On the Interdependence of Sensing and Estimation Complexity in Sensor Networks

Yaron Rachlin
rachlin@ece.cmu.edu

Rohit Negi
negi@ece.cmu.edu

Pradeep Khosla
pkk@ece.cmu.edu

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

Computing the exact maximum likelihood or maximum a posteriori estimate of the environment is computationally expensive in many practical distributed sensing settings. We argue that this computational difficulty can be overcome by increasing the number of sensor measurements. Based on our work on the connection between error correcting codes and sensor networks, we propose a new algorithm which extends the idea of sequential decoding used to decode convolutional codes to estimation in a sensor network. In a simulated distributed sensing application, this algorithm provides accurate estimates at a modest computational cost given a sufficient number of sensor measurements. Above a certain number of sensor measurements this algorithm exhibits a sharp transition in the number of steps it requires in order to converge, leading to the potentially counter-intuitive observation that the computational burden of estimation can be reduced by taking additional sensor measurements.

Categories and Subject Descriptors: H.1.1 [Systems and Information Theory]: Information Theory; C.3 [Special-purpose and Application-based Systems]: Signal processing systems; E.4 [Coding and Information Theory]: Error Control Codes

General Terms: Algorithms, Theory, Performance

Keywords: sensor networks, sequential decoding, estimation, detection

1. INTRODUCTION

Obtaining accurate estimates of the environment using noisy sensor measurements poses a significant computational challenge in many sensor network applications. One of the primary reasons for these high computational costs is that sensor output is typically affected by multiple parts of the environment at the same time. For example: seismic sensors sense a nonlinear function of vibrations from multiple targets, temperature sensors can be affected by multiple heat sources, and the time of flight measurements of a sonar range sensor are affected by multiple reflective surfaces in the environment. In this paper we present a novel tradeoff between the number of sensor measurements and computational complexity of estimation. To demonstrate this tradeoff, we describe an algorithm that can take advantage of additional sensor measurements to *reduce* the amount of computation required to obtain an accurate estimate. The algorithm we present is based on extending the concept of sequential decoding used in error correcting codes to the context of sensor networks. We focus on distributed sensing applications where the environment being estimated can be modelled as a discrete field. Such environments include a wide array of practical applications (e.g. distributed detection and classification tasks).

Obtaining the maximum likelihood (ML) or maximum a posteriori (MAP) estimate of a discrete field based on sensor measurements can be very computationally expensive due to the large scale dependencies that sensor measurements induce in the environment. As a result, different parts of the environment must be estimated jointly. Without exploiting any structure, a brute force approach to computing ML or MAP estimates is exponential in the size of the estimated field, and therefore typically impractical. In order to exploit probabilistic structure, the complex dependencies induced in the environment by various sensor measurements can be captured using the formalism of graphical models. In such graphical models, message passing based inference algorithms have been proposed by many researchers as a means for combining sensor measurements into an accurate global estimate [13],[12],[3]. Such algorithms suffer from exponential complexity in either the largest degree of the graph or clique size in an induced graph. Thus, such algorithms are promising for sensors that interact with a small number of random quantities, such as estimating temperature and bias in a temperature sensor [13]. However, many sensors have significant range and thus induce precisely such large degrees and cliques. This occurs because the output of such sensors is affected by multiple parts of the environment. For example, the Senscomp sonar[1], a sensor popularly used in robotics, emits a wide acoustic pulse over a range of 35 feet and uses the time of flight of this pulse to estimate distance. This sensor has a low angular resolution, and thus the reflection of the pulse is affected by the presence or absence

[1]Senscomp 600 Series Instrument Transducer: www.senscomp.com

of obstacles in a large area. In a graphical model where the environment is modelled as a discrete field, such a sensor would have a large degree since it is connected to every node in the environment that can affect the acoustic pulse. Similarly, sensor deployments where multiple sensors observe the same location in the environment produce nodes with large degrees and induce large cliques in a graphical model.

Faced with the computational complexity of estimating discrete fields using sensor measurements, a sensor network designer is typically confronted with a choice between accurate algorithms that incur high computational complexity, and algorithms that are less accurate but are significantly easier to compute. In practice, inaccurate but computationally efficient algorithms (e.g. occupancy grids [4]) are combined with high quality but expensive sensors (e.g. SICK laser range finders[2]) to achieve good estimation results. Accurate but computationally expensive algorithms can enable accurate sensing with cheap sensors such as Sencomp sonars, but can be too computationally intensive for real time implementation (e.g. [14],[20]). In this paper, we argue that this tradeoff between the complexity and accuracy of estimation algorithms can be altered by increasing the number of sensor measurements. We present a novel family of algorithms based on a connection between decoding convolutional codes [8] and estimation in sensor networks. Interestingly, for a sufficient number of sensor measurements, our algorithms show sharp empirical performance transitions, becoming both computationally efficient and accurate.

The idea of extending techniques used to efficiently decode convolutional codes to the problem of estimation in sensor networks was motivated by the connection between error correcting codes in communications and sensor networks demonstrated in our previous work [16],[17],[18]. This connection allowed us to define and bound the 'sensing capacity' of sensor networks, a quantity analogous to the channel capacity of a communications channel. The channel capacity of a communications channel [2] is the maximum rate at which one can transmit messages through a noisy communications channel such that the probability of decoding error at the receiver is close to zero. The rate of communication is defined as the ratio of bits in the message to the number of bits into which the message is encoded. Our work on sensing capacity was based on modelling a sensor network as an encoder, where a message corresponds to a state of the environment and the sensor network acts as the message encoder. For a particular state of the environment, the vector of noiseless sensor outputs is the codeword associated with that state. The sensing capacity is the maximum ratio of the size of the environment to the number of sensor measurements, below which accurate estimation to within a desired accuracy is possible.

We briefly contrast our work with previous and ongoing research that applies distributed source coding to sensor networks [23]. Distributed source coding provides limits on the compression of correlated outputs of multiple sensors that do not communicate with each other. In contrast to a compression problem, we focus directly on the problem of estimating the underlying state of the environment using noisy sensor observations. The notion of sensing capacity characterizes the limits that sensing (e.g. sensor type, range, and noise) imposes on the attainable accuracy of estimation. We do not examine the compression of sensor observations. Instead, we focus on the limits of estimation accuracy assuming complete availability of noisy sensor observations. Thus the problem we study is better characterized as a channel problem (though not a communications channel) than as a source coding problem.

Convolutional codes [8] constitute one of the main approaches to encoding messages for transmission in communications channels. In such codes, an encoder has a finite memory. This means that the output of the encoder depends not only on the bit at the encoder input at that time, but also on a finite number of immediately preceding input bits. Convolutional codes achieve low probability of error for large memory encoders, where accurate ML decoding using algorithms such as the Viterbi algorithm [21] becomes computationally intractable. One of the techniques used to overcome the computational costs of decoding convolutional codes with large memory is sequential decoding. The idea of sequential decoding was first introduced in [22]. This approach accurately decodes messages in practice by relying on a sufficient number of redundant transmitted bits, that is for some rate below the channel capacity. Theoretical results about sequential decoding indicate the existence of this performance regime for some computational cutoff rate below the channel capacity [6]. Below this computational cutoff rate, the average computational effort required to decode a bit remains bounded by a constant that does not depend on the overall message length. Perhaps the most striking feature of sequential decoding algorithms for convolutional codes is that their computational complexity is constant in the size of the memory. Another good property is the comparative simplicity of their implementation.

Our sensor network model shares a fundamental similarity with convolutional codes. While a convolutional encoder has a causal structure not shared by sensor networks, a similar finite memory property exists in the context of spatially distributed sensing. In sensor networks, a portion of the environment is encoded not only based on its value, but also based on the values of nearby parts of the environment. The geometry of a sensor's field of view and the sensor location determine which subsets of the environment are jointly encoded. Similarly to the computational difficulty faced when decoding convolutional codes with large memory, sensors of large range induce many dependencies in the environment that render precise ML or MAP estimation too computationally expensive for practical use. Given this connection between convolutional codes and sensor networks, we were motivated to investigate whether the computational challenge of estimation (i.e. decoding) in sensor networks could be overcome by taking advantage of additional sensor measurements. In the context of the concepts of capacity and computational cutoff rate, taking additional measurement corresponds to operating sufficiently below the sensing capacity to enable low complexity estimation. The empirical results presented later in this paper indicate a sharp transition in the computational complexity of our algorithm as a function of the number of sensor measurements.

We will explore the tradeoff between the number of sensor measurements, accuracy of estimation, and computational complexity. We compare our algorithms to occupancy grids [4] and belief propagation [15], two well-known approaches for estimating a discrete field using sensor measurements. Empirical results demonstrate that the computational com-

[2]SICK website: http://www.sick.com/ecatalog/en.html

plexity of our sequential detection algorithms becomes linear in the size of the field above a certain number of sensors measurements. For a range sensor example, we empirically demonstrate that while belief propagation has exponential complexity in sensor range, sequential decoding has linear complexity in sensor range. Despite this disparity in complexity, the two algorithms achieved a similar accuracy in our simulations.

We will begin with an overview of related approaches and problems in Section 2. In Section 3 we discuss the idea of sequential decoding, and demonstrate its extension to distributed sensing. In Section 4 we show simulation results that characterize sequential decoding performance and computational costs, compare sequential decoding to two other commonly used estimation algorithms, and demonstrate estimation performance in a simulated application of mapping using range sensors. We conclude and discuss future work in Section 5.

2. RELATED WORK

We review a number of related approaches to using sensor measurements for estimating environments modelled as a set of random variables. Since such sensor networks can be modelled using the formalism of graphical models [12],[3], we will examine related work according to the type of inference it represents in a graphical model. Inference in graphical models can be broken into three large categories: exact, sampling, and variational [9]. Exact algorithms compute the exact probabilities of variables in a graphical model by exploiting graph structure. An example of such a method, the junction tree algorithm, was applied by [13] to various problems in sensor networks. Algorithm such as the junction tree algorithm have a computational complexity that is exponential in the largest clique size induced while constructing a junction tree from the original graph. Sampling approaches use ideas such as importance sampling and MCMC methods to conduct inference in graphical models [9]. [1] applies such an MCMC method to the problem of answering queries in a sensor network modelled as a Bayesian network. Finally, the variational approach recasts inference as an optimization problem for a typically simplified model [9]. One example of this approach is loopy belief propagation, which [3] applied to inference in sensor networks. Belief propagation has complexity that is exponential in the largest degree of in the graph. [5] uses sampling-based approximations of belief propagation messages to apply belief propagation to the problem of sensor self-calibration in sensor networks.

In addition to these graphical model approaches, we mention one of the most commonly used algorithms in robotics for using sensor measurements (e.g. sonar and laser range sensors) to estimate environments that can be modelled as discrete fields. Occupancy grids [4] solve the estimation problem by assuming a simpler model that eliminates many of the dependencies in the original graphical model. While this results in a loss of accuracy, it also yields very low computational complexity.

Across the various approaches to estimation described above, we have not seen an emphasis on the tradeoff between sensing and estimation complexity, which is the primary focus of our paper. We will compare the performance of our sequential decoding algorithm to loopy belief propagation and occupancy grids as a function of the number of sensor measurements and sensor range. However, the primary focus of this paper is not to exhaustively compare sequential decoding to other algorithms. Instead, we would like to emphasize a novel tradeoff between sensing and estimation complexity, to further establish the connection between error correcting codes and sensing, and to provide results that will demonstrate the promise of a sequential decoding approach to estimation in sensor networks.

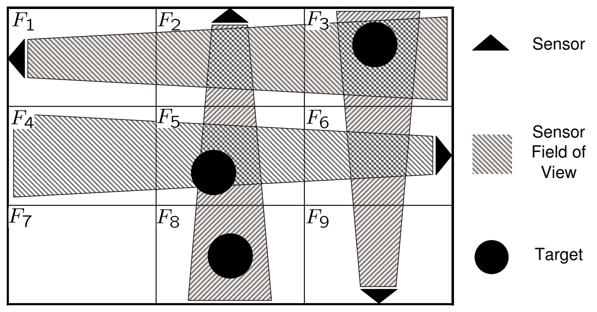


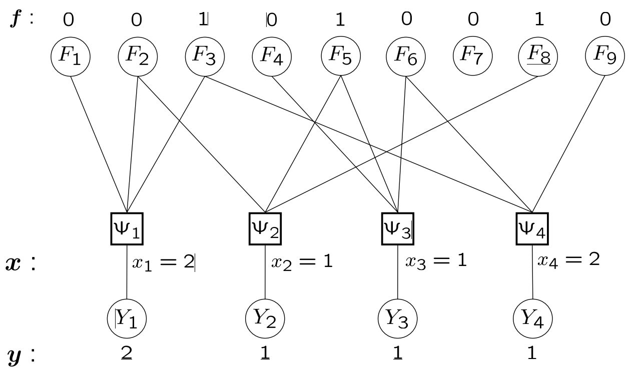
Figure 1: Simple deployment of three range sensors in a 3×3 grid world.

3. SEQUENTIAL DECODING FOR ESTIMATION

In distributed sensing, an estimation algorithm seeks to identify the ML or MAP state of the variables which comprise an environment using a set of sensor measurements. The main insight behind sequential decoding, translated to the context of distributed sensing, is that for sufficiently low noise conditions (achievable using low noise sensors, or multiple sensor measurements), most of the effort expended by exhaustively considering all potential states of the environment is spent on considering very unlikely states. Instead, a more efficient strategy is to explore only the most promising states of the environment. To take advantage of this insight, a sequential decoding algorithm sequentially guesses the variables in the environment, evaluates partial guesses using a path metric, and generates new hypotheses based on extending the hypothesis with the highest path metric.

We consider the Stack algorithm, a sequential decoding algorithm introduced separately by [7] and [24] to decode convolutional codes. We will present this algorithm in a form that we adapted to the problem of estimation in distributed sensing, and we will discuss the most salient differences between its traditional application in convolutional codes and in our adaptation for distributed sensing. The stack algorithm as used for convolutional decoding cannot be trivially applied to a sensing problem due the different structures of the two problems. While convolutional encoders typically have a natural ordering in time, that is, they can operate from left to right on the time axis, there is no such obvious ordering when sensing an environment. Further, the structure of the encoder itself is far less restricted in distributed sensing due to the typically uneven distribution of sensor measurements, and the wide array of possible sensor types.

We begin by reviewing a simple distributed sensing scenario as illustrated in Figure 1. This figure depicts a 3×3 environment, where each grid block may contain either nothing or a target. Four sensors are placed in this environment, each with its own corresponding view of the environment. In general, a discrete field is represented by a set of variables



f :
0 0 1 0 1 0 0 1 0
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9
Ψ_1 Ψ_2 Ψ_3 Ψ_4
x :
$x_1 = 2$ $x_2 = 1$ $x_3 = 1$ $x_4 = 2$
Y_1 Y_2 Y_3 Y_4
y :
2 1 1 1

Figure 2: Graph representation of the simple scenario presented in Figure 1.

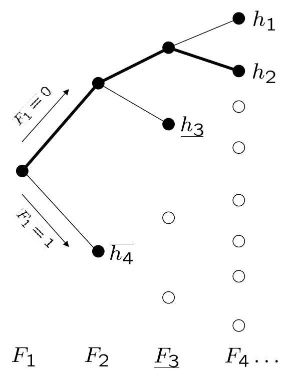


Figure 3: Illustration of the operation of the stack algorithm.

$F_1, \ldots, F_k$. A MAP estimation algorithm seeks to identify the maximum probability assignment of $F_1, \ldots, F_k$ given the observed sensor measurements. We assume that F_i is a binary random variable, corresponding to the presence or absence of a target, though extensions to non-binary targets are possible. To understand the operation of our algorithm, we first translate the sensor deployment depicted in Figure 1 into a graph as shown in Figure 2. The graph consists of the field variables that are to be estimated, the noisy sensor output variables that are observed, and the sensors, shown as squares. The field variables that correspond to a sensor's field of view, referred to as the sensor's scope, are shown by connections in the graph between the sensor and those variables. For example, the scope $\mathcal{S}_1$ of sensor 1 in the graph is $\{F_1, F_2, F_3\}$. The sensor ℓ outputs x_ℓ, which is a function Ψ_ℓ of its scope. For example, $x_1 = \psi_1(f_1, f_2, f_3)$ in Figure 2. In the example shown in Figures 1 we assume that the sensors are simple range sensors. These sensors output the distance to the nearest target. However, we don't observe the noiseless sensor output x_ℓ of sensor ℓ, and instead we observe the noisy sensor output y_ℓ. In the range sensor example, this corresponds to observing a distance that is greater than or less than the actual distance with some probability. The noise model of the sensor is characterized by the probability distribution $P_{Y|X}$.

The algorithm sequentially generates hypotheses about variable values. This process can be viewed as a search on a tree, as shown in Figure 3. Each level of the tree corresponds to a variable F_i. From each node in the tree (except for leaf nodes) there are two possible paths, corresponding to assigning a value of 0 or 1 to the variable F_i corresponding to that node. A path h_p of length k_p in the tree represents a sequence of variable assignments over the first k_p variables. For example, in Figure 3 the path $\boldsymbol{h}_1$ corresponds to assigning $F_1 = 0, F_2 = 0, F_3 = 0$ and the path $\boldsymbol{h}_2$ corresponds to assigning $F_1 = 0, F_2 = 0, F_3 = 1$. The tree represents an exponential number of paths, and therefore exhaustive exploration of all paths is not computationally feasible. We therefore ask the question, which path in the tree is the most promising to extend? Since the paths are not of equal length, this cannot be answered by simply extending the path which makes the sensor observations most likely. The reason for this is that long paths will tend to have a lower likelihood, and thus only shorter paths will be extended, leading to an exhaustive exploration of the tree. Instead, we derive a path metric using the approach used to derive the Fano metric in the sequential decoding of convolutional codes [8]. Differences between the coding and sensing settings induce different path metrics, which is one of the primary differences in the application of the stack algorithm to the problem of distributed sensing. The Fano path metric can be derived using Bayes rule by evaluating the posterior probability of a particular path in a tree conditioned on the structure of the convolutional encoder and the noisy received bits. Similarly, we can derive our path metric by evaluating the posterior probability of a path $\boldsymbol{h}_p$ through a tree, conditioned on the sensor deployment $\mathcal{D}$ (e.g. sensor locations, orientations, fields of view), and the noisy sensor observations $\boldsymbol{y}$. We denote our path metric as μ, and we describe its derivation below. In this paper we focus on fields in which each variable F_i is i.i.d. with $P_F(F_i = 1) = t$ and $P_F(F_i = 0) = 1 - t$. Using Bayes rule, the posterior probability that the first k_p variables, denoted F^{k_p}, equal $\boldsymbol{h}_p$ given $\mathcal{D}$ and $\boldsymbol{y}$ is written as,

$$P(F^{k_p} = \boldsymbol{h}_p|\mathcal{D}, \boldsymbol{y}) = \frac{P(\boldsymbol{y}|\mathcal{D}, \boldsymbol{h}_p)P(\boldsymbol{h}_p|\mathcal{D})}{P(\boldsymbol{y}|\mathcal{D})} \tag{1}$$

We assume the sensor network deployment is independent of the estimated field, and therefore $P(\boldsymbol{h}_p|\mathcal{D}) = P(\boldsymbol{h}_p)$. We denote the number of ones and zeros in path $\boldsymbol{h}_p$ as k_p^1 and k_p^0. Using our independence assumptions about the field variables we can write $P(\boldsymbol{h}_p) = (1-t)^{k_p^0} t^{k_p^1}$. In order to make the path metric computationally inexpensive we make the approximation that the denominator factors as $P(\boldsymbol{y}|\mathcal{D}) = \prod_{i=1}^{k} P_Y(y_i|\mathcal{D})$. Similarly, we assume that sensor observations in the term $P(\boldsymbol{y}|\mathcal{D}, \boldsymbol{h}_p)$ factor in a similar manner. Conditioning on a path $\boldsymbol{h}_p$ allows use to group terms in this factorization according to sensor scope. For any path $\boldsymbol{h}_p$ the set of sensors is split into the sets $\{\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3\}$. $\mathcal{Y}_1$ represents the set of sensors whose scope is completely specified by the path, $\mathcal{Y}_2$ represents the set of sensors whose scope is partially specified by the path, and $\mathcal{Y}_3$ corresponds to the set of sensors whose scopes are completely not specified in the current path. Using these definitions, we approximate the term $P(\boldsymbol{y}|\mathcal{D}, \boldsymbol{h}_p)$ as below,

$$P(\boldsymbol{y}|\mathcal{D},\boldsymbol{h}_p) \approx \prod_{i_1\in\mathcal{Y}_1} P(y_{i_1}|\boldsymbol{h}_p,\mathcal{D}) \prod_{i_2\in\mathcal{Y}_2} P(y_{i_2}|\boldsymbol{h}_p,\mathcal{D}) \cdot \prod_{i_3\in\mathcal{Y}_3} P(y_{i_3}|\mathcal{D}) \quad (2)$$

The terms of the first product in (2) can be rewritten to reflect the fact that sensor scopes in the product are completely specified by the path $\boldsymbol{h}_p$. Therefore, we can write $P(y_{i_1}|\boldsymbol{h}_p,\mathcal{D}) = P_{Y|X}(y_{i_1}|x_{i_1})$. Defining $\mu(\boldsymbol{h}_p,\mathcal{D},\boldsymbol{y}) = \log P(F^{k_p} = \boldsymbol{h}_p|\mathcal{D},\boldsymbol{y})$, and using the simplifications and approximations described above, we write μ as follows,

$$\mu(\boldsymbol{h}_p,\mathcal{D},\boldsymbol{y}) \doteq \sum_{i_1\in\mathcal{Y}_1} \log\frac{P_{Y|X}(y_{i_1}|x_{i_1})}{P_Y(y_{i_1}|\mathcal{D})} + \sum_{i_2\in\mathcal{Y}_2} \log\frac{P(y_{i_2}|\boldsymbol{h}_p,\mathcal{D})}{P_Y(y_{i_2}|\mathcal{D})} + k_p^0\log(1-t) + k_p^1\log(t) \quad (3)$$

As we can see in equation (3), the path metric consists of two main sums. The first sum, is identical to the Fano path metric [8] used in sequential decoding for convolutional codes. This term is typically interpreted as the information gain provided by a particular transmitted bit, or in our case a noisy sensor observation. The second sum, over the set of sensors with partially specified inputs, arises due to the manner in which sensors encode the environment such as the non-unidirectional influence of sensor observations. Though this term has the same general form as the first sum, it causes the heaviest computational burden of the algorithm due to the unspecified sensor inputs. Though in the worst case, the complexity of this term is exponential in the number of unspecified grid blocks in a sensor's range, for many realistic sensor models this term has complexity that corresponds to the size of a sensor's output alphabet. For example, for range sensors (e.g. sonar, infrared, laser range finders) this term can be computed in time linear in a sensor's range. The reason for this is that the range is completely determined by the first solid obstacle to which the sensor has a clear line of sight. Our experimental results in Section 7 reflect this observation.

The path metric μ allows the Stack algorithm to compare partial paths in the tree. The Stack algorithm is written in pseudocode below.

Stack Algorithm

```
1. Load stack with the root of the tree and the metric zero.

2. Remove the top path in the stack, and insert its successors with their associated metrics into the stack. Sort the stack.

3. If the top path in the stack leads to a leaf of the tree, return with the top path as the estimate. Otherwise go to step 2.
```

Figure 4 depicts the operation of the stack algorithm in decoding the graph shown in Figure 2, leading to the partially explored tree depicted in Figure 3. In the algorithm's first step, the algorithm begins at the root of the tree. The root has only two successors in the tree $F_1 = 0$ and $F_1 = 1$. Each path is evaluated using our path metric μ. The paths are then sorted using their path metric values. The path with the highest path metric, is extended by guessing further. These new paths are stored in the stack, evaluated using the path metric, and sorted in order to decide which path to extend. The algorithm is halted when a path reaching a leaf node of the tree has the best value in the stack. It is important to note that while choosing which path to extend is done greedily, sorting of the stack according to the path metric values allows for backtracking in the tree exploration.

One important difference between the coding and sensing settings that we have thus far not addressed arises due to the irregular structure of sensor measurements as compared to the regularity of a convolutional code. A convolutional code can be decoded by proceeding through the received codeword in a unidirectional manner. Due to the structure of the code, the number of noisy observations per decoded bit is constant. However, no such directionality or regularity exists in the context of distributed sensing. It is therefore not clear how one should schedule the guessing of the variables $F_1,\ldots,F_k$. While one can propose a number of intuitive heuristics for constructing such a schedule, choosing an optimal schedule appears to be a hard problem. In the experiments described in this paper we used the simple heuristic of guessing the most frequently observed grid blocks first.

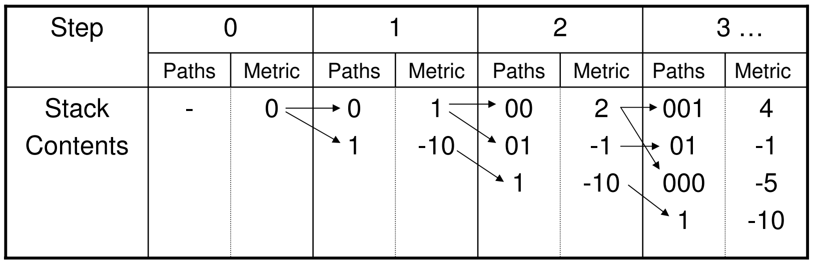
4. SIMULATIONS

4.1 Sequential Decoding Performance

To characterize the performance of our sequential decoding-based estimation algorithm we conducted a series of simulation-based experiments in Matlab. In our experiments, we sensed a fixed size $k \times k$ grid using a variable number of sensor measurements n. Using our sequential decoding algorithm we produced an estimate of the environment and plotted the algorithm's performance as a function of the number of measurements. For each value of n, we repeated our experiment 500 times. The experiment is described in greater detail below.

In each of our experiments, the simulator instantiates a random 20×20 grid world. Each grid block in the environment is independently instantiated as a solid object with probability 0.2. A fixed number of sensors are then dropped on the obstacle-free grid blocks of this environment, and are given a random orientation in one of the four cardinal directions. Given this fixed sensor configuration, each sensor takes a range measurement corresponding to the distance to the nearest obstacle. The output of these range measurements is then corrupted by noise. The noise corrupts the range measurements with a probability of 0.1, with the probability assigned to the reported distance decaying as a Gaussian from the actual range measurement value.

Figure 5 depicts average distortion as a function of the number of sensor measurements and sensor range. Distortion is computed as the fraction of the 400 grid blocks which were misclassified (i.e. free space '0' classified as an obstacle '1' and vice versa). With an increasing number of sensor measurements, the average accuracy of the estimates obtained increases. This figure indicates that the sequential decoding algorithm is capable of combining noisy sensor measurements to produce accurate estimates. It is interesting to note that the difference in sensing accuracy between



Step	0		1		2		3 …	
	Paths	Metric	Paths	Metric	Paths	Metric	Paths	Metric
Stack Contents	-	0	0	1	00	2	001	4
			1	-10	01	-1	01	-1
					1	-10	000	-5
							1	-10

Figure 4: Illustration of the operation of the stack algorithm in decoding the graph depicted in Figure 2, leading to the partially explored tree depicted in Figure 3.

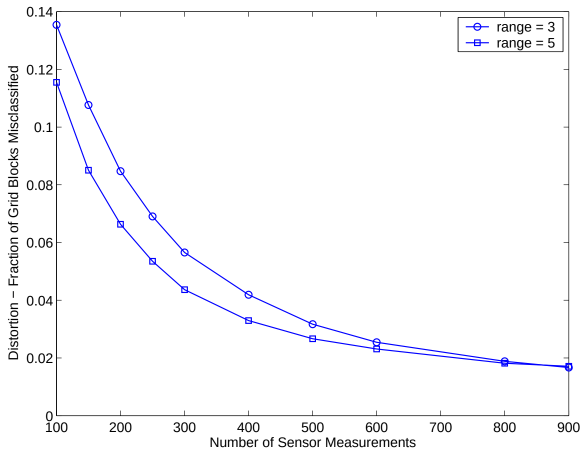


Figure 5: Sequential decoding distortion as a function of the number of sensor measurements, for sensors of two different maximum range values.

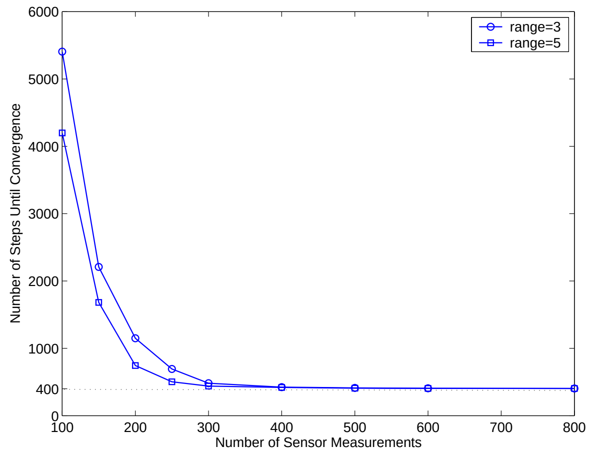


Figure 6: Average number of steps in which the sequential decoding algorithm converged as a function of the number of sensor measurements, for sensors of two different maximum range values.

the short and long range sensors decreases as the number of sensor measurements increases. At 900 measurements, shorter range sensor actually have a lower distortion than the longer range sensors. To understand this result, we note that range sensors have an output alphabet that grows linearly with range, while the possible states of grid blocks in their field of view grows exponentially. As a result, larger range sensors become essentially uninformative above a certain number of sensor measurements, while shorter range sensors continue to be informative. Neither type of sensor seems to approach a distortion of 0. This seems reasonable given the limited ability of range sensors to distinguish among grid block configurations.

Figure 6 illustrates the dependence of the complexity of sequential decoding on the number of sensor measurements. A Stack algorithm converges when a path that extends to a leaf of the tree (i.e. a path that specifies all environment variables) achieves the highest path metric in the stack. This figure plots the average number of steps the sequential decoding algorithm required in order to converge as a function of the number of sensor measurements. This graph depicts two interesting phenomena. First, it demonstrates a threshold behavior in the algorithm's complexity. Below three hundred sensor measurements (for sensors of range 3) the number of steps required for convergence grows very rapidly. Above this number of measurements, the number of steps required for convergence approaches the number of variables that are to be estimated in the environment. The transition in computational complexity occurs at 250 measurements for sensor of range 5. This figure thus demonstrates an empirically observed sharp transition in computational complexity as a function of the number of sensor measurements. Another interesting observation is that for sensors of longer range, the algorithm required fewer steps in order to converge. Drawing on our work on the sensing capacity of a sensor network, we conjecture that this occurs because sensors of longer range yield a higher sensing capacity than shorter range sensors [16]. Therefore, for the same number of measurements, sensor networks using longer range sensors are operating further below their sensing capacity resulting in better sequential decoding performance.

4.2 Comparison with Occupancy Grids and Belief Propagation

We compare our sequential decoding algorithms to two widely used estimation algorithms. The first algorithm, occupancy grids [4], is a computationally efficient algorithm for estimating an environment based on sensor measurements. It is widely used in robotics for applications such as mapping environments using range sensors [19], and in other sensing applications such as pedestrian detection using heat sensors [10]. Occupancy grids model the world as a discrete grid, where grid blocks are independently and identically distributed. This algorithm avoids the problem of jointly estimating the grid blocks by estimating the state of each grid block independently of the other grid blocks. The second algorithm, loopy belief propagation [15], is an ap-

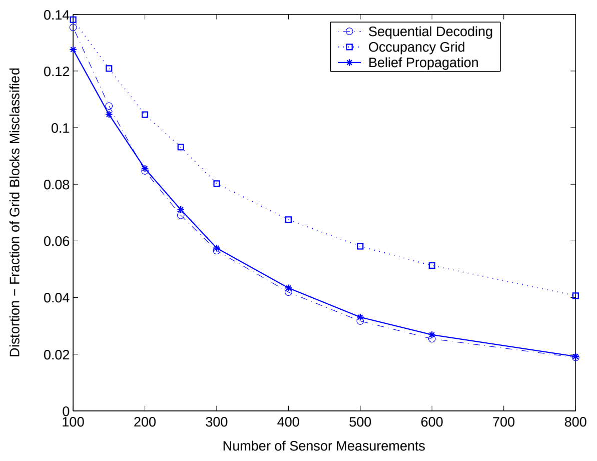


Figure 7: Average distortion of Belief Propagation, Occupancy Grids, and Sequential Decoding as a function of the number of sensor measurements.

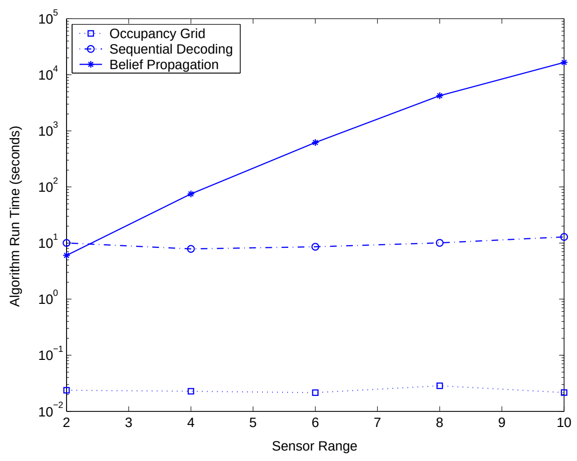


Figure 8: Average running time of Belief Propagation, Occupancy Grids, and Sequential Decoding as a function of sensor range.

proximate algorithm for conducting inference in a graphical model. This algorithm has been applied to a wide variety of applications, including error correcting codes [11] and sensor networks [3].

To compare sequential decoding to the performance of these two algorithms, we conduct experiments similar to those presented in Section 4.1. In our experiments, we sensed a randomly generated size 20×20 grid using a variable number of sensor measurements n. For a given set of noisy sensor observations, we used sequential decoding, occupancy grids, and beliefs propagation to estimate the environment. For each value of n, we repeated our experiment 500 times. In Figure 7 we plot the performance of the three algorithms as a function of the number of sensor measurements. For the same number of measurements, occupancy grids were the least accurate among the three. This is not surprising, since the independence assumptions they make are quite strong, and ignore many of the dependencies induced by sensor observations. The belief propagation algorithm and sequential decoding algorithm exhibit similar accuracy, with sequential decoding being slightly more accurate on average for more than 150 sensor observations, and less accurate below that number. The similar accuracies of belief propagation and sequential decoding algorithms are achieved with significantly disparate computational costs.

Figure 8 depicts the running times of the three algorithms as a function of sensor range. The results were obtained using the same type of simulations used to obtain Figure 7, with the number of sensor measurements fixed at $n = 400$. Occupancy grids run significantly faster than the other two algorithms. This computational efficiency, achieved by strong independence assumptions which also cause low accuracy, is one of the primary reasons for the algorithm's popularity in practical applications. In contrast, the running time of the belief propagation algorithm increased exponentially with sensor range. This occurs because the range of the sensor is directly proportional to its degree in the graphical model, and belief propagation's complexity is exponential in this degree. As a result, though it achieves good accuracy, this algorithm is only computationally efficient for sensors of small range. The sequential decoding approach is more expensive than occupancy grids, as can be expected since it doesn't make the assumption that grid blocks can be estimated independently. Sequential decoding running time grows linearly with sensor range for sensors of range four or greater. Sensors of range two result in slower running times, perhaps as a result of their lower sensing capacity (see discussion of Figure 6).

Taken together, Figures 7 and 8 demonstrate that for a sufficient number of sensor measurements, the sequential decoding approach can provide estimates as accurate as belief propagation, with a significantly smaller amount of computation. While occupancy grids run in an even smaller amount of time, the assumptions they make in order to achieve this reduced complexity significantly degrade estimation accuracy.

4.3 Mapping Examples

To demonstrate the practical potential of sequential decoding algorithms we simulated a sensing application of indoor mapping using range sensors. We compare the performance of sequential decoding to occupancy grids since they are widely used for such applications. Starting with an image of an office floor plan, we obtained a 100×100 discrete grid map depicted in the left hand side of Figure 9. We randomly placed range sensors in the empty grid blocks of this environment, chose a random orientation, and simulated noisy range sensor measurements. The range measurements were corrupted with noise, such that with probability 0.05 they reported a range measurement that was different than the measured range value. Using sensors of range 15, we collected $10,000$ such noisy range measurements. Using our sequential decoding algorithms, we produced the estimate shown in the right hand side of Figure 9 in $13,921$ iterations. An interesting complication in this simulation occurred due to the fact that range sensors are limited to observing the boundary of a solid object, and thus predictions for the interior grid blocks of large solid objects are meaningless. To make this uncertainty explicit, we checked whether the gain in our path metric equaled zero to flag grid blocks where the sequential decoding algorithm lacked evidence for choosing one grid block value over another. In Figure 9 such grid blocks are denoted using light gray. We applied the occupancy grid algorithm to the same sensor measurements in order to obtain the estimate shown in the center of Figure 9. By comparing this estimate with ground truth, we can

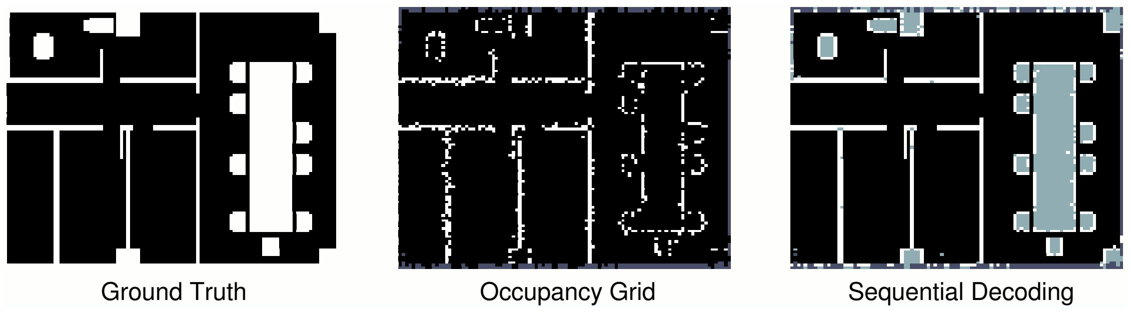


Figure 9: Mapping of a simulated indoor environment using Occupancy Grids and Sequential Decoding.

see that the occupancy-grid estimate is significantly noisier and more inaccurate than the estimate obtained by applying the sequential decoding algorithm. Further, in order to flag grid blocks about which the estimation algorithm was uncertain we applied the same idea used to flag such grid blocks in sequential decoding. If the probabilities for zero and one for a particular grid block were equal, we flagged such grid blocks as uncertain. As can be seen in Figure 9 this approach did not flag any grid blocks as uncertain. This occurs because the occupancy grid algorithm makes strong assumptions that results in incorrect probability estimates. As a result, reasoning with these probabilities requires a more ad hoc solution.

5. CONCLUSIONS AND FUTURE WORK

Our experimental results show that given sufficient sensor measurements, sequential decoding is more accurate than an approach such as occupancy grids, and can yield potentially large computational advantage over approaches such as belief propagation. We thus demonstrated that by taking advantage of redundant sensor measurements, one can obtain estimation algorithms that are both accurate and computationally efficient. We were motivated to explore this trade-off by our work on the connections between sensor networks and error correcting codes. This connection, combined with the results presented in this paper, points to a number of possibilities for future work. One potential direction is the improved application of the sequential decoding idea and the extension of other sequential decoding algorithms to the context of estimation. In convolutional codes, there are a number of interesting variants of sequential decoding that trade-off memory, computation, and accuracy. There is also a need for extending theoretical work on the computational cutoff rate for convolutional codes in order to characterize and give intuition about the convergence threshold phenomena we observed empirically in our experiments. Can we predict the number of sensor measurements above which sequential decoding becomes efficient? Finally, given the potential advantages in the centralized form of sequential decoding, and the relative simplicity of its implementation, a distributed version of this algorithm provides another interesting avenue for research.

6. REFERENCES

[1] R. Biswas, S. Thrun, and L. Guibas. A probabilistic approach to inference with limited information in sensor networks. In *Third Int. Symp. Info. Proc. in Sensor Networks*, Apr. 2004.

[2] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley-Interscience, 1991.

[3] C. Crick and A. Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proc. of the 18th Conf. on Uncertainty in Artificial Intelligence*, 2003.

[4] A. Elfes. *Occupancy grids: a probabilistic framework for mobile robot perception and navigation.* PhD thesis, Electrical and Computer Eng. Dept., Carnegie Mellon University, 1989.

[5] A. Ihler, J. F. III, R. Moses, and A. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *Fourth Int. Symp. on Info. Proc. in Sensor Networks*, 2004.

[6] I. M. Jacobs and E. R. Berlekamp. A lower bound to the distribution of computation for sequential decoding. *IEEE Trans. Inform. Theory*, 13(2):167–174, 1967.

[7] F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM J. Res. Dev.*, 13:675–685, 1969.

[8] R. Johannesson and K. Zigangirov. *Fundamentals of Convolutional Coding.* Wiley-IEEE Press, 1999.

[9] M. J. Jordan. Graphical models. *Statistical Science, Special Issue on Bayesian Statistics*, 2004.

[10] D. Linzmeier, M. Mekhaiel, J. Dickmann, and K. Dietmayer. Pedestrian detection with thermopiles using an occupancy grid. In *IEEE Int. Transportation Sys. Conf.*, 2004.

[11] R. McEliece, D. MacKay, and J. Cheng. Turbo decoding as an instance of pearl's 'belief propagation' algorithm. *IEEE J. on Sel. Areas in Comm.*, 16(2):140–152, 1998.

[12] J. Moura, J. Liu, and M. Kleiner. Intelligent sensor fusion: A graphical model approach. In *IEEE Int. Conf. on Sig. Proc.*, Apr. 2003.

[13] M. Paskin, C. Guestrin, and J. McFadden. A robust architecture for inference in sensor networks. In *Fourth Int. Symp. on Info. Proc. in Sensor Networks*, 2005.

[14] M. Paskin and S. Thrun. Robotic mapping with polygonal random fields. In *21st Conf. on Uncertainty in Artificial Intelligence*, 2005.

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

[16] Y. Rachlin, R. Negi, and P. Khosla. Sensing capacity for target detection. In *Proc. IEEE Inform. Theory Wksp.*, Oct. 24-29 2004.

[17] Y. Rachlin, R. Negi, and P. Khosla. Sensing capacity for discrete sensor network applications. In *Proc. Fourth Int. Symp. on Information Processing in Sensor Networks*, April 25-27 2005.

[18] Y. Rachlin, R. Negi, and P. Khosla. Sensing capacity for markov random fields. In *Proc. Int. Symp. on Information Theory*, 2005.

[19] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

[20] S. Thrun. Learning occupancy grids with forward sensor models. *Autonomous Robots*, 15:111–127, 2003.

[21] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13:260–269, 1967.

[22] J. M. Wozencraft. Sequential decoding for reliable communications. *IRE Convention Record*, 5:11–25, 1957.

[23] Z. Xiong, A. Liveris, and S. Cheng. Distributed source coding for sensor networks. *IEEE Signal Processing Magazine*, 21:80–94, 2004.

[24] K. S. Zigangirov. Some sequential decoding procedures. *Probl. Peredachi Inform.*, 2:13–25, 1966.