A Probabilistic Approach to Inference with Limited Information in Sensor Networks

Rahul Biswas
Stanford University
Stanford, CA 94309 USA
rahul@cs.stanford.edu

Sebastian Thrun
Stanford University
Stanford, CA 94309 USA
thrun@cs.stanford.edu

Leonidas J. Guibas
Stanford University
Stanford, CA 94309 USA
guibas@cs.stanford.edu

ABSTRACT

We present a methodology for a sensor network to answer queries with limited and stochastic information using probabilistic techniques. This capability is useful in that it allows sensor networks to answer queries effectively even when present information is partially corrupt and when more information is unavailable or too costly to obtain. We use a Bayesian network to model the sensor network and Markov Chain Monte Carlo sampling to perform approximate inference. We demonstrate our technique on the specific problem of determining whether a friendly agent is surrounded by enemy agents and present simulation results for it.

1. INTRODUCTION

Recent advances in sensor and communication technologies have enabled the construction of miniature nodes that combine one or more sensors, a processor and memory, plus one or more wireless radio links [9, 12]. Large numbers of these sensors can then be jointly deployed to form what has come to be known as a sensor network. Such networks enable decentralized monitoring applications and can be used to sense large-area phenomena that may be impossible to monitor in any other way. Sensor networks can be used in a variety of civilian and military contexts, including factory automation, inventory management, environmental and habitat monitoring, health-care delivery, security and surveillance, and battlefield awareness.

In some settings, sensors enjoy the use of external power sources and need not factor the energy costs of computation, sensing, and communication into their decisions. Information theory tells us that more information is always better [4] and any algorithm analyzing sensing results should be able to perform at least the same and hopefully better with more information. In most actual deployment scenarios however, sensors must operate with limited power sources and carefully ration their energy utilization. In this world, computation, sensing, and communication comes at a cost and algorithms designed for sensor networks must strike a good balance between increasing network longevity by decreasing costs on the one hand and providing useful answers to queries on the other.

IPSN'04, April 26–27, 2004, Berkeley, California, USA.

One aspect in which algorithms used for sensor networks must differ from traditional algorithms is that they must be able to reason with incomplete and noisy information. Some areas of interest may lie outside sensor range. Any information, if available, is not free but has a sensing cost and comes at the expense of decreased network longevity. Also, real sensors do not always operate as they should and any algorithm that deals with sensor information must be able to handle partial corruption of the input data.

We present an algorithm that leverages both limited and stochastic sensor data to probabilistically answer queries normally requiring complete world state to answer correctly. By answering probabilistically, i.e. by assigning a probability to each possible answer, we provide the inquirer with more useful information than only one answer whose certainty is not known. This technique is applicable to a broad range of queries and we demonstrate its effectiveness on an open problem in sensor networks, that of determining whether a friendly agent with known location is surrounded by enemy agents with unknown but stochastically sensable locations [7].

We structure the probabilistic dependencies between sensor measurements, enemy agent locations, and the query of interest as a Bayesian network. We then enter the sensor observations into the Bayesian network and use Markov Chain Monte Carlo (MCMC) methods to perform approximate inference on the Bayesian network and extract a probabilistic answer to our original query.

Our experimental results demonstrate the effectiveness of our inference algorithm on the problem of determining surroundedness on a wide variety of sensor measurements. The inference algorithm consistently provides accurate estimates of the posterior probability of being surrounded. It demonstrates the ability to incorporate both positive and negative information, to use additional information to disambiguate almost colinear points, to cope effectively with sensor failure, and provide useful prior probabilities of being surrounded without any sensor measurements at all.

2. PROBLEM DEFINITION

2.1 Overview

We seek to determine whether an agent F is surrounded by N enemy agents $E_1 \dots E_N$. Let the location of agent F be given as LF and the locations of agents $E_1 \dots E_N$ be given by $LE_1 \dots LE_N$ respectively. Let the assertion D refer to the condition that LF lies in the convex hull generated by points $LE_1 \dots LE_N$, which we will henceforth refer to in aggregate as simply LE.

We assume that LF and N are known. We have no prior knowledge over the LE but form posteriors over them from the evidence gathered by the sensor measurements.

2.2 Sensor Model

Sensors can detect the presence of enemy agents in a circular region surrounding the sensor. We have a total of M sensors $S_1 \dots S_M$ at our disposal. Let the sensor locations be given by $LS_1 \dots LS_M$. For a given sensor S_i, let $\hat{d}_i$ equal 1 if there exists one or more enemy agents inside the circle with its center collocated with the sensor and radius R, and 0 otherwise. In other words, $\hat{d}_i$ is a boolean variable that is the response sensor S_i should obtain if it is operating correctly. Let d_i be the value sensor S_i actually obtains and shares with other sensors. The variable d_i depends stochastically on $\hat{d}_i$ as follows:

$$d_i = \begin{cases} \hat{d}_i & : \text{ with probability } \zeta \\ 0 & : \text{ with probability } \frac{1-\zeta}{2} \\ 1 & : \text{ with probability } \frac{1-\zeta}{2}. \end{cases} \tag{1}$$

Setting ζ to 1 is equivalent to having perfect sensors and setting ζ to 0 is equivalent to having no sensors. Typically, ζ will be quite high since sensors usually tend to work correctly.

2.3 Objective Function

Our objective is to determine, given LF and a set of P stochastic sensor measurements $R_1 \dots R_P$, the posterior probability of D. The set of sensor measurements may be empty.

We do not seek to address the issues of network control and communication in this paper. While these are important topics and are revisited in Section 6.2, our inference algorithm works independently of the control algorithm that decides who will sense and manages inter-sensor communication.

3. PRIOR WORK

The traditional approach to solving this sort of problem is to determine the values of each of the variables of interest and then use standard techniques to answer the query given the full world state. The potency of our approach lies in its ability to take partial world state and estimate the complete world state in expectation. By doing so, we can use the same off-the-shelf techniques the other algorithms can. In fact, in the example we have chosen to showcase, that of determining whether a friendly agent is surrounded, we use an unmodified traditional computational geometry algorithm and simply plug it into our inference. This allows us to probabilistically answer any other query just as easily by just plugging in existing algorithms requiring full world state into the core of our MCMC sampler.

For the specific problem at hand, we can determine whether the friendly agent lies within the convex hull of enemy agents by starting with the exact values of LE, computing their convex hull, and then determining whether LF lies within this convex hull. Many well-known computational geometry

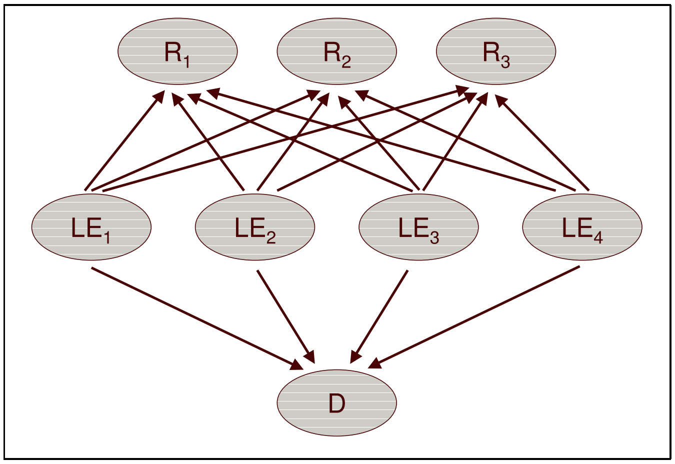

Figure 1: Bayesian Network with 3 sensor measurements and 4 enemy agents

algorithms exist that compute the convex hull of N points in the plane in $\Theta(N \cdot logN)$ time [16].

In fact, convex hull inclusion can be tested directly in $O(N)$ time. Let CH represent the convex hull of LE and let $CH_1 \dots CH_S$ represent the enemy agents, in order, lying on this hull. S is less than or equal to N. LF lies within CH if and only if either of the following conditions holds:

$$\forall s \in S, CCW(CH_s, CH_{s+1}, LF) \tag{2}$$

or

$$\forall s \in S, CW(CH_s, CH_{s+1}, LF). \tag{3}$$

Here, CH_{S+1} wraps around to CH_1, $CCW(a, b, c)$ refers to the assertion that point c is strictly counterclockwise to ray ab, and $CW(a, b, c)$ refers to the assertion that point c is strictly clockwise to ray ab.

Closest in spirit to our effort are techniques that try to approximate a convex body using a number of probes [1], or those that try to maintain approximate convex hulls in a data stream of points, using bounded storage [8]. These works both address similar problems but their methods cannot be applied to our problem at hand.

4. OUR APPROACH

We show in this section how the query, in this case, whether the friendly agent is surrounded by enemy agents can be represented by a variable in a Bayesian network. The probabilistic relationships between sensors, enemy agent locations, and the query are faithfully represented by the Bayesian network. Thus the problem of answering the query reduces to computing the posterior probability of the variable in the network representing our query. We use MCMC sampling to perform this inference. We start with the formal definition of our Bayesian network, identify why exact inference is not possible, and then discuss our MCMC implementation.

4.1 Bayesian Network

A Bayesian network [11] is characterized by its variables and the prior and conditional probabilities relating the variables to one another. We start with the variables and what they represent and then move on to their conditional probabilities. Figure 1 shows a Bayesian Network for the case of three sensor measurements and four enemy agents. In practice, the network structure will vary depending on the number of sensor measurements and number of enemy agents.

4.1.1 Variable 1: Sensor Measurements

Let there be P sensor measurement variables $R_1 \dots R_P$ in our Bayesian network. These variables are identical to the variables mentioned in Section 2 and contain both the index of the sensor it originated from and the stochastic value d that the sensor perceived. These variables are observed in the Bayesian network.

4.1.2 Variable 2: Enemy Agent Locations

Let there be N enemy agent location variables $LE_1 \dots LE_N$ in our Bayesian network. These variables are also identical to the variables mentioned in Section 2. The variables are not observed in the Bayesian network, i.e., their values are not known to the inference algorithm. By doing inference however, it is possible to form posterior probability distributions over them.

4.1.3 Prior Probability: Enemy Agent Locations

We assume that the enemy agents are independently and uniformly distributed across a bounded rectangular region.

4.1.4 Variable 3: Whether the Friendly Agent is Surrounded

Let there be a variable D in our Bayesian network. Like the variables that came before it, this variable is identical to the variable mentioned in Section 2. Indeed, the potency of our Bayesian network lies in the simplicity of its conception. This variable is also not observed in the Bayesian network but it is possible to form a posterior probability distribution over it as well.

4.1.5 Conditional Probability 1: How Enemy Agent Locations Affect Sensor Measurements

Sensor measurements are stochastically determined by enemy agent locations as presented in Section 2.

4.1.6 Conditional Probability 2: How Enemy Agent Locations Affect the Query

The value of D is deterministic given $LE_1 \dots LE_N$. We use the off-the-shelf convex hull generation and containment algorithm described in section 3 to establish this relation between D and $LE_1 \dots LE_N$.

4.2 Problems with Exact Inference

Inference in Bayesian networks refers to the process of discovering posterior probabilities of unobserved variables given the values of observed variables. A family of techniques known as clique tree methods is used for this purpose.

Unfortunately, our Bayesian network has both continuous (LE) and discrete (R and D) variables. In fact, this is the hardest class of Bayesian networks to perform inference on [10]. Moreover, the nature of the problem preempts the use of multivariate Gaussians to approximate the LE variables while maintaining the rich hypotheses inference required to form accurate posteriors. Faced with such a network, clique tree methods would form an immense clique containing each R variable along with each of the LE variables. The inadequacy of multivariate Gaussians would force us to perform discretization of the joint space. The exponential blowup that arises from this enormous state space would render the problem computationally intractable.

4.3 MCMC for Approximate Inference

Fortunately, we can get an accurate estimate of the posterior probability of interest, $P(D|R_1 \dots R_P)$. We do this with the Metropolis algorithm, an instance of a MCMC method that performs approximate inference on Bayesian networks. It is equipped to handle the continuous nature of this distribution as it samples evenly over the entire joint space.

We follow the treatment of Metropolis and use the terminology as presented in [5]. The function $f(X)$ we want to estimate is the variable D and the space X_t we sample from is the joint distribution over LE. Let us choose to draw T samples using the Metropolis algorithm and let the hypothesis over LE while taking sample t be LE^t. The starting state often systematically biases the initial samples so we discard the first $\iota \cdot T$ samples and estimate our posterior probability of interest as the mean of the remaining samples:

$$P(D|R_1 \dots R_P) = \frac{1}{(1-\iota) \cdot T} \sum_{t=\iota \cdot T}^{T} P(D|LE^t) \qquad (4)$$

where ι is the fraction of samples discarded until the Markov chain converges.

There are four aspects to evaluating this expression. The first three deal with the inner workings of Metropolis. First, where do we start the chain? Second, how does LE^t transition to LE^{t+1}? Third, how do we decide whether to accept the proposed transition? Fourth, how do we evaluate $P(D|LE^t)$?

4.3.1 Starting the Chain

We place each of the enemy agents at a random location uniformly distributed over the rectangular region where enemy agents are known to be located.

4.3.2 Proposal Distribution

One of the enemy agents is chosen at random and is moved to a random location uniformly chosen over the same rectangular region as before. Thus, each component of LE^{t+1}, that is, each enemy agent location, is conditionally independent, given LE^t. This condition is required for Metropolis to work correctly.

4.3.3 Determining Proposal Acceptance

We use the Metropolis proposal acceptance criterion:

$$\alpha(LE^t, LE^{t+1}) = min(1, \frac{P(LE^{t+1}, R_1 \dots R_P)}{P(LE^t, R_1 \dots R_P)}). \qquad (5)$$

The proposal is accepted with probability $\alpha(LE^t, LE^{t+1})$ as given above. This is easily done by generating a random number U uniformly drawn from between 0 and 1 and accepting the proposal if $U \le \alpha(LE^t, LE^{t+1})$.

We can simplify the expression

$$\frac{P(LE^{t+1}, R_1 \dots R_P)}{P(LE^t, R_1 \dots R_P)} \qquad (6)$$

substantially. We discover from the Bayesian network that the expression $P(LE^t, R_1 \ldots R_P)$ decomposes as follows:

$$\prod_{i=1}^{N} P(LE_i^t) \prod_{j=1}^{P} P(R_j|LE^t) \quad (7)$$

where LE_i^t refers to the hypothesized location of enemy agent i when Metropolis takes sample t.

Similarly, the expression $P(LE^{t+1}, R_1 \ldots R_P)$ decomposes as:

$$\prod_{i=1}^{N} P(LE_i^{t+1}) \prod_{j=1}^{P} P(R_j|LE^{t+1}). \quad (8)$$

Thus, we can rewrite 6 as:

$$\frac{\prod_{i=1}^{N} P(LE_i^{t+1}) \prod_{j=1}^{P} P(R_j|LE^{t+1})}{\prod_{i=1}^{N} P(LE_i^{t}) \prod_{j=1}^{P} P(R_j|LE^{t})}. \quad (9)$$

If we recall that LE_i^t is uniformly distributed and that each of the LE terms cancel out in both the numerator and denominator, this expression simplifies to:

$$\frac{\prod_{j=1}^{P} P(R_j|LE^{t+1})}{\prod_{j=1}^{P} P(R_j|LE^{t})}. \quad (10)$$

Our Bayesian network does not directly allow us to simplify this expression further but since sensors are not affected, even stochastically, by enemy agents outside their sensing range, we can exploit context-specific independence to reduce this expression further. Specifically, only one of the LE_i variables has changed value from step t to step $t+1$. Only sensors for which this enemy agent is within range will be affected by this transition. Let there be K such sensors and let K_k refer to the index of the k-th such sensor.

The sensor measurements that are not affected by the change in LE_i cancel out in the numerator and denominator, leaving us with:

$$\prod_{k=1}^{K} \frac{P(R_{K_k}|LE^{t+1})}{P(R_{K_k}|LE^{t})}. \quad (11)$$

Computing this expression is extremely fast and is further sped up in our implementation by discretizing the rectangular region and associating with each grid cell a list of sensor measurements that affect that cell.

The conditional probabilities of the sensor measurements given the enemy agent locations can be computed directly from the expressions of the conditional probability. This formulation of the problem, which relies only on the conditional probabilities as they are stated, i.e. the forward model, and not on evidential reasoning make computation of likelihoods and thus inference easier to implement, faster, and more accurate [17].

4.3.4 Computing $P(D|LE_t)$

This conditional probability can be computed directly from the definition as presented in 4.1.6. The locations of the enemy agents are given by LE_t. The expression will yield a probability that is either 0 or 1 but not in between.

4.4 Alternate Queries

Up to this point, we have explored our algorithm for the specific query of determining whether a friendly agent is surrounded by enemy agents. This problem dealt with determining properties of the convex hull of a set of stochastically sensable points. While we present the scenario in 2D, it generalizes directly to higher dimensional spaces.

We briefly note two other scenarios where we can use our algorithm to discover useful geometric properties of a set of points. The first scenario explores discovering both the extent and area of a set of stochastically sensable points. The second deals with finding a point maximally distant in expectation from all other points. These scenarios were chosen to explore different areas of computational geometry but our algorithm works on any query where small changes in the input induce small changes in the output.

4.4.1 Extent and Area of a Point Set

The problem of finding the farthest point along a certain direction in a point set is more generally known as the extremal polytope query and can be solved efficiently by Kirkpatrick's algorithm [16]. Let us define a new variable B that contains both the minimal and maximal point in each direction. We replace the last step of our algorithm, computing $P(D|LE_t)$, with computing $P(B|LE_t)$. This conditional probability is deterministic, as its predecessor was. The expected value of the extents are the mean of the components of the individual samples. The expected value of the area is the mean of the areas stemming from the individual samples and will typically not equal the area stemming from the mean extents. Using only discovered points in this scenario will tend to underestimate the extent of the point set.

4.4.2 Maximally Distant Point

The problem of finding a point that is maximally distant from its nearest neighbor stems from the Voronoi diagram of a point set and can be computed as described in [6]. Let us define a new variable E that uses a discretized grid to represent the distance of each cell to its nearest neighbor in the point set. By replacing the last step with computing $P(E|LE_t)$ (again deterministic), we calculate the expected distance from each cell and choose the one with the highest expected distance. Using only discovered points in this scenario will tend to ignore the effects of undiscovered points on the solution.

5. EXPERIMENTAL RESULTS

We show in this section some examples of the sort of results the inference algorithm produces. First, we have some specific networks and show the posterior probabilities produced conditioned on different sets of sensor measurements. Second, we show how the average posterior probabilities vary as different parameters of the sensor network varies. Third, we examine how many iterations MCMC requires to converge.

5.1 Specific Inference Examples

In Figure 2, we see a sensor network. Here, LF corresponds to the friendly agent and is designated by a plus sign, LE_i corresponds to the ith enemy agent and is designated by a circle, and LS_j the jth sensor and is designated by an asterisk. The circle surrounding a sensor shows the extent of its sensing range. The labels and circles for only a subset

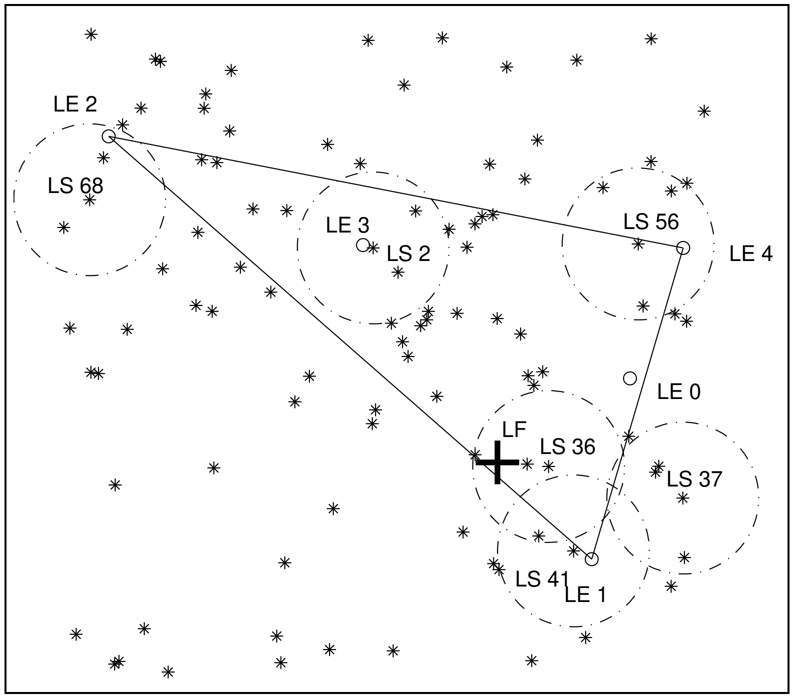

Figure 2: An Example Sensor Network

of sensors are drawn to reduce clutter in the diagram. The following table provides the posterior probabilities resulting from inference when different sets of sensors are chosen to sense. None of the sensors failed during the simulation runs but the algorithm runs with ζ set to 0.01.

Posterior Probabilities for Figure 2	
Sensors Sensing	Posterior Probability
56, 68	44
none	51
2, 41	80
41	81
41, 56, 68	86
36, 37, 41, 56, 68	93

We analyze our results from a geometric perspective. It is important to note that our inference algorithm approaches the problem in a fundamentally different way than the discussion that follows.

Let us start with the prior probability that the agent is surrounded without any sensor measurements. In this instance, the entire field is 100 by 100 and the friendly agent is at 58, 40. Ignoring collinearity, which is of measure 0 anyway, the friendly agent is only surrounded if all of the enemy agents do not lie to one side of it. Let us analyze the case that all the enemy agents are to the north of the friendly agent. The probability of any one of them being there is 60% and the probability of all five of them being there is approximately 8%.

The alternative scenario is that each of the agents is to the south. The probability of any one of them being there is 40% and the probability of all five of them being there is approximately 1%. Thus the probability of not being surrounded because all the agents lie to one side of the east-west meridian is approximately 9%. We need to extend this case to each line passing through LF but without double counting. For example, the probability of all of the enemies lying to one side of the north-south meridian is also approximately 9% because of the approximate symmetry of the situation. We do not seek to compute the prior via this method but only to show where the prior comes from and demonstrate the difficulty of computing it.

We next consider the situation where sensors 56 and 68 have sensed and found enemy agents but the posterior has decreased to 44%. This may seem counterintuitive that we have useful information and our posterior has dropped but with a fixed number of enemies, finding two of them in inessential locations decreases the probability of enemies elsewhere. Specifically, the probability of enemy agents to the south and east of the friendly agent, where they were already less likely to be, is further reduced. Contrast this with the situation where only sensor 41 has sensed. Finding an enemy where it was unlikely but more potentially useful increases the posterior probability substantially, in this case to 81%.

Let us next consider the case where sensors 2 and 41 have both sensed. The posterior probability is lower than the scenario where only 41 has sensed. Intuitively, the sensor is almost collinear to the previously detected sensor and friendly agent and the discovery of this enemy is neither immensely helpful nor it useless. It does however reduce the number of unknown enemies that might have been used to expand the convex hull in a useful direction.

The next scenario we consider, where sensors 41, 56, and 68 have sensed, is perhaps the most intuitive. We can see that the discovered enemies form a triangle that just encloses the friendly agent. The sensors have less information than we do though – they only know that enemy agents lie within their sensing radius. It may be that both enemy agents lie north of the sensors that have located them. If this were the

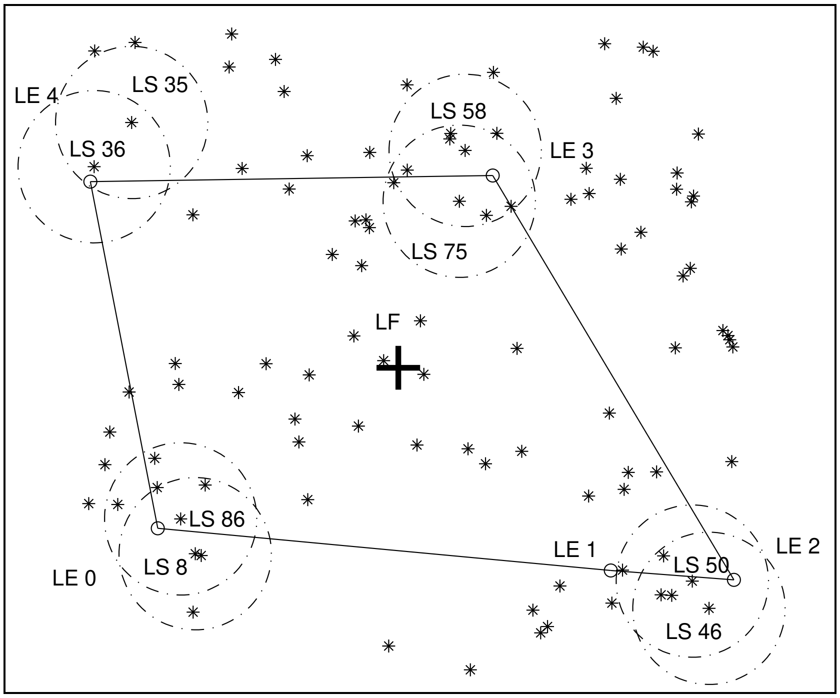

Figure 3: Another Example Sensor Network

case, the friendly agent would not longer be surrounded.

The last scenario, where sensors 36, 37, 41, 56, and 68 have sensed, shows the powerful effect of negative information. The addition of results from sensors 36 and 37 demonstrate that the enemy agent that sensor 41 found is not in the sensable region of either sensor 36 and 37 and therefore must lie farther south, where it is more likely to participate in surrounding the friendly agent.

Another example sensor network and inference results follow. In this case, none of the sensors failed during the simulation runs but the algorithm now runs with ζ set to 0.2. We will see how this difference affects inference.

Posterior Probabilities for Figure 3	
Sensors Sensing	Posterior Probability
none	67
35, 50, 58	70
8, 36, 58	74
8, 36, 46	75
8, 35, 50, 58	78
8, 35, 36, 46, 50, 58, 75, 86	91

There is much to be learned from these results but we focus on the three essential points that were not demonstrated by the previous example. First, the prior probability of being surrounded is much higher. This is due solely to the fact that the friendly agent is closer to the center of the region where enemy agents are distributed.

Second, even when the sensor evidence supports the assertion that the friendly agent is surrounded, as in the case where sensors 8, 35, 50, and 58 have sensed, the posterior probability is not as high as one may expect. The sensor network realizes due to the relatively high value of ζ that it is likely that one or more of these sensor measurements are inaccurate and considers these possibilities in its inference.

Third, when there is evidence from multiple sensors supporting the presence of an enemy agent, as in the case where sensors 8, 35, 36, 46, 50, 58, 75, and 86 have all sensed and each enemy agent is covered by at least two sensors, the posterior probability increases substantially.

5.2 Effects of Changing Network Parameters

We now move on to aggregate results of how different parameters of the scenario affect the posterior probability of being able to detect that the friendly agent is surrounded. It is possible to perform such analysis for any query and can provide valuable insights into how the parameters of the sensor network help or hinder in answering specific types of queries.

Figure 4 shows the aggregate of two sets of ground truth scenarios. The Surrounded case shows examples where the agent is actually surrounded. The Not Surrounded case shows examples where the agent is actually not surrounded. The distinction is determined by the simulator and not through sensor measurement. The chart thus shows how the average posterior of the inference algorithm across a variety of scenarios evolves. The number of enemies is set to four and the sensing radius is set to 12. As we see, more information brings the posterior probability closer and closer to the ground truth.

Figure 5 shows the same scenario but with the number of enemies varying now. The number of sensors is set to twenty-five and the sensing radius remains at twelve. We see that it is easier to locate enemies and believe that one is surrounded when there are more enemies present but that the effect is not strong.

Figure 6 shows the same scenario but with the range of the sensor varying now. The number of sensors is set to twenty-five and the number of enemies is set at four. While it is easier to locate enemies with increased sensing range,

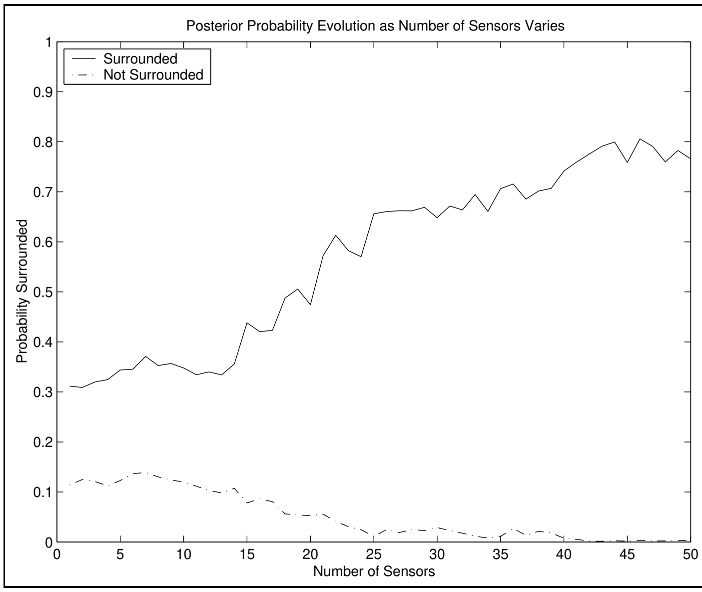

Figure 4: Posterior Probability Evolution as Number of Sensors Varies

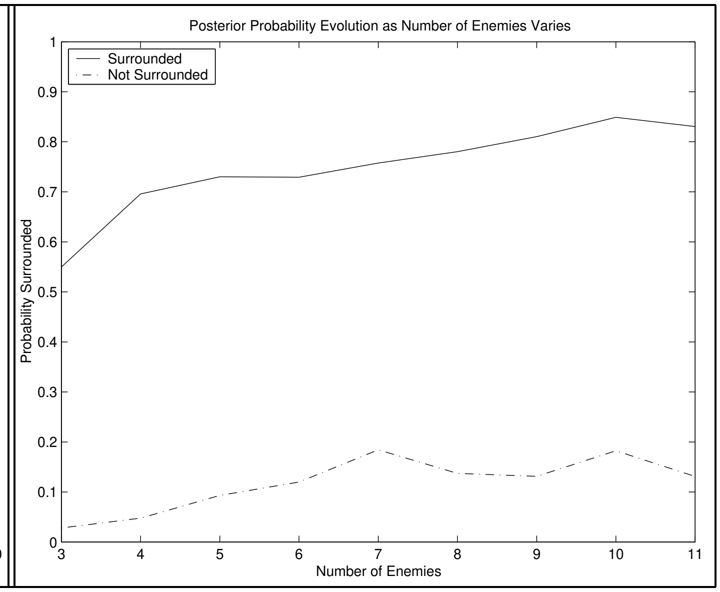

Figure 5: Posterior Probability Evolution as Number of Enemies Varies

the uncertainity associated with such information makes this information less useful. These contrary forces appear to balance out and varying the sensor range appears to neither help nor hurt inference.

5.3 MCMC Convergence

Although MCMC converges in the limit to the exact posterior, it is useful to know how many iterations are required in practice to obtain accurate results. It is unfortunately only feasible to compute analytical bounds for MCMC algorithms for only the simplest distributions though and the conditions required by most bounding theorems are not satisfied by general distributions. However, it often suffices in practice to answer this question heuristically through experimentation [15]. Figure 7 shows the standard deviation of the MCMC posterior estimate errors before convergence occurs. Regardless of the number of queries, we find that the posterior estimate errors level off after approximately 15,000 steps of MCMC, a computation that takes only a few hundred milliseconds to perform.

6. FUTURE WORK

While we feel that we have made significant progress with the methods presented in this paper, there are two major directions we plan to proceed with this work. First, we want to use this algorithm with a set of actual sensors. Second, we want the sensor network to guide its own sensing activities and direct sensing in such a way to minimize both sensing and communication cost. We discuss these ideas in turn.

6.1 Moving from Simulation to Actual Sensors

We plan to use this algorithm to help mobile robots playing laser tag with one another [14] determine where their enemy agents may or may not be. For example, if a robot knew that a certain corridor was most likely free of enemy agents, it could focus its searching efforts elsewhere.

Specifically, we plan to use vibration sensors that will detect the absence or presence of enemy robots and route this information to the friendly agents to determine where enemies may or may not be. These sensors will be subject to noise as they will sometimes miss a robot going by or mistake a person walking by for an enemy robot.

6.2 Intelligent Sensor Selection

While our algorithm can effectively leverage very limited information, to run on an autonomous sensor network, it needs to be paired with a decentralized algorithm that runs on each sensor node that decides when to sense and also when and what type of information to communicate to its neighbors.

Such an algorithm will need to make tradeoffs between energy conservation and answering queries more effectively. One way to address this problem is a cost-based scheme that casts utility and battery depletion into a single currency [3]. While the cost of sensing, asking another node to sense, or sending information to another node will fall directly out of a coherent energy-aware sensor model, the value of such information is intricately tied to the inference algorithm. Specifically, to prevent bias, the value of sensing should be the expected reduction in entropy stemming from the acquisition of this information.

Constructing such an algorithm requires answering several difficult questions. First, how can sensors with only partial world state display utility-maximizing emergent behavior? Second, what is the tradeoff between expending more energy and acquiring a more accurate answer? Third, how can sensors distribute work in a way that takes into account the effects of remaining power at the granularity of individual nodes?

7. CONCLUSION

We have presented an inference algorithm for a sensor network to effectively answer generalized queries. Using probabilistic techniques, the algorithm is able to answer these queries effectively with only partial information over

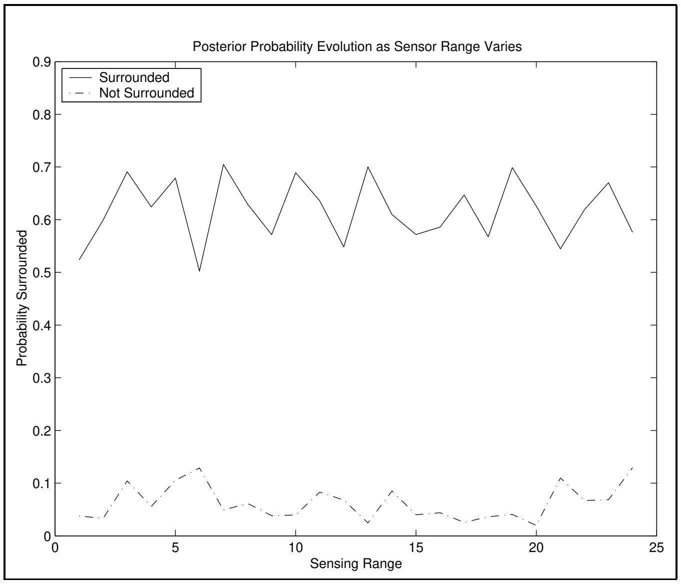

Figure 6: Posterior Probability Evolution as Sensing Range Varies

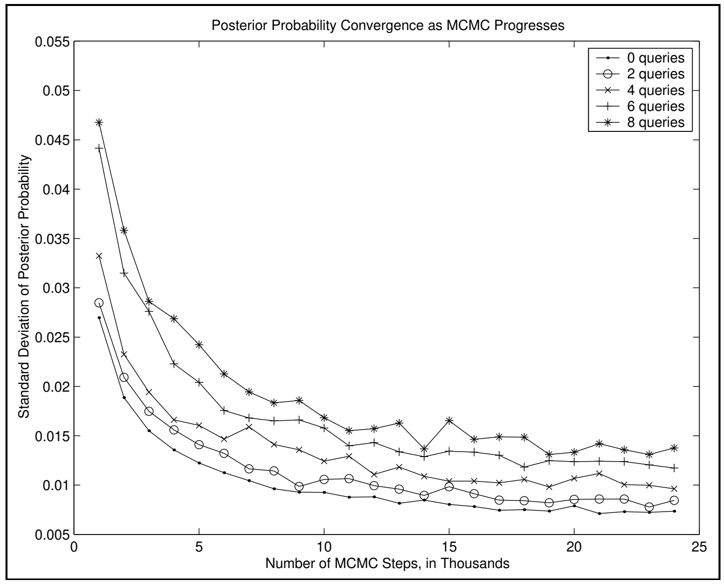

Figure 7: Posterior Probability Convergence as MCMC Converges

the world state. Moreover, it can reason coherently despite the stochastic nature of the sensor data. The results shown demonstrate that the inference algorithm is highly accurate in the scenario presented and is able to handle a wide range of types of sensor information.

8. ACKNOWLEDGEMENTS

This research is supported in part by DARPA's MARS Program (Contract number N66001-01-C-6018), DARPA's Sensor Information Technology Program (Contract number F30602-00-C-0139), the National Science Foundation (CAREER grant number IIS-9876136 and regular grant number IIS-9877033), Intel Corporation, the Stanford Networking Research Center, ONR MURI grant N00014-02-1-0720, and a grant from the DARPA Software for Distributed Robotics program under a subcontract from SRI, all of which is gratefully acknowledged. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the United States Government or any of the sponsoring institutions.

9. REFERENCES

[1] G. Battista, R. Tamassia, and I. Tollis. Constrained Visibility Representation of Graphics Proc. IPL, 1992.

[2] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[3] J. Byers and G. Nasser. "Utility-based decision-making in wireless sensor networks" Proc. IEEE MobiHOC Symposium, 2000

[4] T. Cover and J. Thomas *Elements of Information Theory* John Wiley and Sons, Inc., 1991.

[5] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall / CRC, 1996.

[6] L. Guibas, and J. Stolfi. "Primitives for the manipulation of general subdivisions and the computation of Voronio diagrams". ACM Trans. Graph. 4, 74-123. 1985.

[7] L. Guibas. "Sensing, Tracking and Reasoning with Relations". IEEE Signal Processing Magazine. Volume: 19 Issue: 2, March 2002.

[8] J. Hershberger and S. Suri. "Convex Hulls and Related Problems in Data Streams". Proc. ACM SIGMOD/PODS Workshop on Management and Processing of Data Stream, 2003.

[9] J. Kahn, R. Katz, and K. Pister. "Mobile networking for smart dust" Proc. ACM/IEEE Intl. Conf. on Mobile Computing and Networking, 1999.

[10] U. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems* Ph.D. Thesis, Stanford University, October 2002.

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

[12] G. Pottie and W. Kaiser. "Wireless integrated network sensors." Proc. ACM Communications, 2000.

[13] T. Richardson. "Approximation of planar convex sets from hyperplane probes". Proc. Discrete and Computational Geometry, 1997.

[14] M. Rosencrantz, G. Gordon, and S. Thrun. "Locating moving entities in indoor environments with teams of mobile robots." Proc. AAMAS-2003.

[15] J. Rosenthal. "Minorization Conditions and Convergence Rates for Markov Chain Monte Carlo." Proc. Journal of the American Statistical Association, 1995.

[16] J. Rourke *Computational Geometry in C* Cambridge University Press, 1998.

[17] S. Thrun. "Learning occupancy grids with forward models". IEEE Conference on Intelligent Robots and Systems. 2001.