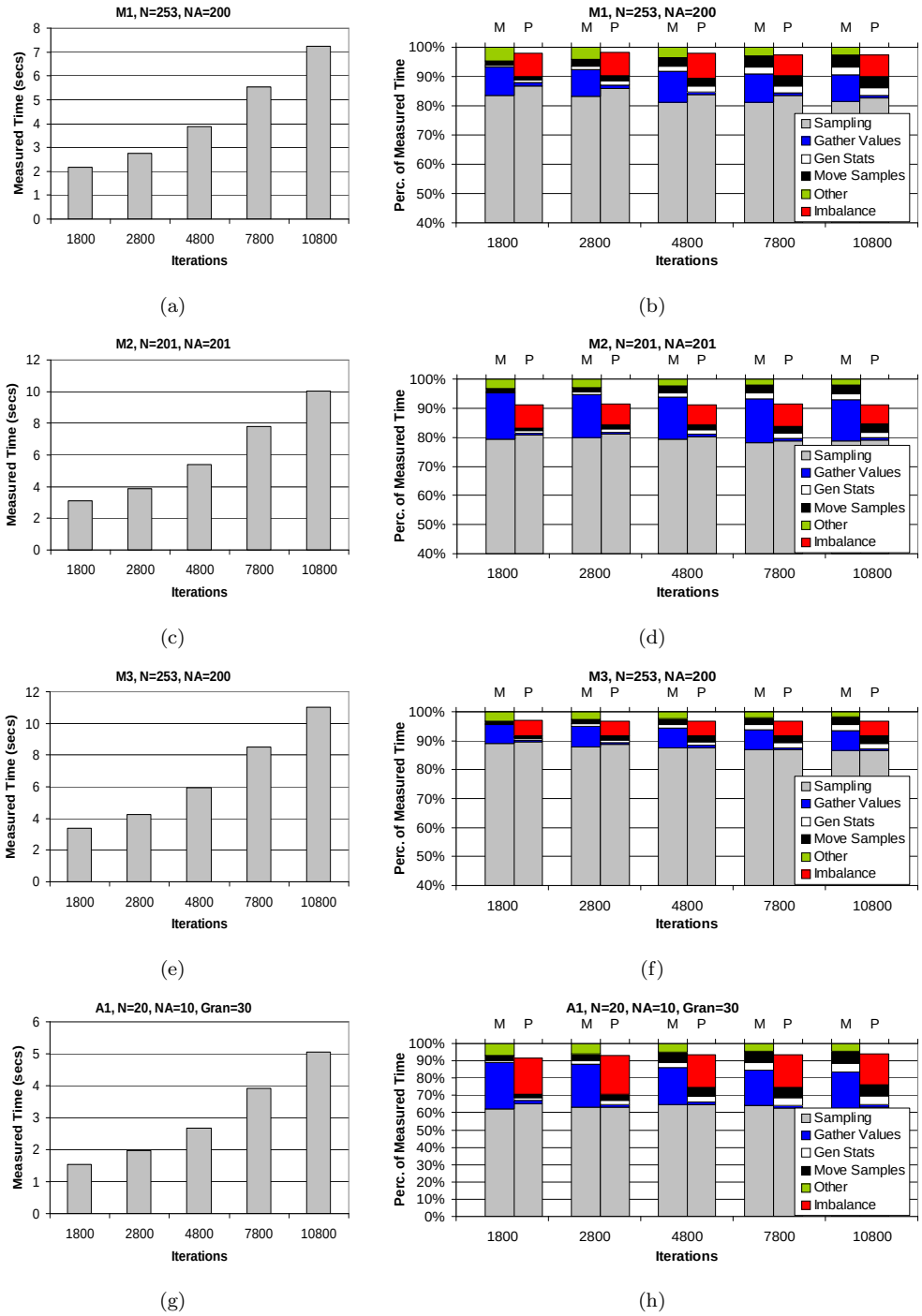


Figure 3.15. Performance of the intra-chain parallelism using 16 threads (one per processor) on a cluster of 4 quad-processor machines. Graphs (b), (d), (f), (h) depict phases as a percentage of the measured time shown in graphs (a), (c), (e), (g) respectively. "M" is for measured and "P" for predicted.

Chapter 4

The Impact of Using Multiple Antennas on Wireless Localization

4.1 Introduction

Indoor environments are particularly challenging for radio-based localization, because effects such as reflection, diffraction and scattering make signal characterization with respect to location difficult. This is one of the reasons that positioning of wireless devices indoors remains an active research area.

To date, most wireless localization systems, based on commercially available components, use received signal strength (RSS) as the base modality. However, a significant problem with RSS is that small-scale multipath fading adds high frequency components with large amplitudes to the signal at a given location. Thus, the RSS can vary by 5-10 dBm with small (a few wavelengths) changes in location. We confirm these results in this chapter. However, because the small-scale fading effects occur at the level of several wavelengths (about 12 cm at 2.4 GHz), and the granularity of the localization system is typically much larger (2-3 meters), using multiple receivers spaced on the order of a few wavelengths presents the opportunity to smooth out these effects, while maintaining the same number of landmarks used by the localization system. In particular, multiple receivers can be realized by multiplexing between multiple antennas for a given landmark.

In this chapter we investigate the impact on the localization system of using multiple receivers spaced closely together. We performed a trace-driven study on an 802.11 wireless testbed in a real office building environment. Each landmark location supported 3 antennas spaced within 1-2 feet of each other. We first investigated signal variability,

and found that using multiple antennas resulted in signal-to-distance models with better fits to a theoretical curve based on free-space models than when using a single antenna, thus confirming that additional antennas help average out small-scale environmental effects.

We then evaluated the effects of using multiple antennas on wireless localization. In order to evaluate the generality of applying multiple antennas, we evaluated the impact of multiple antennas on a diverse set of algorithms, which use an array of techniques ranging from nearest neighbor matching in signal space, represented by RADAR [9], to statistical maximum likelihood estimation, represented by the Area-Based Probability (ABP) [26], and to multilateration, represented by Bayesian Networks (BNs) [51]. We found that all algorithms under study improved their absolute position accuracy when using multiple antennas. Another key finding is that using multiple antennas significantly reduces the fraction of poor localization results across almost all of the algorithms. In one case, the median and the 90th percentile error were reduced up to 70%.

In addition to accuracy, we also investigated *stability*. We define stability as the localization system's ability to maintain a position in the face of small-scale movements of a device. For example, if a device moves 1 foot, ideally the localization system should return a result that is 1 foot away from the previous position. Instability is a common anecdotal problem with localization systems, but has not received much attention by the research community. We thus conducted a detailed evaluation of the impact of multiple antennas on the localization stability. We quantified how much the localized position of a device moves in the physical space as a function of small-scale movements of the device around its current position. Our results show that multiple antennas help improve localization stability significantly. Specifically, we can achieve up to 100% improvement in stability over the single antenna case.

A third set of experiments examined how averaging or not averaging the data from multiple antennas at a landmark position impacted the results. If averaging has no measurable impact, then a host using multiple antennas could save bandwidth and

computational resources by averaging the RSS values at a single location before localization occurred. However, we found that there is not a clear trend whether we should average or not the data from multiple antennas.

The final set of experiments explored the algorithms' sensitivity to the assumption that RSS follows a Gaussian distribution. The main reason to make such an assumption is that it makes the mathematics tractable, because the Gaussian distribution is closed under summation, i.e., the sum of two Gaussians is a Gaussian. This property also allows for averaging of multiple antenna streams to rest on a sound theoretical foundation. We generated synthetic traces that followed a Gaussian distribution using parameters from fitted measured data. Our results show that the performance behavior on real data is consistent with the localization performance under Gaussian distribution for RSS at each testing position.

The rest of this chapter is organized as follows. We present our testbed infrastructure, accuracy and stability metrics as well as our methodology for a series of investigations in Section 4.2. In Section 4.3 we present our experimental results. Specifically, we show the goodness of fit of RSS data to a theoretic model under multiple antennas, and we describe the accuracy and stability performance of localization using RADAR, ABP and BNs with real and Gaussian fingerprint sets. We provide a discussion in Section 4.4. Section 4.5 presents previous research in localization and related antenna work. Finally, we summarize in Section 4.6.

4.2 Methodology

In this section we describe our experimental methodology. We first describe the infrastructure we used, and then describe the metrics to quantify the localization accuracy and stability. We also present our methodology for a series of investigations, which include: a) the impact of small-scale movements on localization accuracy and stability, b) the impact of averaging or not RSS data on a single landmark, c) the effects of modeling RSS as a Gaussian distribution at a testing location.

Figure 4.1. WINLAB floor plan.

Location	Landmark	Antenna	x	y	z
A	1	1	136	96	6.25
	2	2	134	96	6.25
		12	135	96	6.25
B	3	3	131	43	5
	4	4	134	43	5
		14	134	43	6
C	5	5	62	48.5	7.41
	6	6	62	46.5	7.41
		16	62	47.5	7.41
D	7	7	83	1	5.83
	8	8	81	1	5.83
		18	82	1	5.83
E	9	9	151	5	5.83
	10	10	149	1	5.83
		20	148	1	5.83

Table 4.1. Coordinates x, y, z (in feet) of the 15 antennas in our testbed. Locations A, B, C, D, E are depicted as red stars in Figure 4.1.

4.2.1 Testbed Infrastructure

All data was collected using an 802.11 (Wi-Fi) network in the Wireless Network Laboratory (WINLAB) at Rutgers University. Figure 4.1 depicts the floor plan of our experimental site, where the floor size is 219ft $\times$ 169ft. All experiments were conducted in the yellow/shaded area, which is the WINLAB space. There are 10 *landmarks* (also called access points, anchors, or base-stations) deployed at five different locations with 2 landmarks per location. The locations are shown as stars in Figure 4.1 and called A, B, C, D, and E. Each landmark is a Linux machine with a 1-GHz CPU, 512 MBs of RAM and a 20-GB disk. At each location, one landmark has two Atheros miniPCI

Placement		Coordinates, Description
Floor		$(x, y, 0)$
Desk	Center	$(x, y, 3)$
	East	$(x+1, y, 3)$
	West	$(x-1, y, 3)$
	North	$(x, y+1, 3)$
	South	$(x, y-1, 3)$
	Vertical	$(x, y, 3)$, keyboard and monitor vertical to the floor with Orinoco card pointing to the ceiling
	Parallel	$(x, y, 3)$, keyboard vertical to the floor, monitor parallel to the floor
Shoulder		$(x, y, 5.16)$

Table 4.2. Placements of a mobile around a given location (x, y, z) (coordinates in feet). Each location (x, y, z) is depicted as a green dot in Figure 4.1.

802.11 wireless cards, whereas the other only one of the same type. Each card can be connected to an external 7 dBi Omni directional antenna. Thus, there can be up to 3 antennas per location or 15 antennas total. Table 4.1 presents the x, y and z coordinates of all antennas along with their numerical IDs.

The green dots in Figure 4.1 are a total of 101 *testing spots* where we collected RSS data for testing. For each testing spot (x, y, z), we collected measurements from 7 unique positions with 2 additional orientations, for a total of 9 unique *placements*. Table 4.2 summarizes all the different placements of a mobile device around a given testing spot. The mobile we consider here is a Dell laptop running Linux and equipped with an Orinoco silver card. Along the height dimension, we call the placements at 0ft, 3ft and 5.16ft as *floor, desk* and *shoulder* respectively. At the desk level, we call the small 1-foot movements around the main *center* placement the *north, south, east* and *west* placements. Finally, we call the two orientations the *vertical* and *parallel* placements.

For each placement i, we estimated an RSS vector $\overline{S_i} = (\overline{s_{i1}}, \overline{s_{i2}}, \ldots, \overline{s_{ij}}, \ldots)$. This vector is called a *fingerprint*, where $\overline{s_{ij}}$ is the average RSS corresponding to antenna j (the value of j is based on Table 4.1). Given the number of testing spots and placements around each spot, the total number of fingerprints in our experimental data set is 101 $\times$ 9 = 909. To compute a fingerprint, our laptop would transmit packets. Every landmark would forward the packets observed from all the antennas to a centralized server. The server would wait for at least 350 packets from each antenna before computing a

Combination	Description
1-antenna	Use the RSS of the landmarks with only one antenna (i.e., 1, 3, 5, 7, 9)
2-antenna-noavg	- Use the RSS of the landmarks with only one antenna - Use the RSS of the antenna with smaller ID from the landmarks with two antennas (i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
2-antenna-avg	Average the RSS of the landmarks with two antennas (i.e., avg(2, 12), avg(4, 14), avg(6, 16), avg(8, 18), avg(10, 20))
2-antenna-avg-plus-1	- Use the RSS of the landmarks with only one antenna - Average the RSS from the landmarks with two antennas (i.e., 1, avg(2, 12), 3, avg(4, 14), 5, avg(6, 16), 7, avg(8, 18), 9, avg(10, 20))
3-antenna-noavg	Use the RSS from the three antennas that exist at each landmark position (i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20)
3-antenna-avg	Average the RSS from the three antennas that exist at each landmark position (i.e., avg(1, 2, 12), avg(3, 4, 14), avg(5, 6, 16), avg(7, 8, 18), avg(9, 10, 20))

Table 4.3. Localization antenna combinations for a given landmark position.

fingerprint. We used the GRAIL infrastructure [15] to collect the packets.

We examined the effect of the orientation of multiple antennas and found it had little overall impact on the RSS from multiple antennas at one location. Specifically, we collected fingerprints with all 3 antennas at each landmark location vertical to the floor (the top of the antennas pointing to the ceiling), and another set with one of the 3 antennas being parallel to the floor. After analyzing the two data sets, we concluded that there is no significant difference on the RSS in the two data sets, and thus the results we present here are based on data collected with all landmark antennas being vertical to the floor.

4.2.2 Metrics

In this section we formalize our two metrics that apply to all localization algorithms:

Accuracy: For a given localization attempt, accuracy is the Euclidean distance between the location estimate obtained from the localization system and the actual location of the mobile device in the physical space. We refer to this distance as localization error. To capture the statistical characterization of the localization error, we study the Cumulative Distribution Function (CDF) of the localization error for all the testing placements.

Stability: Stability measures how much the location estimate moves in the physical

space in response to small-scale movements of a mobile device. We believe that stability is a desirable property in localization systems, since a position should not move too far in the physical space if there is a small-scale movement of a mobile device. For instance, when someone works at his office desk and moves his laptop 1 foot away, the localized position of the laptop should not change too much. Thus, we would like to know how stability is affected by using multiple antennas at each landmark.

We define stability by taking the Euclidean distance between the location estimate, p_1, of a mobile device at its "original" position and the localization results $p_2, p_3, \ldots, p_n$ obtained when the mobile device is moved around its original location. In essence, if p_1 and p_i ($i \neq 1$) are k feet apart, stability tells us whether the localization results of these two positions are close to the actual distance (k feet). We characterize stability by studying the CDF of the Euclidean distance between location p_1, and $p_2, p_3, \ldots, p_n$.

4.2.3 Experiments

In this section we describe three types of experiments we conducted. In all cases, our results are trace-driven. That is, we collected the fingerprints in a real environment, and then performed the localization off-line by running different localization algorithms using the collected fingerprints. When an algorithm required a training phase, e.g., training data for BNs or a signal map for RADAR, we always use the points from the center placements. We use a leave-one-out methodology for computing the accuracy and stability CDFs. That is, if applicable, we give an algorithm a training or signal map with measured or interpolated fingerprints from the center locations and give it a fingerprint from an unknown location to localize. Note that some versions of BNs do not use fingerprints with known coordinates; we describe these later in Section 4.3.4.

Horizontal and Vertical Movements Experiments. The first set of experiments we performed examined accuracy and stability as a function of small-scale movements within a given testing spot. We tested both these metrics in the horizontal plane, i.e., (x, y), using the desk-level fingerprints including the center, north, south, east, west, vertical and parallel placements. In the vertical, i.e., z plane, we used the floor, center and shoulder placements. In both cases, the center location serves as the

"original" p_1, and the other positions are the additional small-scale movements.

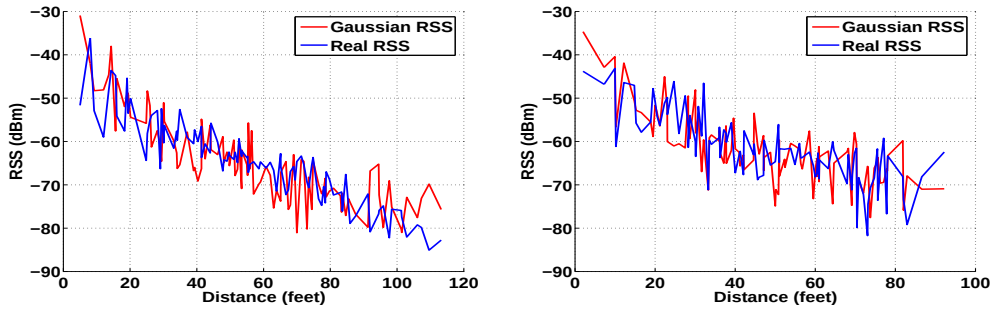
Data Averaging and Non-averaging Experiments. An important open question is if landmarks should aggregate the RSS readings from the different antennas at a given landmark location or a localization algorithm should use directly the raw RSS data. In our case, the simple aggregation scheme we examined was to perform an averaging between the antennas at a given landmark; more complex schemes are left as our future work.

We derived a systematic way to evaluate the localization performance under the cases of a single antenna, two antennas, and three antennas by either using the raw RSS readings from each individual antenna or averaging the RSS readings over two or three antennas from a landmark position. Table 4.3 summarizes various antenna combinations we consider here. In order to insure the generality of the results, we tried different combinations of 1 and 2 antennas. Each combination is given a specific name as shown in Table 4.3.

Distribution Experiments. A third class of experiments investigated the impact of assuming that RSS data follows a Gaussian distribution. Such assumptions are quite common among localization algorithms. For example, both the ABP and BNs algorithms assume the data follows a Gaussian distribution.

In order to measure the impact of this assumption, we generated a data set of fingerprints we call the *Gaussian* one. To generate this data set, we used the signal propagation constants fitted to a simple propagation model described in Section 4.3.1. The model defines the mean RSS that should be observed given the distance between a mobile and a landmark. To compute the variance, we used the variance of the fitted distribution. We then generated fingerprints using a Gaussian distribution for each one of the 101 testing spots in our testbed for all 15 antennas.

Figure 4.2 depicts RSS vs. distance graphs of real and Gaussian data for antennas 1 and 3 (see Table 4.1). As can be seen, our Gaussian data follows closely the real data, which means that our methodology of generating it is valid. Also, for all antennas in our testbed, the values of the real RSS range from -30 dBm to -90 dBm.



(a) Antenna 1

(b) Antenna 3

Figure 4.2. Gaussian and real RSS vs. distance

4.3 Results

In order to get some intuition if multiple antennas do help average out small-scale effects, we first describe a small experiment examining the goodness of fit of RSS data to a theoretic model. We then present the accuracy and stability results for RADAR, Area Based Probability (ABP), and Bayesian Networks (BNs). For each algorithm, we show overall accuracy and stability results, small-scale movements, when averaging and not averaging the antenna data, and when applying the Gaussian distribution.

4.3.1 Impact on Free Space Models

In this section we look for evidence on how multiple antennas "smooth out" the effects of small-scale variations in signal strength. An intuitive definition of "smooth out" is that the change in RSS does not vary much with a change in location. We experimented with several metrics examining rates of change in signal space vs. location when using multiple antennas. However, we did not find them grounded with sufficient theoretical foundations to use them.

Our metric is to examine how well readings from multiple antennas fit a simple propagation model. Recall that in free space, signal power decays approximately linearly with log distance. Specifically, signal strength S can be described by the following propagation model:

$$S = b_0 + b_1 \log(D) \tag{4.1}$$

Antenna Combination	Description
1-antenna-odd	RSS from antenna with odd ID (i.e., 1, 3, 5, 7, 9)
1-antenna-even-small	RSS from antenna with small, even ID (i.e., 2, 4, 6, 8, 10)
1-antenna-even-large	RSS from antenna with large, even ID (i.e., 12, 14, 16, 18, 20)
2-antenna-avg	average RSS from antennas with even IDs (i.e., avg(2, 12), avg(4, 14), avg(6, 16), avg(8, 18), avg(10, 20))
3-antenna-avg	average RSS from all three antennas (i.e., avg(1, 2, 12), avg(3, 4, 14), avg(5, 6, 16), avg(7, 8, 18), avg(9, 10, 20))

Table 4.4. Variability antenna combinations for a given landmark position.

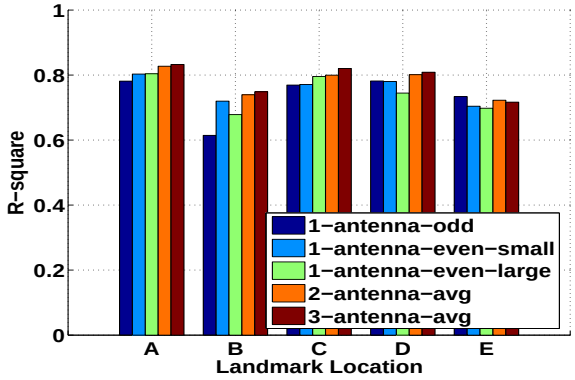
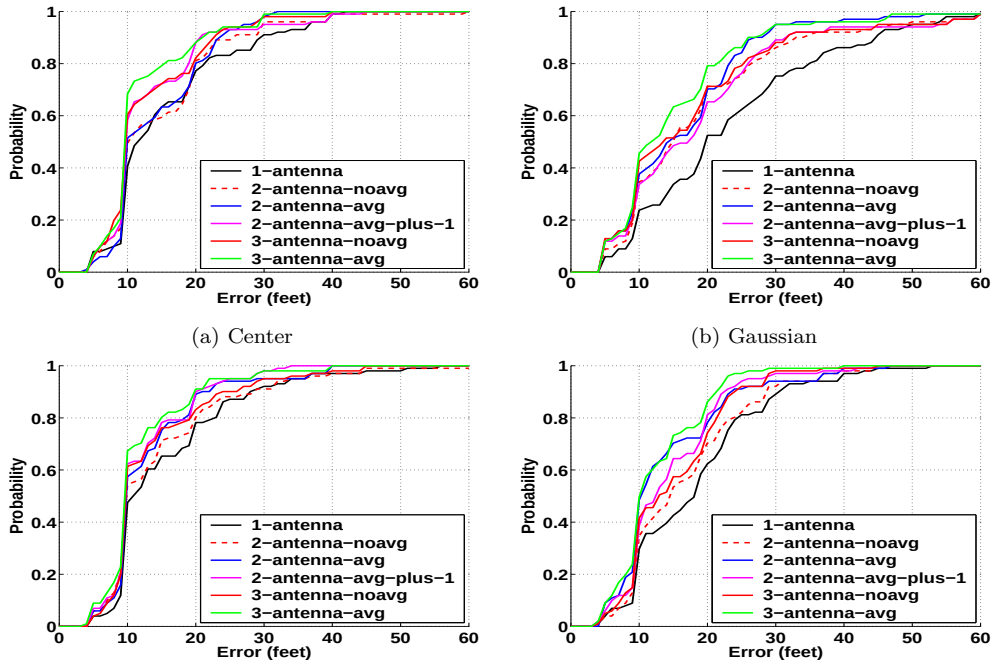


Figure 4.3. Goodness of fit of real RSS to the free space model of Equation 4.1.

where in equation (4.1), b_0, b_1 are propagation constants of the model and D is the Euclidean distance between the transmitter and the receiver. We call such a log-linear model a *free space* model.

Our approach is to add multiple antennas, and then observe the goodness of fit of the data to a best fit free-space model. The goodness of fit is observable as the coefficient of determination or R^2. Recall that R^2 can take values from 0 to 1, with a value of 1 indicating a perfect fit to the model, and a value close to 0 indicating a poor fit. Table 4.4 describes various cases of antenna combinations: a single antenna, averaging on two antennas, and averaging on three antennas.

Figure 4.3 presents the R^2 for the five landmark locations. For positions A, B, C, D by averaging the RSS of all three antennas $(3 - antenna - avg)$, the RSS data set achieves the best fit, with R^2 around 0.8. We also see that averaging $(3 - antenna -$



(a) Center (b) Gaussian

(c) Floor (d) Shoulder

Figure 4.4. Localization error CDF using RADAR

$avg, 2-antenna-avg$) gives a better fit when compared to the fit of single antennas. Thus, adding multiple antennas does improve the data fit to a simple free-space model, although the effect is not very large.

4.3.2 RADAR

The RADAR algorithm is a classic scene-matching localization algorithm [9]. RADAR requires a signal map, which is a set of fingerprints with known (x, y) locations. Given a fingerprint with an unknown location, i.e., one to localize, RADAR returns the x, y of the closest fingerprint in the signal map to the one to localize, where "closest" is defined as the Euclidean distance of the fingerprints to each other in an N-dimensional "signal space" with N landmarks [17]. That is, it views the fingerprints as points in an N-dimensional space, where each landmark forms a dimension, and returns the corresponding x, y of the closest point.

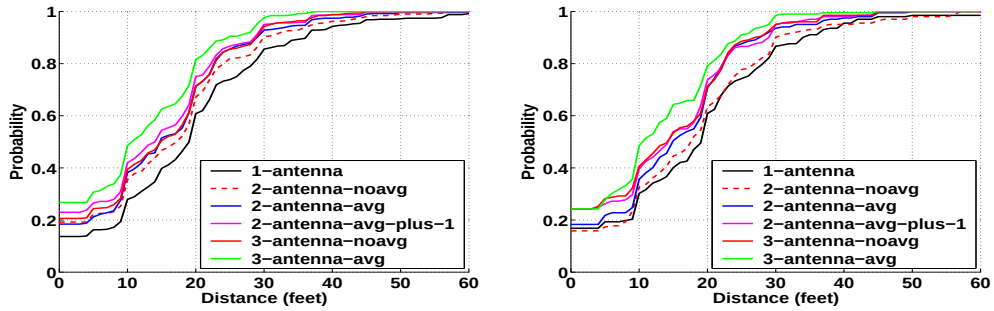
Accuracy. Figure 4.4 presents the localization error CDFs of RADAR for the antenna combinations displayed in Table 4.3. Figure 4.4(a) shows the localization error

for the center position at the desk level. We see that using 3 antennas at a landmark position results in better performance than using only 1 antenna or using 2 antennas. Specifically, at the error of 10ft, the probability increases from 42% for the $1-antenna$ case to 70% under the $3-annteana-avg$ case, and at the error of 20ft, the probability increases from 77% for the $1-antenna$ case to 90% for the $3-antenna-avg$ case. The overall improvement for the median error is 20%, moving from 12ft $(1-antenna)$ to 9.6ft $(3-antenna-avg)$, and for the 90th percentile error is 29%, moving from 30ft $(1-antenna)$ to 21.2ft $(3-antenna-avg)$.

Figures 4.4(c) and 4.4(d) are the error CDFs for the floor and shoulder placements respectively. We observe performance that is similar to that at the center placement. Specifically, at the floor level, the median error improves by 10%, moving from 10.7ft $(1-antenna)$ to 9.6ft $(3-antenna-avg)$, whereas the 90th percentile error improves by 28%, moving from 28ft $(1-antenna)$ to 20ft $(3-antenna-avg)$. At the shoulder level, the median improves by 44%, moving from 18ft $(1-antenna)$ to 10ft $(3-antenna-avg)$, whereas the 90th percentile error improves by 29%, moving from 30.6ft $(1-antenna)$ to 21.7ft $(3-antenna-avg)$. Further, the long CDF tails in Figure 4.4(c) indicate we have larger maximum localization errors at the floor level, compared to the center and shoulder placements. This is due to the fact that at the floor level the signal suffers from shadowing.

We further studied the localization performance for RADAR when modeling the RSS as a Gaussian distribution. The resulting localization errors are presented in Figure 4.4(b). By comparing 4.4(a) to 4.4(b) we see that we have higher error with Gaussian data, but the trends in the two figures are the same.

Stability. Figure 4.5 presents the localization stability for RADAR when using multiple antennas. By examining the distance CDFs at the (x, y) plane in Figure 4.5(a), we see that the total percentage of the small-scale movements being localized back to the center position increases from 13.7% for a single antenna to 26.7% when averaging the RSS from 3 antennas at one landmark position. This means we have a 100% improvement. Further, the stability at the 50th percentile moves from 19ft $(1-antenna)$ to 11ft $(3-antenna-avg)$, indicating a 42% improvement, whereas the



(a) (x, y) plane (b) z-axis

Figure 4.5. Localization stability when using RADAR

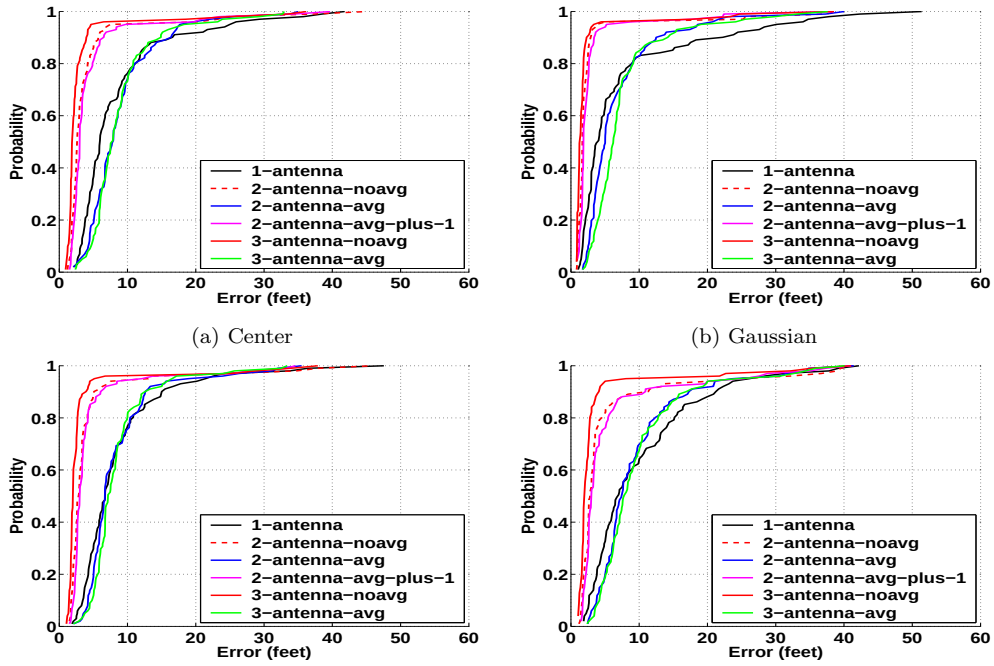
90th percentile achieves a 30% improvement by moving from 36.1ft $(1 - antenna)$ to 25.2ft $(3 - antenna - avg)$.

Moreover, as shown in Figure 4.5(b), the stability along the z-axis exhibits similar behavior. Specifically, at the 50th percentile it improves by 44%, moving from 19ft $(1 - antenna)$ to 10.5ft $(3 - antenna - avg)$, and at the 90th percentile it improves by 30%, moving from 35.4ft $(1 - antenna)$ to 24.7ft $(3 - antenna - avg)$. This is very encouraging as better localization stability strongly indicates that using multiple low-cost antennas for improving localization performance is effective.

4.3.3 Area Based Probability

Area Based Probability (ABP) utilizes an Interpolated Map Grid (IMG) to interpolate the signal map and cover the entire experimental floor. Specifically, the floor is divided into a regular grid of equal-sized tiles. Since direct measurement of the fingerprint for each tile is expensive and prohibitive for fine-grained tiles, it follows an interpolation approach. The goal of using an IMG fitting is to derive an expected RSS fingerprint for each tile from the data set that would be similar to an observed one.

ABP returns a set of tiles bounded by a probability that a mobile device is within the returned tile set. The probability is called the confidence α and it is adjustable by the user. We used a tile size of 10in $\times$ 5in, which is comparable to the distance between antennas at a landmark location (1 or 2 feet). ABP assumes the distribution of RSS for each landmark follows a Gaussian distribution with mean as the expected



(a) Center (b) Gaussian

(c) Floor (d) Shoulder

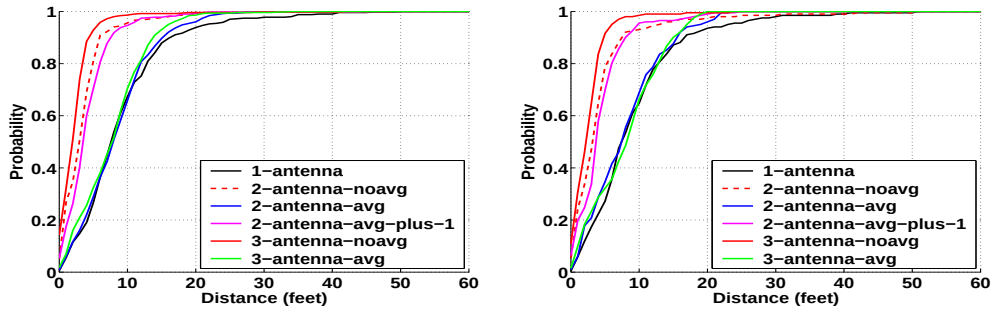
Figure 4.6. Localization error CDF using ABP

value of RSS reading vector $\mathbf{s}$. The Gaussian random variable from each landmark is independent. ABP then computes the probability of the mobile device being at each tile L_i, with $i = 1...L$, on the floor using Bayes' rule:

$$P(L_i|\mathbf{s}) = \frac{P(\mathbf{s}|L_i) \times P(L_i)}{P(\mathbf{s})} \tag{4.2}$$

Given that a mobile device must be at exactly one tile satisfying $\sum_{i=1}^{L} P(L_i|\mathbf{s}) = 1$, ABP normalizes the probability and returns the most likely tiles/grids up to its confidence α [26]. In order to normalize for accuracy and stability results, we select the tile with the median localization error from the tile set. In all results we show next, the value of the confidence level is $\alpha = 0.75$.

Accuracy. Figure 4.6(a) shows the localization error CDFs of ABP at the center placement when using multiple antennas. The $3 - antenna - noavg$ case has the best performance. Comparing $3 - antenna - noavg$ to $1 - antenna$, we observe that the median error moves from 7ft to 2ft and the 90th percentile error moves from 16ft to



(a) (x, y) plane (b) z-axis

Figure 4.7. Localization stability when using ABP

4ft. Thus, the location accuracy has an improvement over 70% for both the median as well as the 90th percentile error when using 3 antennas at a given location.

The error CDFs for the floor and shoulder level in Figures 4.6(c) and 4.6(d) have qualitatively similar performance to the center position. Specifically, by using the RSS readings from each of the 3 antennas at a given landmark location we get the best performance. Moreover, we notice that the CDFs at the floor level only have slightly longer tails than those at the center and shoulder placement. This indicates that interpolating the signal map across the experimental floor helps smooth out the signal variability and thus reduces the maximum localization errors.

Finally, Figure 4.6(b) presents the localization errors using the Gaussian data set. As with real data, we observe better performance under the cases of multiple antennas. Moreover, the performance of the $3-antenna-noavg$ case with the Gaussian simulated data is even better than with the real experimental data.

The results in location accuracy show that when using the approach of an interpolated signal map with grid size smaller than the distance between 2 adjacent antennas at a landmark location, by using multiple antennas we can achieve better location accuracy improvement than using solely the raw fingerprints in the signal map (as RADAR does).

Stability. Figure 4.7 shows that using multiple antennas at a given location helps improve localization stability, with the $3-antenna-noavg$ case providing the highest

stability improvement. Notably, the total percentage of testing points, under small-scale movements, with stability distance zero increases from less than 5% for a single antenna to over 14% for the case of $3 - antenna - noavg$. Similar to RADAR, this is an over 100% stability improvement. Further, examining Figure 4.7(a) the stability distance at the 50th percentile moves from 8ft for the case of $1 - antenna$ to 2ft for the $3 - antenna - noavg$ case, resulting in a stability improvement of 75%. We also observe over 73% improvement for the 90th percentile. In the z axis as shown in Figure 4.7(b), employing multiple antennas at a given location again provides similar improvement in localization stability.

One effect we observe is that when using signal map interpolation, the cases of averaging the RSS readings from multiple antennas at a given location such as $2 - antenna - avg$ and $3 - antenna - avg$ have the same localization performance as the single antenna case. We believe that this is because for tiny grids (10in $\times$ 5in), averaging RSS at a given location is just like placing a single landmark at a location, which is the same as having a single antenna in a landmark.

4.3.4 Bayesian Networks

The Bayesian Networks we consider here are M_1, M_2, M_3, were presented in Section 2.4, and depicted in Figure 2.3. They encode the relationship between the RSS and a location based on the signal-versus-distance propagation model shown in Equation 4.1. In addition, they capture noise and outliers by modeling the signal strength as a Gaussian distribution given by expression 2.6.

Network M_1 is the simplest amongst the three, and requires a training set in order to give good localization results. Network M_2, as was shown in [51], can localize with no training fingerprints, leading to a zero-profiling technique for location estimation. The impact of multiple antennas on zero-profiling is a key effect we tested for, since this approach has the benefit of not having to collect fingerprints at known locations. Finally, network M_3 extends M_2 by incorporating the corridor effect. That is, when a location (X, Y) shares a corridor with a landmark, then the signal strength tends to be stronger along the entire corridor. We define "sharing a corridor" as having an

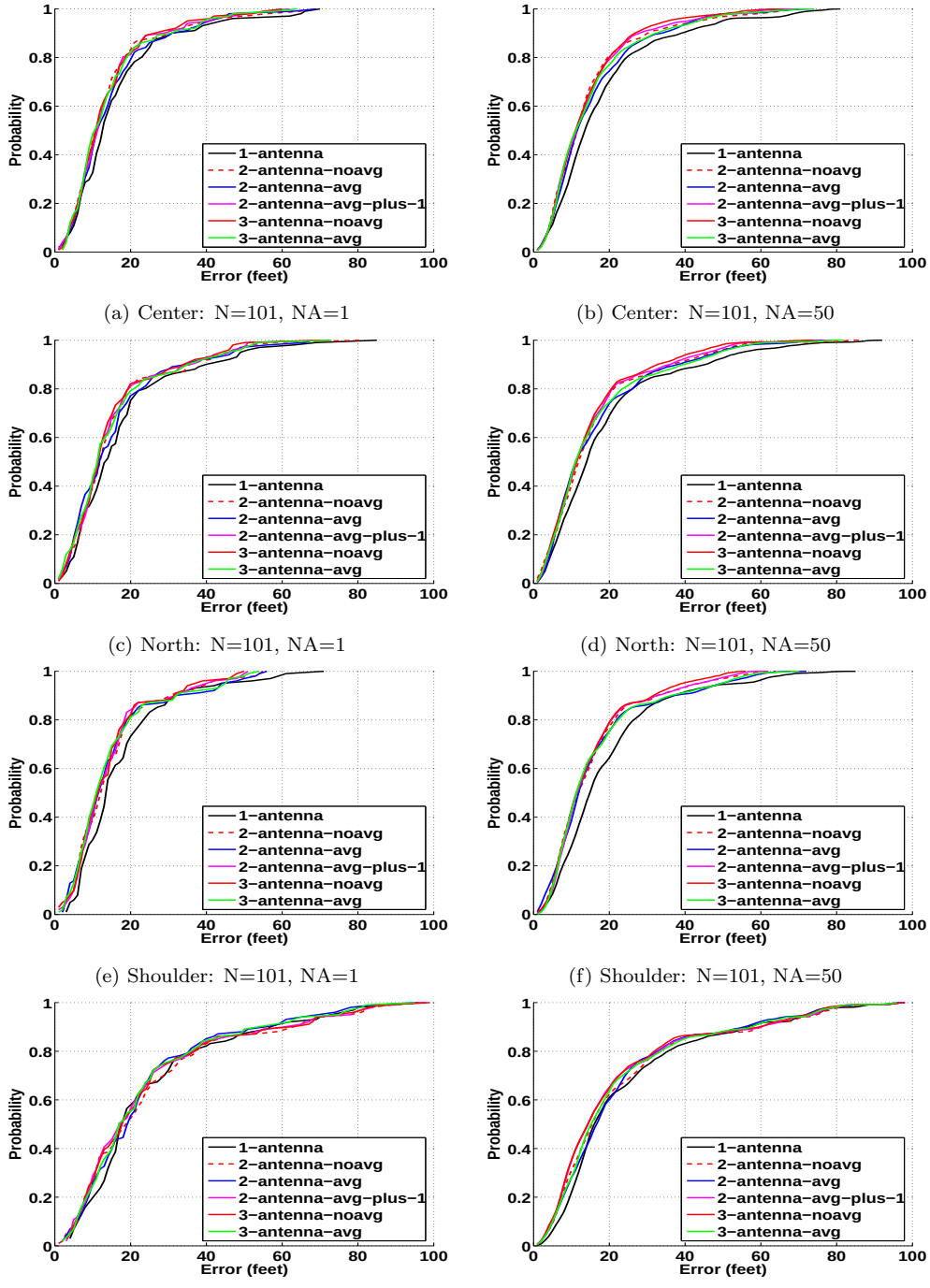
X- or Y-coordinate within three feet of the corresponding landmark coordinate. In all graphs we present in this section, N denotes the size of the training set, out of which we localize NA devices.

Accuracy. Figure 4.8 presents the localization error CDFs of M_2 under multiple antennas, when using a training set. Specifically, Figures 4.8(a), 4.8(c), 4.8(e), 4.8(g) present the error CDFs, when localizing one device (NA=1) at the center, north, shoulder, floor placements respectively, whereas Figures 4.8(b), 4.8(d), 4.8(f), 4.8(h) the error CDFs for the same placements when localizing 50 devices (NA=50).

When localizing one device at the center placement, we see that all curves have similar performance, although using only one antenna, $1-antenna$ case, has slightly worse performance than the other cases. The curves for the north, shoulder, floor placements exhibit similar trend, although at the floor level, we notice higher errors, probably due to shadowing that the signal strength suffers from at this placement.

As the number of devices for localization increases from 1 to 50, we notice that the error CDFs corresponding to multiple antennas are more clearly separated from the single antenna case. The improvement is primarily on the 90th percentile. In particular, when comparing $1-antenna$ to $3-antenna-noavg$ at the center placement, the median error from 14ft ($1-antenna$) reduces to 11ft ($3-antenna-noavg$), and the 90th percentile error reduces from 39ft ($1-antenna$) to 27ft. The trends of the curves at the other placements are similar, although at the floor level we notice higher errors as was the case when localizing one device.

Figure 4.9 presents the error CDFs of M_2 when localizing 51 devices simultaneously with no training set at the center, north, shoulder, floor placements. Unlike when using training set to localize, we see that now case $3-antenna-noavg$ has the best performance. The accuracy improvement on the median error between $1-antenna$ and $3-antenna-noavg$ is 40% (from 22ft to 13ft) at the center placement, 36% (from 22ft to 14ft) at the north placement, 46% (from 26ft to 14ft) at the shoulder placement, and 33% (from 18ft to 12ft) at the floor placement. The improvement on the 90th percentile for the same pairs of antenna combinations is from 54ft to 28ft (center), from 55ft to 33ft (north), from 54ft to 31ft (shoulder), and from 52ft to 34ft (floor). Conclusively,



(a) Center: N=101, NA=1

(b) Center: N=101, NA=50

(c) North: N=101, NA=1

(d) North: N=101, NA=50

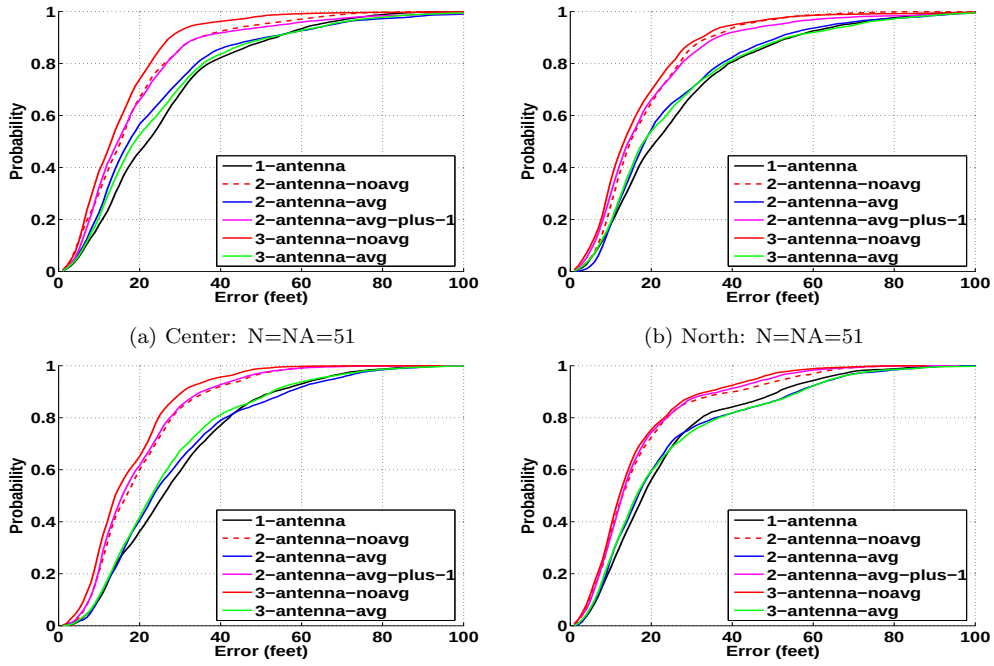
(e) Shoulder: N=101, NA=1

(f) Shoulder: N=101, NA=50

(g) Floor: N=101, NA=1

(h) Floor: N=101, NA=50

Figure 4.8. Localization error CDFs using Bayesian network M_2.



(a) Center: N=NA=51

(b) North: N=NA=51

(c) Shoulder: N=NA=51

(d) Floor: N=NA=51

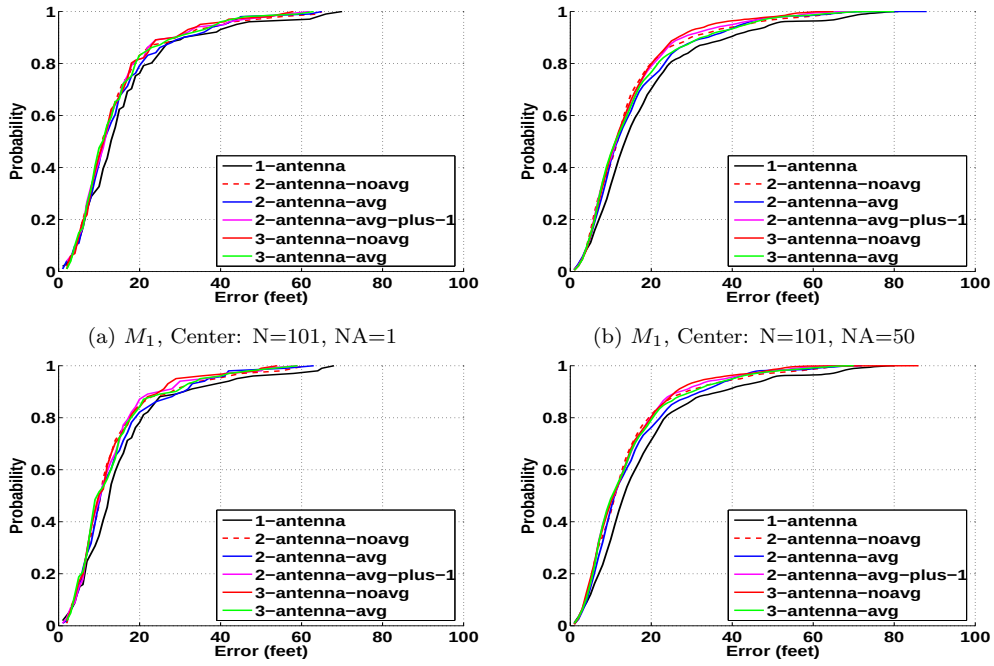
Figure 4.9. Localization error CDFs using Bayesian network M_2 with no training fingerprints.

our empirical results suggest that more antennas per landmark location primarily help improve the localization accuracy of Bayesian networks when there is no training set.

Another important observation is that 3 antennas per landmark location, case $3 - antenna - noavg$, makes the localization error when locating multiple mobile devices comparable to that when locating a single device. Specifically, from Figures 4.8(a), 4.8(b), 4.9(a), which present the error CDFs when localizing 1, 50, and 51 devices respectively, we see that the median error for $3 - antenna - noavg$ is 11ft, 11ft and 13ft, whereas the 90th percentile error is 24ft, 27ft and 28ft. Thus, both the median and the 90th percentile errors are of the same magnitude.

Figure 4.10 depicts the error CDFs at the center placement when localizing 1 and 50 devices for the other two BNs, namely M_1, M_3. The graphs show that these networks perform similarly to M_2 and this applies to the other placements too.

Figure 4.11 presents the localization accuracy of BNs M_1, M_2, M_3 when using the Gaussian simulated data set. We observe that in all figures the different antenna combinations can be placed into three groups based on their performance: (a) $3 -$



(a) M_1, Center: N=101, NA=1

(b) M_1, Center: N=101, NA=50

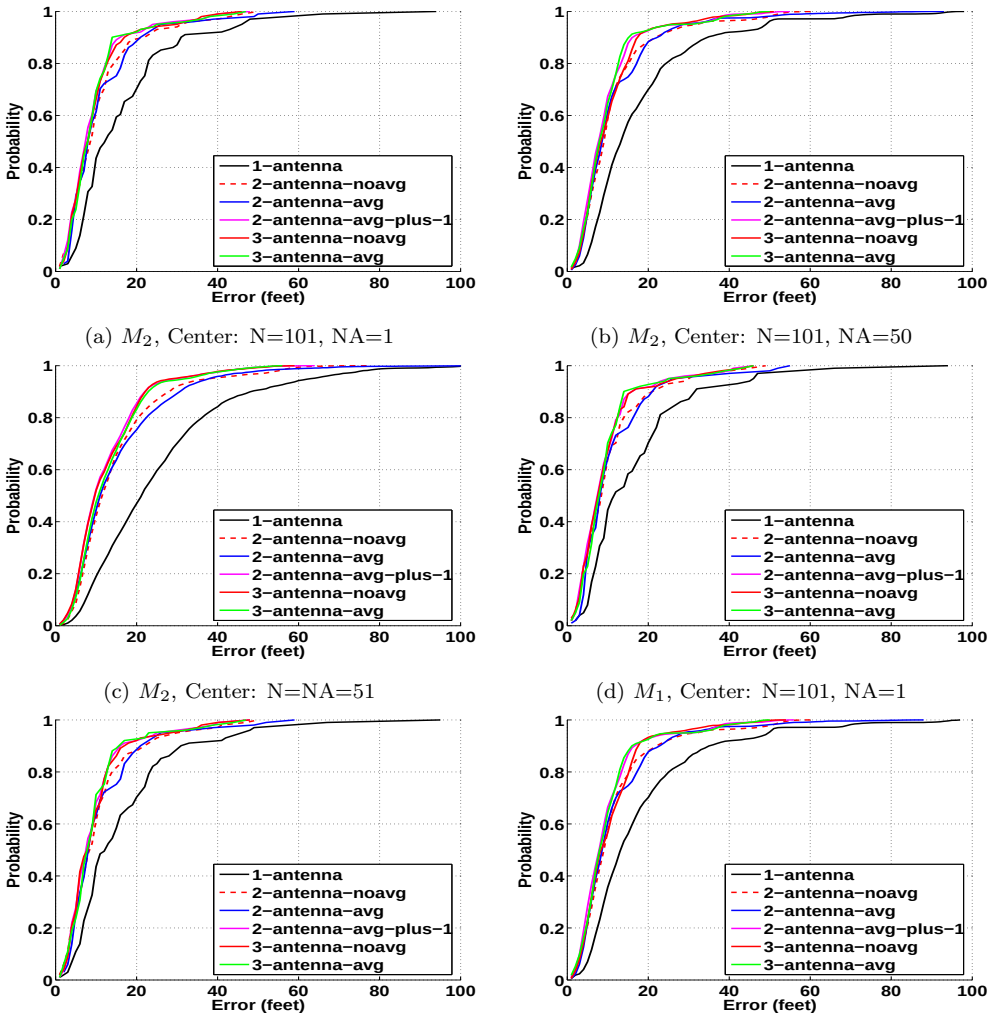
(c) M_3, Center: N=101, NA=1

(d) M_3, Center: N=101, NA=50

Figure 4.10. Localization error CDFs using Bayesian networks M_1, M_3.

$antenna-noavg$, $3-antenna-avg$, $2-antenna-avg-plus-1$, (b) $2-antenna-noavg$, $2-antenn-avg$, and (c) $1-antenna$. The first group has the best performance, whereas the last the worst. Intuitively, this kind of grouping should be expected, since BNs assume that the RSS follows a Gaussian distribution, and thus the averaged RSS of antennas that belong to the same landmark position is also Gaussian [14]. Furthermore, the RSS measured at each antenna is close to the RSS of the other antennas at a given landmark location (since the landmark antennas are close to each other), and thus the averaged RSS should be close to the RSS of each antenna. Therefore, unlike with real experimental data, BNs perform similarly either we average or not the RSS of the multiple antennas at a given landmark location.

Stability. Figure 4.12 presents localization stability CDFs for Bayesian network M_2, on the (x, y) plane (desk placements) and the z-axis. Figures 4.12(a), 4.12(c) show that when localizing 1 or 50 devices on the (x, y) plane, the more antennas per landmark location, the better the stability. When locating 1 device, stability improves by 36% at the 50th percentile, moving from 11ft ($1-antenna$) to 7ft ($3-antenna-noavg$), and by



(a) M_2, Center: N=101, NA=1

(b) M_2, Center: N=101, NA=50

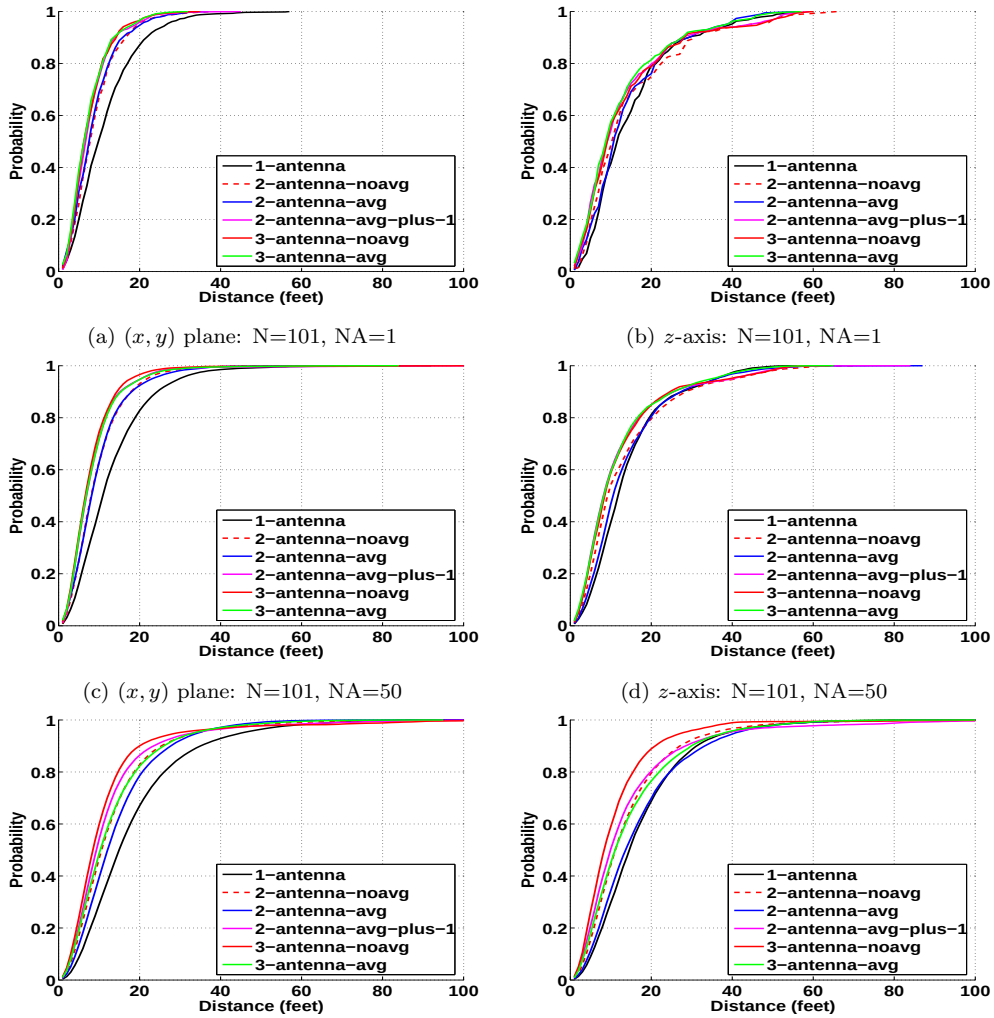
(c) M_2, Center: N=NA=51

(d) M_1, Center: N=101, NA=1

(e) M_3, Center: N=101, NA=1

(f) M_3, Center: N=101, NA=50

Figure 4.11. Gaussian approach: localization error CDFs using Bayesian networks M_1, M_2, M_3.



(a) (x, y) plane: N=101, NA=1

(b) z-axis: N=101, NA=1

(c) (x, y) plane: N=101, NA=50

(d) z-axis: N=101, NA=50

(e) (x, y) plane: N=NA=51

(f) z-axis: N=NA=51

Figure 4.12. Localization stability of Bayesian network M_2.

33% at the 90th percentile, moving from 21ft ($1-antenna$) to 14ft ($3-antenna-noavg$). When locating 50 devices, stability improves by 41% at the 50th percentile, moving from 12ft ($1-antenna$) to 7ft ($3-antenna-noavg$), and 42% at the 90th percentile, moving from 26ft ($1-antenna$) to 15ft ($3-antenna-noavg$). However, the difference between the cases of $3-antenna-noavg$ and $3-antenna-avg$ is negligible. In general, we noticed that when our networks use training data, averaging or not the RSS from multiple antennas at a given location does not make any difference on the localization stability on the (x, y) plane, regardless of the number of multiple devices we localize simultaneously. Along the z-axis, Figures 4.12(b), 4.12(d) show that there

is not large improvement by using multiple antennas.

Figures 4.12(e) and 4.12(f) show that when no training data is used, case $3-antenna-noavg$ has the best performance. Moreover, the more antennas, the better the performance, but $3-antenna-noavg$ gives better stability than $3-antenna-avg$. On the (x, y) plane, the median of stability improves by 43%, from 16ft $(1-antenna)$ to 9ft $(3-antenna-noavg)$, and the 90th percentile by 44%, from 36ft $(1-antenna)$ to 20ft $(3-antenna-noavg)$. Along the z-axis, there is an improvement of 40% at the median, moving from 15ft $(1-antenna)$ to 9ft $(3-antenna-noavg)$, and 34% at the 90th percentile, moving from 32ft $(1-antenna)$ to 21ft $(3-antenna-noavg)$.

Figure 4.13 presents stability CDFs for networks M_1, M_3 when locating 1 and 50 devices. The graphs show that the stability achieved by using multiple antennas in these two networks is similar to the one of network M_2 when using training data (Figures 4.12(a), 4.12(b), 4.12(c), 4.12(d)). Overall, we conclude that multiple antennas can help BNs improve their localization stability.

4.4 Discussion

We believe that the distance between antennas at a landmark location and the distance between points where training/testing fingerprints are collected can have a significant impact on the results when averaging or not the RSS of multiple antennas. Figure 4.4 shows that the location accuracy of RADAR in the $3-antenna-avg$ case is slightly better than in the $3-antenna-noavg$ case. We suspect this is because in our study the distance between two antennas is small, only 1 to 2 feet away from each other, whereas the testing points are about 5-10 feet away from each other, which is a magnitude of 5 times larger. The fine-grained RSS differences between the 3 antennas will not affect the coarse-grained fingerprint matching. Therefore, we believe the 3 separate antennas at one landmark location will not be treated as 3 separate landmarks by the algorithm when performing fingerprint matching in the signal space. On the other hand, averaging the RSS from 3 antennas reduces the RSS variability and thus $3-antenna-avg$ provides the best performance for RADAR.

ABP uses an interpolated signal map and a tile size of 10in $\times$ 5in, which is comparable to the distance between two antennas at a landmark location. As a result we believe each antenna will be treated as a separate landmark. We found that using the RSS from each individual antenna ($3 - antenna - noavg$), achieves the best improvement in both location accuracy and stability as shown in Figures 4.6, 4.7.

A full characterization of the effect of distances between multiple antennas and distances to collect fingerprints is left as future work.

4.5 Related Work

There have been active research efforts in positioning wireless devices indoors. Among these, improving localization accuracy is the main focus, and range from algorithm development, to landmark placement, and to increasing the landmark density. Various localization schemes [9, 26, 48, 51, 59] utilizing different physical modalities, such as RSS and Time-Different-Of-Arrival (TDOA), and different mapping functions, such as fingerprint matching and statistical approaches, have been developed to more accurately position mobile devices. [16] investigates the impact of landmark placement on localization performance and proposes an optimal landmark deployment approach to improve the performance without increasing the number of landmarks (about 1 landmark per 4000 square feet). On the other hand, [46] shows that by using the truncated singular value decomposition technique and increasing the landmark density (about 1 landmark per 1000 square feet) a better localization accuracy can be achieved. Similarly, [8] shows that the greater the number of landmarks, the more accurate the location estimate is. Also, [8, 71] demonstrate that by reducing the grid spacing, localization results improve, but at the same time the computational cost (or delay) of the required position increases. In this chapter, we take a different approach by exploring the impact on the localization system when using multiple antennas at a given location.

Moreover, a lot of work has focused on landmark selection, since subsets of available landmarks may report correlated readings, leading to needless redundancy and possible biased estimates. The most commonly used selection methodology is to choose a subset

of landmarks with the highest observation RSS, as the strongest landmarks provide the highest probability of coverage over time [74]. However, it is also known [38] that the variance of measurements from a landmark increases with its mean power at a given location. In cases where the measured RSS from a landmark exhibits a high degree of variance, the RSS values of the training fingerprints may be very different than the online measurement, degrading the accuracy of location estimation [37]. Recently, [18] proposed a strategy where the landmarks that best discriminate the training fingerprint points are the ones selected for positioning. In that work, landmark selection is carried out offline, whereas in [42] the selection is performed during the online operation of the system to introduce resiliency to loss of landmarks. Unlike all this work, we focus on how to reduce the environmental effects on RSS and RSS-based localization algorithms.

Work that is closely related to ours is [32, 44, 50, 61]. [44] shows that by making RSS measurements over many frequency channels, it is possible to isolate frequency specific fades. A simple averaging operation on the bands, can reduce the multipath effect on the RSS measurements leaving a much flatter fading response. [50] presents a detailed characterization of signal strength behavior in an 802.15.4 network environment with monopole antennas. Their findings demonstrate that the relative antenna orientation between receiver-transmitter pairs is a major factor in signal strength variability, even in the absence of multipath effects. Further, [32] reviews the principles of radio propagation in indoor environments and also explores relevant concepts such as spatial and temporal variations of the channel, large scale path losses and mean excess delay. Theoretical distributions of the sequences of arrival times, amplitudes and phases are presented. Moreover, [61] provides a survey of various propagation models for both indoor and outdoor environments. Our work is different in that in addition to a signal variability study, we investigate the impact of using multiple antennas on wireless localization including accuracy and stability.

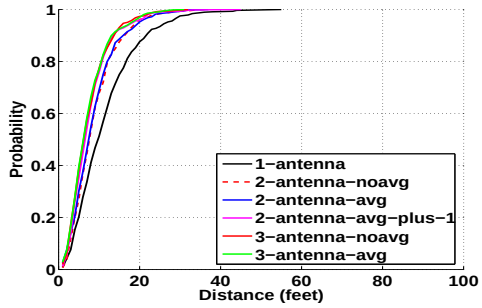
Finally, there has been a wide range of research covering development of antennas suitable for mobile communications systems, and many experimental results have been

reported to show the system requirements and feasibility. An application of phased-array and adaptive antennas has been suggested in recent years for mobile communications to overcome the problems of single-antenna systems [10, 11, 19, 30, 31, 52, 68, 72]. Specifically, these two types of antennas have been shown to help improve a mobile system's performance in several ways, such as by increasing channel capacity and spectrum efficiency, extending range coverage, reducing co-channel interference and multipath fading.

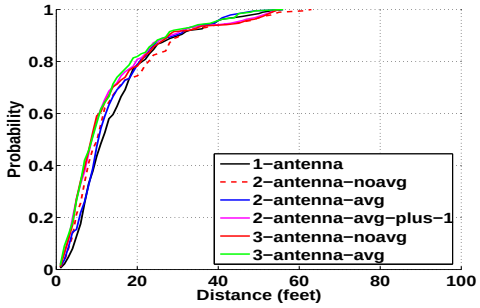
4.6 Summary

By employing multiple antennas spaced closely at a given location, we investigated the impact on wireless localization. We performed a trace-driven study on an 802.11 testbed in a real office environment. First, through a signal variability study, we found that adding additional antennas helps average out small-scale environmental effects.

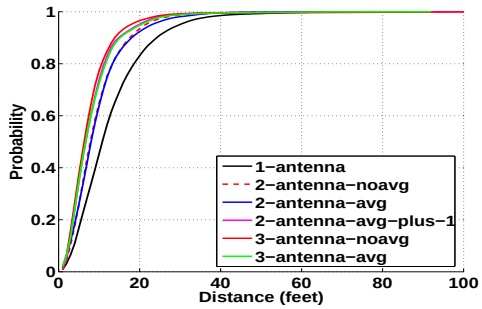
We then studied the performance of a representative set of localization algorithms, in accuracy and stability, when using multiple antennas. We found that all algorithms under study improved their absolute accuracy by either averaging or not the RSS from multiple antennas at a given location. Specifically, both the median error as well as the 90th percentile error can be reduced up to 70%. Our investigation of the localization stability when there are small-scale movements of a mobile device shows that multiple antennas help improve stability significantly; up to 100% improvement over the single antenna case. In summary, we found that adding multiple antennas gives performance benefits for localization that are worth the low cost of the antennas.



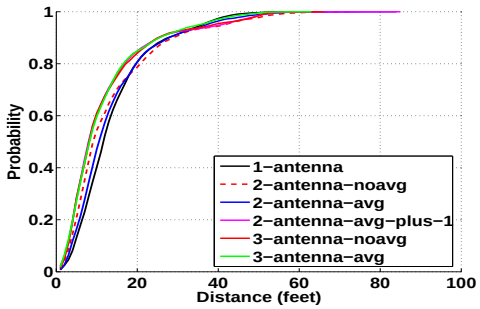
(a) M_1, (x, y) plane: N=101, NA=1



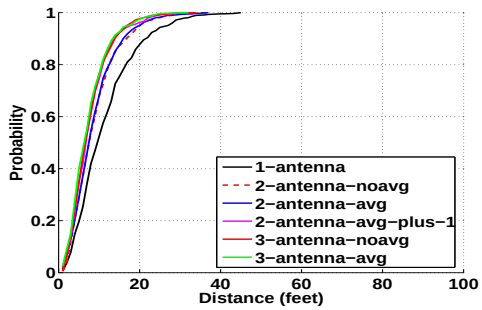
(b) M_1, z-axis: N=101, NA=1



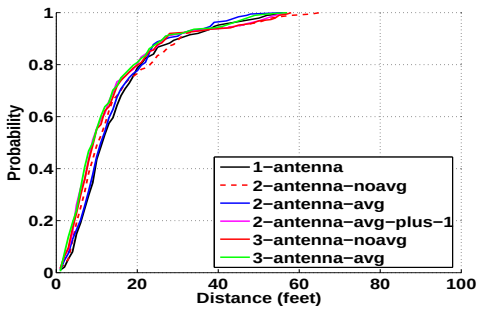
(c) M_1, (x, y) plane: N=101, NA=50



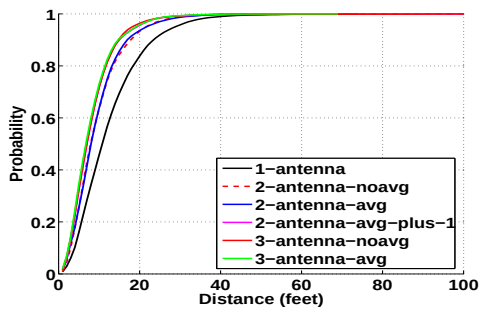
(d) M_1, z-axis: N=101, NA=50



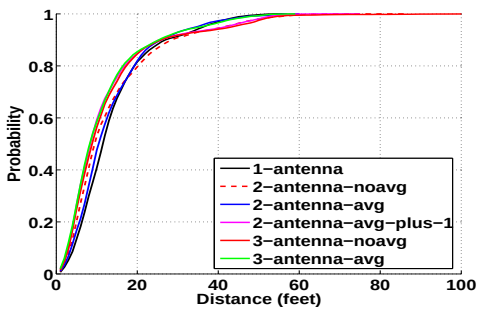
(e) M_3, (x, y) plane: N=101, NA=1



(f) M_3, z-axis: N=101, NA=1



(g) M_3, (x, y) plane: N=101, NA=50



(h) M_3, z-axis: N=101, NA=50

Figure 4.13. Localization stability of Bayesian networks M_1, M_3.

Chapter 5

Conclusions

In this thesis we focused on two issues related to indoor localization: (a) reducing localization latency when using a particular method that relies on Bayesian Networks, (b) improving the absolute accuracy of different localization algorithms by alleviating the impact of environmental effects on radio signal strength.

First, since the Bayesian Networks (BNs) that we use for localization do not have closed-form solutions, we resorted to simulation methods. Specifically, we implemented various Markov Chain Monte Carlo (MCMC) algorithms to infer values for the coordinates of wireless devices. We tackled the problem of reducing localization latency by introducing a new metric called relative accuracy. The metric measures the Euclidean distance of the result returned by an MCMC algorithm to a "gold standard" solution returned by a well-tested statistical package like WinBugs. So, in this work reducing latency means converging to the "gold standard" solution of WinBugs as fast as possible. We realized that the probability distributions of the coordinates are flat, and hence presented an algorithm, called "whole domain sampling", that has the best ratio of relative accuracy vs. time. In addition, the algorithm was characterized by its simplicity, making it a very attractive approach for a localization system. Via an analytic model we showed why this algorithm has better performance when compared to more complex ones.

Second, since a large number of devices are connected to wireless networks, a localization system might have to localize hundreds of them. However, although BNs have the capability of localizing multiple devices simultaneously, the MCMC methods we proposed can still take a lot of time to localize that many devices even on a well-equipped machine. Hence, we proposed two schemes to parallelize the MCMC process

and hence reduce latency in this case. By implementation in Berkeley UPC and evaluation on different computing platforms, we showed that one scheme, inter-chain, is good for long Markov chains, whereas the other, intra-chain, for short chains. The latter scheme is a good candidate for our BNs, because they do not need long chains to give good localization results. An important issue of the intra-chain is that in order to give the same relative accuracy that we get from a single thread, certain groups of variables need to be assigned to the same processor. Also, the inter-chain is not a good candidate for our BNs, because the whole domain sampling algorithm converges fast to the gold standard solution. However, for other applications, an MCMC process might not mix rapidly, hence it will require a long Markov chain, in which case inter-chain might be suitable. Moreover, we used the LogGP model of parallel computation to understand and predict the performance of the two schemes on various platforms.

Having improved the relative accuracy of BNs vs. time, we then turned to improve the absolute accuracy of various localization algorithms by alleviating the environmental effects, like reflection, diffraction and scattering, that signal strength suffers from indoors. Specifically, we proposed the deployment of multiple antennas at fixed locations and by doing a trace-driven study on an 802.11 testbed we found out that the received signal strength (RSS) from multiple antennas better fits a theoretical signal-to-distance curve. That was a clear indication that multiple antennas help average out environmental effects. We then evaluated the impact of using multiple antennas on the performance of three algorithms, namely RADAR, ABP, BNs, that use different techniques to localize. Our results showed that the accuracy and stability can be improved for all algorithms, and in some cases, the improvement can be significant. In our work, we define stability as the localization system's ability to maintain a position in the face of small-scale movements of a device. However, our results suggest, that due to the diversity of the techniques that the algorithms use, there is no clear conclusion as to whether averaging the RSS from different antennas is a preferred method or not.

In this thesis a lot of work was related to the MCMC process, which is primarily used in fields like statistics. However, unlike statisticians, we focused mainly on the computational aspect of the process rather than issues like how fast the process mixes.

Since we apply MCMC to provide location estimates, as computer scientists we want the method to be efficient in order to be valuable to a localization system. The various algorithms explored in Chapter 2 give a tradeoff of complexity and speed, and we saw that "whole domain sampling" is a very attractive algorithm because of its simplicity and the small computational cost it requires. Moreover, we realized that when solving a particular problem, there are special properties, like the flatness of probability distributions, that could be taken into consideration, which general-purpose software, like WinBugs, does not. In addition, general-purpose software might do extra work that incurs additional computational cost.

Similarly, in Chapter 3, the key idea of parallelization is that we viewed the working load of the MCMC process as a 2D array that should be distributed to the available processors. This distinguishes our approach from previous works, where parallelizing the inference process of a graphical model is seen as a problem of distributing the nodes of a graph to available processors. We believe that the latter approaches can not give good performance on different platforms.

As future work we would like to see whether the idea of "whole domain sampling" can be applied to problems of physical estimation other than localization. Examples could be estimating temperature and the volume of a convex body in d dimensions, and motion tracking. We suspect that in many problems of this kind, the probability distributions of the variables of interest will be flat, and hence the idea of "whole domain sampling" will be applicable. We also believe that the schemes of parallelism we proposed can be applied to the inference of graphical models other than BNs. So, we hope that other applications can benefit from them. Furthermore, although the main reason for using MCMC to infer location estimates from our BNs was that MCMC has provable convergence, we would like to explore the computational cost and localization accuracy of variational approximations [36] that can also estimate values for random variables in a BN. Finally, although in Chapter 4 we experimented with 3 antennas at each landmark location, we plan to study the improvements expected with the use of more antennas and also what the limiting number of antennas is where the improvements tail off.

References

[1] Crossbow Technology Inc. http://www.xbow.com.

[2] IEEE 802.11 Standards. http://standards.ieee.org/getieee802/802.11.html.

[3] IEEE 802.15.1 Standards. http://standards.ieee.org/getieee802/download/802.15.1-2003.pdf.

[4] IEEE 802.15.4 Standards. http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

[5] Moteiv Corporation. http://www.moteiv.com.

[6] D. K. Agarwal and A. E. Gelfand. Slice Gibbs Sampling for Simulation Based Fitting of Spatial Data Models. *Statistics and Computing*, 15:61–69, 2005.

[7] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. LogGP: Incorporating Long Messages into the LogP Model. *Journal of Parallel and Distributed Computing*, 44:71–79, 1997.

[8] O. Baala and A. Caminada. WLAN-based Indoor Positioning System: Experimental Results for Stationary and Tracking MS. In *Proceedings of the International Conference on Communication Technology (ICCT)*, pages 1–4, Nov. 2006.

[9] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, March 2000.

[10] M. Barrett and R. Arnott. Adaptive Antennas for Mobile Communications. *Electronics and Communication Engineering Journal*, 6, August 1994.

[11] V. A. N. Barroso, M. J. Rendas, and J. P. Gomes. Impact of Array Processing Techniques on the Design of Mobile Communications Systems. In *Proceedings of the IEEE 7th Mediterranean Electrotechnical Conference*, Antalya, Turkey, April 1994.

[12] R. Battiti, M. Brunato, and A. Villani. Statistical Learning Theory for Location Fingerprinting in Wireless LANs. Technical Report DIT-02-086, University of Trento, Informatica e Telecomunicazioni, October 2002.

[13] C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. Welcome, and K. Yelick. An Evaluation of Current High-Performance Networks. In *Proceedings of the 17th Parallel and Distributed Processing Symposium (IPDPS)*, October 2003.

[14] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, Belmont, California, 1990.

[15] Y. Chen, E. Elnahraway, J.-A. Francisco, K. Kleisouris, X. Li, H. Xue, and R. P. Martin. Grail: General Real Time Adaptable Indoor Localization. In *Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SENSYS), Demo Abstract*, November 2006.

[16] Y. Chen, J. Francisco, W. Trappe, and R. P. Martin. A Practical Approach to Landmark Deployment for Indoor Localization. In *Proceedings of the Third Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, September 2006.

[17] Y. Chen, K. Kleisouris, X. Li, W. Trappe, and R. P. Martin. The Robustness of Localization Algorithms to Signal Strength Attacks: A Comparative Study. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 546–563, June 2006.

[18] Y. Chen, Q. Yang, J. Yin, and X. Chai. Power-Efficient Access-Point Selection for Indoor Location Estimation. *IEEE Transactions on Knowledge and Data Engineering*, 18:877–888, 2006.

[19] M. Chryssomallis. Smart Antennas. *IEEE Antennas and Propagation Magazine*, 42(3), June 2000.

[20] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 1–12, 1993.

[21] D. E. Culler, L. T. Liu, R. P. Martin, and C. O. Yoshikawa. LogP Performance Assessment of Fast Network Interfaces. *IEEE Micro*, February 1996.

[22] B. D'Ambrosio, T. Fountain, and Z. Li. Parallelizing Probabilistic Inference: Some Early Explorations. In *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 59–66, 1992.

[23] L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex Position Estimation in Wireless Sensor Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, AK, April 2001.

[24] A. C. Dusseau. Modeling Parallel Sorts with LogP on the CM-5. Technical Report UCB//CSD-94-829, University of California, Berkeley, Department of Electrical Engineering and Computer Science, 1994.

[25] E. Elnahrawy, J.-A. Francisco, and R. P. Martin. Adding Angle of Arrival Modality to Basic RSS Location Management Techniques. In *Proceedings of IEEE International Symposium on Wireless Pervasive Computing (ISWPC)*, February 2007.

[26] E. Elnahrawy, X. Li, and R. P. Martin. The Limits of Localization Using Signal Strength: A Comparative Study. In *Proceedings of IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, CA, October 2004.

[27] P. Enge and P. Misra. *Global Positioning System: Signals, Measurements and Performance*. Ganga-Jamuna Pr, 2001.

[28] X. Feng, D. A. Buell, J. R. Rose, and P. J. Waddell. Parallel Algorithms for Bayesian Phylogenetic Inference. *Journal of Parallel and Distributed Computing*, 63:707–718, 2003.

[29] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian Filters for Location Estimation. In *IEEE Pervasive Computing (special issue on Dealing with Uncertainty)*, volume 2, pages 24–33, 2003.

[30] L. C. Godara. Applications of Antenna Arrays to Mobile Communications, Part I: Performance Improvement, Feasibility, and System Considerations. *Proceedings of the IEEE*, 85(7), July 1997.

[31] M. Goldburg and R. H. Roy. The Impacts of SDMA on PCS System Design. In *Proceedings of the IEEE 3rd Annual International Conference on Universal Personal Communications*, San Diego, CA, 1994.

[32] H. Hashemi. The Indoor Radio Propagation Channel. *Proceedings of the IEEE*, 81(7), July 1993.

[33] M. Hazas and A. Ward. A High Performance Privacy-Oriented Location System. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, Dallas, TX, March 2003.

[34] D. Hensgen, R. Finkel, and U. Manber. Two Algorithms for Barrier Synchronization. *International Journal of Parallel Programming*, 17(1), 1998.

[35] J. Hightower, C. Vakili, G. Borriello, and R. Want. Design and Calibration of the SpotON Ad-Hoc Location Sensing System, unpublished., 2001.

[36] T. S. Jaakkola and M. I. Jordan. Bayesian Parameter Estimation via Variational Methods. *Statistics and Computing*, 10(1), January 2000.

[37] K. Kaemarungsi and P. Krishnamurthy. Modeling of Indoor Positioning Systems Based on Location Fingerprinting. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1012–1022, March 2004.

[38] K. Kaemarungsi and P. Krishnamurthy. Properties of Indoor Received Signal Strength for WLAN Location Fingerprinting. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*, pages 14–23, Aug. 2004.

[39] A. V. Kozlov and J. P. Singh. A Parallel Lauritzen-Spiegelhalter Algorithm for Probabilistic Inference. In *Proceedings of Supercomputing*, November 1994.

[40] A. V. Kozlov and J. P. Singh. Parallel Implementations of Probabilistic Inference. *Computer*, 29:33–40, 1996.

[41] P. Krishnan, A. S. Krishnakumar, W.-H. Ju, C. Mallows, and S. Ganu. A System for LEASE: Location Estimation Assisted by Stationary Emitters for Indoor RF Wireless Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, October 2004.

[42] A. Kushki, K. N. Plataniotis, and A. N. Venetsanopoulos. Kernel-Based Positioning in Wireless Local Area Networks. *IEEE Transactions on Mobile Computing*, 6:689–705, 2007.

[43] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach. Robotics-Based Location Sensing using Wireless Ethernet. In *Proceedings of The Eighth ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, GA, September 2002.

[44] C. Ladha, B. S. Sharif, and C. C. Tsimenidis. Mitigating Propagation Errors for Indoor Positioning in Wireless Sensor Networks. In *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pages 1–6, Oct. 2007.

[45] X. Li. *Characterizing and Accommodating Spatial Aspects of Wireless Networks*. PhD thesis, Rutgers, The State University of New Jersey, New Brunswick, NJ, 2006.

[46] H. Lim, L. Kung, J. Hou, and H. Luo. Zero-Configuration, Robust Indoor Localization: Theory and Experimentation. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, March 2006.

[47] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics, 2001.

[48] K. Lorincz and M. Welsh. MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. *Springer Personal and Ubiquitous Computing*, October 2006.

[49] D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. WinBUGS - A Bayesian Modelling Framework: Concepts, Structure, and Extensibility. *Statistics and Computing*, 10:325–337, 2000.

[50] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An Empirical Analysis of Radio Signal Strength Variability in IEEE 802.15.4 Networks using Monopole Antennas. Technical Report 050501, Yale Univeristy, ENALAB, 2006.

[51] D. Madigan, E. Elnahrawy, R. Martin, W. Ju, P. Krishnan, and A. Krishnakumar. Bayesian indoor positioning systems. In *Proceedings of the 24th IEEE International Conference on Computer Communications (INFOCOM)*, pages 324–331, March 2005.

[52] M. Mizuno and T. Ohgane. Application of Adaptive Array Antennas to Radio Communications. *Electronics and Communications in Japan (Part I: Communications)*, 77(2), 1994.

[53] R. M. Neal. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, September 1993.

[54] R. M. Neal. Slice Sampling (with discussion). *Annals of Statistics*, 31:705–767, 2003.

[55] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *GLOBECOM (1)*, pages 2926–2931, 2001.

[56] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, California, 1988.

[57] D. M. Pennock. Logarithmic Time Parallel Bayesian Inference. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 431–438, 1998.

[58] M. A. Peot and R. D. Shachter. Fusion and Propagation with Multiple Observations in Belief Networks (Research Note). *Artificial Intelligence*, 48:299–318, 1991.

[59] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, August 2000.

[60] T. Roos, P. Myllymaki, and H.Tirri. A Statistical Modeling Approach to Location Estimation. *IEEE Transactions on Mobile Computing*, 1(1), Jan-March 2002.

[61] T. K. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma. A Survey of Various Propagation Models for Mobile Communication. *IEEE Antennas and Propagation Magazine*, 45(3), June 2003.

[62] A. Savvides, C.-C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Proceedings of the Seventh Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, Rome, Italy, July 2001.

[63] N. Saxena, S. Sarkar, and N. Ranganathan. Mapping and Parallel Implementation of Bayesian Belief Networks. In *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing (SPDP)*, 1996.

[64] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from Mere Connectivity. In *Fourth ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MOBIHOC)*, Annapolis, MD, June 2003.

[65] A. Smailagic and D. Kogan. Location Sensing and Privacy in a Context Aware Computing Environment. *IEEE Wireless Communications*, 9(5), October 2002.

[66] D. Spiegelhalter, A. Thomas, N. Best, and D. Lunn. WinBUGS Version 1.4 User Manual. Technical report, MRC Biostatistics Unit, Institute of Public Health, UK, Department of Epidemiology and Public Health, Imperial College School of Medicine, UK, January 2003.

[67] D. J. Spiegelhalter. Bayesian Graphical Modelling: A Case-Study in Monitoring Health Outcomes. *Applied Statistics*, 47(1):115–133, 1998.

[68] S. C. Swales, M. A. Beach, D. J. Edwards, and J. P. McGeehan. The Performance Enhancement of Multibeam Adaptive Base-Station Antennas for Cellular Land Mobile Radio Systems. *IEEE Transactions on Vehicular Technology*, 39(1), February 1990.

[69] The Berkeley UPC Compiler, 2002. http://upc.lbl.gov.

[70] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[71] G. I. Wassi, C. Despins, D. Grenier, and C. Nerguizian. Indoor Location Using Received Signal Strength of IEEE 802.11b Access Point. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1367–1370, May 2005.

[72] J. H. Winters, J. Salz, and R. D. Gitlin. The Impact of Antenna Diversity on the Capacity of Wireless Communication Systems. *IEEE Transactions on Communications*, 42(234), February 1994.

[73] W.R.Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996.

[74] M. Youssef, A. Agrawal, and A. U. Shankar. WLAN Location Determination via Clustering and Probability Distributions. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, Fort Worth, TX, March 2003.

Vita

Konstantinos Kleisouris

1996 **B.S. in Computer Science,**
University of Patras, Patras, Greece

1999 **M.S. in Computer Science,**
Rutgers University, New Brunswick, New Jersey, USA

2009 **Ph.D. in Computer Science,**
Rutgers University, New Brunswick, New Jersey, USA

Selected Publications

2006 "The Robustness of Localization Algorithms to Signal Strength Attacks: A Comparative Study". Yingying Chen, Konstantinos Kleisouris, Xiaoyan Li, Wade Trappe, Richard P. Martin. In Proceeding of the International Conference on Distributed Computing in Sensor Systems (DCOSS), June, 2006.

2006 "Reducing the Computational Cost of Bayesian Indoor Positioning Systems". Konstantinos Kleisouris, Richard P. Martin. In Proceedings of the Third IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON), September, 2006.

2007 "Parallel Algorithms for Bayesian Indoor Positioning Systems". Konstantinos Kleisouris, Richard P. Martin. In Proceedings of the 2007 International Conference on Parallel Processing (ICPP), September, 2007.

2008 "A Security and Robustness Performance Analysis of Localization Algorithms to Signal Strength Attacks". Yingying Chen, Konstantinos Kleisouris, Xiaoyan Li, Wade Trappe, Richard P. Martin. In ACM Transactions on Sensor Networks (TOSN), 2008.

2008 "The Impact of Using Multiple Antennas on Wireless Localization". Konstantinos Kleisouris, Yingying Chen, Jie Yang, Richard P. Martin. In Proceedings of the Fifth IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON), June, 2008.