EECS 545 Project Report: Query Learning for Multiple Object Identification

Dan Lingenfelter, Tzu-Yu Liu, Antonis Matakos and Zhaoshi Meng

1 Introduction

In a typical query learning setting we have a set S of N objects and a set Q of M distinct subsets of S that is known as queries. The purpose of query learning is to identify a single active object $S_i \in S$ through as few queries from Q as possible, where a query $q \in Q$ returns 1 or 0 depending on whether or not $S_i \in q$ [1]. For example, a healthcare professional can choose from a series of tests to find the disease that is causing a patient to have symptoms. These tests, such as imaging procedures, bloodwork, biopsies, etc. usually come at a cost, so we would like to find the correct diagnosis using as few tests as possible. Current query learning techniques assume that there is only one active object, or in the diagnosis example, the patient only has one disease. This is not always accurate in practice as the patient in our example may have multiple diseases. The purpose of this work is to extend current query learning algorithms to accommodate the case of multiple active objects.

If we consider that each object has a state 1 when present and 0 otherwise, our approach will identify the most probable configuration of objects (can be thought as a bitstream of length N) where an arbitrary number of objects can be present (we can have $0 \leq n \leq N$ number of ones in the bitstream). In this setting we can see that typical query learning is a special case where we constrain the possible number of configurations (bitstreams) to the N configurations that have only $n = 1$ 1's. With this model the set Q of queries consists of subsets of objects (as in typical query learning), but the response to the query corresponds to subsets of configurations, where again with the constraint $n = 1$ 1's this reduces to subsets of objects.

The problem of finding the most probable configuration of N objects when the "answers" to all queries are known a priori is addressed in [2] as the problem of MAP estimation in graphical models. This work addresses only the case of sparse networks, since it has exponential complexity on the number of connections for each node. An approximate method presented in [3], that reduces to polynomial complexity on the number of connections, can be applied in fully connected Bayesian networks. In this work, unlike [2] and [3], we will consider the "answers" as unknown a priori and we will find the most probable configuration (MAP estimate) by actively selecting queries.

The problem of optimal query selection could be seen from an information theoretical view. We are essentially searching for a query that could produce the most reduction in the "uncertainty" of the unknown configuration of components. This is equivalent to searching the query that gives the minimum conditional entropy of the component configuration given the configuration of queries. However, exact computation of the conditional entropy in a general Bayesian network

can be intractable. A method proposed in [4] approximates the conditional entropy using a belief propagation algorithm.

In our work, based on the idea of approximating the entropy, we follow a different approach where the Max Product algorithm presented in [3] is used to approximate the marginal posterior probabilities. Then, using these marginal probabilities, we propose a simple approximation to the calculation of the conditional entropy that is used for greedy query selection. In this work we also consider the more realistic case of unreliable queries and connections between components and queries. Thus, we adopt the model of unreliable alarms and connections presented in [2].

To verify the effectiveness of the proposed method we provide some theoretical guarantees on the algorithm's performance and we also evaluate the performance experimentally through simulations on random networks and through the use of real data from the WISER chemicals database.

2 Problem Formulation

We consider a system of N components $S = \{S_1, S_2, \ldots, S_N\}$ and a set of M queries (or alarms or symptoms) $Q = \{q_1, q_2, \ldots, q_M\}$, such that each query is connected to all or a subset of the components. A component S_n is active when $S_n = 1$ and a symptom q_m is present when $q_m = 1$. The model we are investigating can be described by a bipartite graph $\mathcal{G} = (S, Q, E)$ where S and Q are the disjoint sets of components and queries respectively and the set of edges E describes the connections between components and queries. We investigate two main cases of graphs; first we consider sparse 0/1 networks where each query is connected to a small number of components and then we consider fully connected Bayesian networks where every query is connected to all components. In the Bayesian networks we associate each edge $E_{mn} \in E$ with a weight w_{mn} which is a conditional probability $w_{mn} = \Pr(q_m = 1 | S_n = 1)$. Obviously the 0/1 network is a special case of a Bayesian network where each weight is 1 when a connection is present and 0 otherwise. Alternatively we can describe the graph using a$(M \times N)$ test matrix $T = [t_{mn}]$ where each element is a weight, *i.e.* $t_{mn} = w_{mn} = \Pr(q_m = 1 | S_n = 1)$.

We use $\mathcal{S}$ to denote the configuration of the states of the N components ($\mathcal{S}$ is a bitstream of length N) and we use $\mathcal{Q}$ to denote the configuration of the M query outcomes. At any point of the process where k of the M queries have been asked we will use Q_k to denote the set containing the queries and $\mathcal{Q}_k$ to denote the combination of the outcome of these k queries. Similarly to query learning [1] we choose the subsequent query in a greedy way. The chosen query is the one that conveys the most information about the true configuration. Thus we choose the query that maximizes an appropriate performance metric $f(\mathcal{S}, q_{k+1}; \mathcal{Q}_k)$ that depends on the new query and the combination of previous outcomes $\mathcal{Q}_k$. This can be formulated as

$$q_{k+1} = \underset{q \in Q \setminus Q_k}{\operatorname{argmax}} f(\mathcal{S}, q; \mathcal{Q}_k) \tag{1}$$

In section §3 we suggest performance metrics based on posterior probabilities or conditional entropies. Exact calculation of these metrics is intractable (exponential complexity), thus we propose approximations that are easy to compute. These approximations are based on Belief Propagation [4] and the Max Product algorithm presented in [3].

3 Methods

3.1 Greedy query learning

To generalize query learning from identifying a single object to multiple objects, we replace the idea of a component with a configuration of component, i.e. the bitstream $\mathcal{S} = S_1 S_2 \dots S_M$. Therefore, the set S of all configurations contains 2^N elements. Suppose $Q_k = \{q_1, q_2, ..., q_k\}$ is the set of queries that have been proposed up until step k and $\mathcal{Q}$ is the configuration of their outcomes. A straightforward generalization to query selection as described in [1] would be to select the query q that minimizes a modified reduction factor $\rho_k(q)$. We define two types of reduction factors; a worst case (2) and an average case (3).

$$\rho_k^w(q_{k+1}) = \min_{x=0,1} \max_S \Pr(\mathcal{S}|\mathcal{Q}, q_{k+1} = x) \tag{2}$$

$$\rho_k^a(q_{k+1}) = \sum_{x=0,1} \max_S \Pr(\mathcal{S}|\mathcal{Q}, q_{k+1} = x) \tag{3}$$

Then the chosen query will be

$$q_{k+1} = \operatorname*{argmin}_{q \in Q \backslash Q_k} \rho_k(q),$$

where $\rho_k(\cdot)$ can be either the worst or average case reduction factor.

The process stops when one configuration is identified or when all relevant queries have already been asked. A query becomes "irrelevant", *i.e.* conveys no information, when all objects involved in the query are already identified as 0's or it contains at least one object identified as 1. This happens because in the former case the answer will always be 0 and in the latter it will always be 1.

As we have already mentioned, the active query selection problem is equivalent to searching for the next query that gives the most information about the unobserved components(chemicals, etc.) that remain. From and information theoretic point of view, this measure of uncertainty can be the mutual information or the conditional entropy. Therefore, we can choose the next query q_{k+1} from $Q \backslash Q_k$ such that the mutual information between $\mathcal{S}$ and q_{k+1} given $\mathcal{Q}$ is maximized, which is the same as minimizing the conditional entropy of the configurations given all the queries. In this case we will also define a worst case and an average case metric, thus we get the following estimation methods.

$$\begin{aligned} q_{k+1}^w &= \operatorname*{argmax}_{q \in Q \backslash Q_k} \min_{x=0,1} I(\mathcal{S}; q = x|\mathcal{Q}) \\ &= \operatorname*{argmax}_{q \in Q \backslash Q_k} \min_{x=0,1} H(\mathcal{S}|\mathcal{Q}) - H(\mathcal{S}|\mathcal{Q}, q = x) \\ &= \operatorname*{argmin}_{q \in Q \backslash Q_k} \max_{x=0,1} H(\mathcal{S}|\mathcal{Q}, q = x) \end{aligned} \tag{4}$$

$$q_{k+1} = \operatorname*{argmin}_{q \in Q \backslash Q_k} \sum_{x=0,1} H(\mathcal{S}|\mathcal{Q}, q = x) \tag{5}$$

The stopping criterion is $Q \backslash Q_k$ is empty or $I(\mathcal{S}; q|\mathcal{Q}) = 0$, or smaller than some predefined thresshold, for all $q \in Q \backslash Q_k$, meaning that proposing any new query we obtain no additional information; the set of queries $\mathcal{Q}$ and configuration $\mathcal{S}$ form a sufficient statistic.

If the number of possible configurations is small, we can apply the above methods to the query learning problem for identifying multiple objects. However, as the number of possible objects grows, the number of configuration grows exponentially (2^N), and the algorithm becomes impractical in most situations, because we have to estimate the posterior distribution over 2^N configurations in order to calculate the probability or the conditional entropy. To circumvent this problem [4] proposes a method of approximating the conditional entropy based on belief propagation. We present this method briefly in section §3.2 as we will compare our proposed method to that. In our proposed method we use the Max–Product algorithm presented in [3] to calculate the marginals of the posteriors and then we use them to approximate the reduction factors or the conditional entropy. The details of this method are presented in section §3.3

3.2 Entropy Approximation via Belief Propagation

Before introducing the Belief Propagation approximation to the conditional entropy we need to define some appropriate notation. We denote with $\mathcal{N}(m)$ the set of components that connect to query q_m and with $\mathcal{S}_{\mathcal{N}(m)}$ the combination of components that belong to the set $\mathcal{N}(m)$. We also use $\mathcal{Q}$ to denote the outcome of all previously asked queries. According to [4] the conditional entropy for a new candidate query q_m can be factored as follows:

$$H(\mathcal{S}|\mathcal{Q}, q_m) = -\sum_{q_m, \mathcal{S}_{\mathcal{N}(m)}} \Pr\big(\mathcal{S}_{\mathcal{N}(m)}, q_m|\mathcal{Q}\big) \log \Pr(q_m|\mathcal{Q}) + \sum_{q_m} \Pr(q_m|\mathcal{Q}) \log \Pr(q_m|\mathcal{Q}) + \textit{const}. \tag{6}$$

To calculate the marginal probabilities, define unnormalized belief on factor node $\tilde{Q} = (q_m, \mathcal{S}_{\mathcal{N}(m)})$ and variable node Q as:

$$\begin{aligned} \tilde{b_{\tilde{Q}}}(q_m, \mathcal{S}_{\mathcal{N}(m)}) =& P(q_m \mid \mathcal{S}_{\mathcal{N}(m)}) \prod_{j \in \mathcal{N}(m)} n_{j \to \tilde{Q}}(s_j), \\ \tilde{b_Q}(q_m) =& m_{\tilde{Q} \to Q}(q_m) \\ =& \sum_{\mathcal{S}_{\mathcal{N}(m)}} P(q_m, \mathcal{S}_{\mathcal{N}(m)}) \prod_{j \in \mathcal{N}(m)} n_{j \to \tilde{Q}}(s_j) \\ =& \sum_{\mathcal{S}_{\mathcal{N}(m)}} P(q_m \mid \mathcal{S}_{\mathcal{N}(m)}) P(\mathcal{S}_{\mathcal{N}(m)}) \prod_{j \in \mathcal{N}(m)} n_{j \to \tilde{Q}}(s_j), \end{aligned}$$

where $n_{j \to \tilde{Q}}(s_j)$ is the BP message passed from node j (at value s_j) to factor node $\tilde{Q}$.

Once the unnormalized beliefs are given, the first term in the expression of conditional entropy (6) can be approximated as:

$$-\sum_{q_m, \mathcal{S}_{\mathcal{N}(m)}} P(\mathcal{S}_{\mathcal{N}(m)}, q_m \mid \mathcal{Q}) \log P(q_m \mid \mathcal{S}_{\mathcal{N}(m)}) = -\frac{1}{\sigma} \sum_{q_m, \mathcal{S}_{\mathcal{N}(m)}} \tilde{b_{q_m}}(q_m, \mathcal{S}_{\mathcal{N}(m)}) \log \tilde{b_{q_m}}(q_m, \mathcal{S}_{\mathcal{N}(m)}),$$

where $\sigma = \sum_{q_m, \mathcal{S}_{\mathcal{N}(m)}} \tilde{b_{q_m}}(q_m, \mathcal{S}_{\mathcal{N}(m)}$ is the normalization factor. The second term in conditional entropy expression is the negative conditional entropy $-H(q_m \mid \mathcal{Q})$. Using belief for the node Q

we approximate this entropy as:

$$\sum_{q_m} P(q_m \mid \mathcal{Q}) \log P(q_m \mid \mathcal{Q}) = \frac{1}{\tilde{\sigma}} \sum_{q_m} \tilde{b_Q}(q_m) \log \tilde{b_Q} + \log \tilde{\sigma},$$

where $\tilde{\sigma} = \sum_{q_m} \tilde{b_Q}(q_m)$, is the normalization factor.

Now, for the approximation to work, we need to efficiently calculate the beliefs through the belief propagation algorithm described in [4].

3.3 Approximation via Max-Product Algorithm

To approximate the performance metrics presented in section §3.1 we use the marginal posterior MMLP probabilities from the Max–Product algorithm as described in [3]. The MMLP probability for a component S_n is defined as

$$\mathrm{MMLP}_n^x = \max_{S_{\mathcal{N}(n)}} \Pr\big(S_n = x, S_{\mathcal{N}(n)} | \mathcal{Q}\big)\,, \tag{7}$$

where $S_{\mathcal{N}(n)}$ is the set of all components excluding S_n and $\mathcal{Q}$ is the combination of the query outcomes.

The approach presented here is similar to the belief propagation approach presented in §3.2 in the following way. The beliefs that propagated between queries nodes and component nodes in the algorithm are:

- $Pd'_{mn}s$, the MMLP associated with each state of component S_n given the information obtained from the queries except q_m
- $Pu'_{mn}s$, the conditional probabilities of the most likely combination of all components except S_n, whose value is x, for a certain query q_m

The expressions of those beliefs are:

$$\begin{aligned}
\mathrm{Pd}_{mn}^x =& p_n^x \prod_{Q'_m \in \mathcal{M}(n)\setminus m} \mathrm{Pu}_{m'n}^x \\
\mathrm{Pu}_{mn}^x =& \max_{\mathcal{N}(m)\setminus n} [\Pr(Q_m \mid S_n = x, S_{\mathcal{N}(m)\setminus n}) \Pr(S_{\mathcal{N}(m)\setminus n})].
\end{aligned}$$

In each iteration of max-product algorithm, the beliefs Pd's propagate downward from the components to the queries to update Pu's, and the beliefs Pu's propagate upward from the queries to the components to update Pd's. By observing that, we see that Pd's actually act as the same role as $n_{j\to\tilde{Q}}$'s mentioned above in §3.2. Therefore, we use Pd's as an approximation of $n_{j\to\tilde{Q}}$'s.

As a final step the desired MMLP's are calculated from the Pu's as follows

$$\mathrm{MMLP}_n^x = p_n^x \prod_{q_m \in \mathcal{M}(n)} \mathrm{Pu}_{mn}^x.$$

Now having an expression for the MMLP's we approximate the posterior distribution using the calculated marginals. The approximation can be expressed as

$$\Pr(\mathcal{S}|\mathcal{Q}) \approx \prod_{n=1}^{N} \max_{S_{\mathcal{N}(n)}} \Pr\left(S_n, S_{\mathcal{N}(n)}|\mathcal{Q}\right). \tag{8}$$

Using the expression of (8) as the joint posterior density it is easy to find efficiently approximate expressions for the metrics presented in section §3.1.

4 Theoretical Performance Analysis

Since it is difficult to calculate the joint conditional entropy $H\left(S_1, S_2, \ldots, S_n|\mathcal{Q}\right)$ of the components given the alarms, we propose to use the MMLP values from the Max–Product algorithm to compute an approximate upper bound to the joint conditional entropy

$$H_{\mathsf{MMLP}}\left(S_1, S_2, \ldots, S_N|\mathcal{Q}\right) \triangleq -\sum_{n=1}^{N}\sum_{s_n} \max_{S_{\mathcal{N}(n)}} p\left(s_n, S_{\mathcal{N}(n)}|\mathcal{Q}\right) \log\left(\max_{S_{\mathcal{N}(n)}} p\left(s_n, S_{\mathcal{N}(n)}|\mathcal{Q}\right)\right)$$

Lemma 1. $H_{\mathsf{MMLP}}\left(S_1, S_2, \ldots, S_N|\mathcal{Q}\right) \geq H\left(S_1, S_2, \ldots, S_N|\mathcal{Q}\right)$

Proof. Let $\mathcal{S} = \{0,1\}^N$ be the set of all possible component configurations, and let $\mathcal{Q}$ be the queries that have been asked. First, we define a new probability distribution on the sample space of component configurations

$$q(s_1, s_2, \ldots, s_N|\mathcal{Q}) \triangleq \frac{1}{c}\prod_{n=1}^{N} \max_{S_{\mathcal{N}(n)}} p\left(s_n, S_{\mathcal{N}(n)}|\mathcal{Q}\right),$$

where

$$c \triangleq \sum_{s_1,s_2,\ldots,s_n=0}^{1} \prod_{n=1}^{N} \max_{S_{\mathcal{N}(n)}} p\left(s_n, S_{\mathcal{N}(n)}|\mathcal{Q}\right).$$

Observe that

$$c \leq \sum_{s_1,s_2,\ldots,s_n=0}^{1} \prod_{n=1}^{N} p\left(s_n|\mathcal{Q}\right) = 1, \tag{9}$$

because the product of the marginals defines a probability distribution on $\mathcal{S}$.

Let $D(p||q)$ be the Kullback-Leibler divergence [5, p. 19] between two distributions p and q, where $s \in \mathcal{S}$, is defined by

$$D(p||q) \triangleq \sum_{s\in\mathcal{S}} p(s) \log\left(\frac{p(s)}{q(s)}\right),$$

By the divergence inequality [5, p. 28],

$$0 \leq D(p(\boldsymbol{S}|\mathcal{Q})||q(\boldsymbol{S}|\mathcal{Q})) = \sum_{s_1,s_2,\ldots,s_N} p\left(s_1,s_2,\ldots,s_N|\mathcal{Q}\right)\log p\left(s_1,s_2,\ldots,s_N|\mathcal{Q}\right) - \sum_{s_1,s_2,\ldots,s_N} p\left(s_1,s_2,\ldots,s_N|\mathcal{Q}\right) q(s_1,s_2,\ldots,s_N|\mathcal{Q}),$$

where $p\left(\boldsymbol{S}|\mathcal{Q}\right)$ is the true conditional distribution of the component states given the alarms, and $\boldsymbol{S}$ is the random vector of component states. Rearranging the above inequality, we have

$$\begin{aligned}
H\left(S_1,S_2,\ldots,S_N\right) \leq & -\sum_{s_1,s_2,\ldots,s_N} p\left(s_1,s_2,\ldots,s_N|\mathcal{Q}\right)\log\left(\prod_{n=1}^{N}\max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\right) \\
& + \sum_{s_1,s_2,\ldots,s_N} p\left(s_1,s_2,\ldots,s_N|\mathcal{Q}\right)\log c \\
= & -\sum_{n=1}^{N}\sum_{s_n} p\left(s_n|\mathcal{Q}\right)\log\left(\max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\right) + \log c \\
& \text{Because } \max_{S_{\mathcal{N}(i)}} p\left(s_i,S_{\mathcal{N}(i)}|\mathcal{Q}\right) \leq p\left(s_i|\mathcal{Q}\right) \text{ for } i=1,2,\ldots,N \\
\leq & -\sum_{n=1}^{N}\sum_{s_n} \max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\log\left(\max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\right) + \log c \\
& \text{and by (9),} \\
\leq & -\sum_{n=1}^{N}\sum_{s_n} \max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\log\left(\max_{S_{\mathcal{N}(n)}} p\left(s_n,S_{\mathcal{N}(n)}|\mathcal{Q}\right)\right) \\
= & H_{\mathsf{MMLP}}\left(S_1,S_2,\ldots,S_n|\mathcal{Q}\right).
\end{aligned}$$

□

5 Simulations and Experimental Results

Here we present results from our entropy approximation method using the Max–Product algorithm. We performed experiments on small random networks and also on the WISER chemicals database. The performance is measured in terms of average error as a function of the number of queries that are asked. In the first set of experiments we compare the four different metrics that we proposed for query selection; namely worst case entropy, average case entropy, worst case probability and average case probability. As we see in Figure 1 the worst case entropy is the most effective metric with the smallest average error.

In the second set of experiments we compares the performance of our entropy approximation based on the Max–Product algorithm to the approximation based on belief propagation proposed in [4]. As we see from Figure 2 both methods performs equaly well in most situations.

Finally, we tested our method on real from the WISER chemicals database. For this experiment we used the most effective worst case entropy metric and the results compared to random query

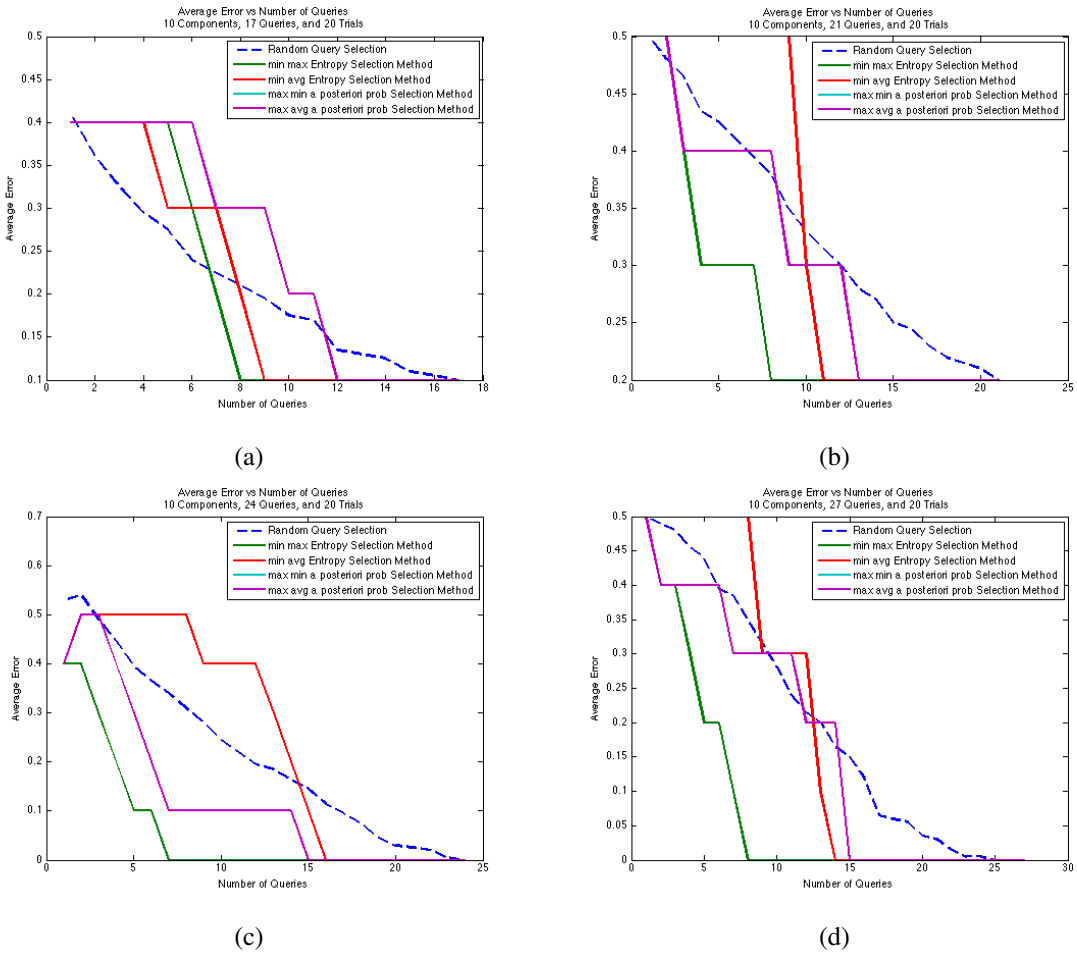

Figure 1: Comparison of algorithm performance using each of the proposed approximate performance measures. The approximate values were calculated from the marginal probabilities produced with the Max–Product algorithm.

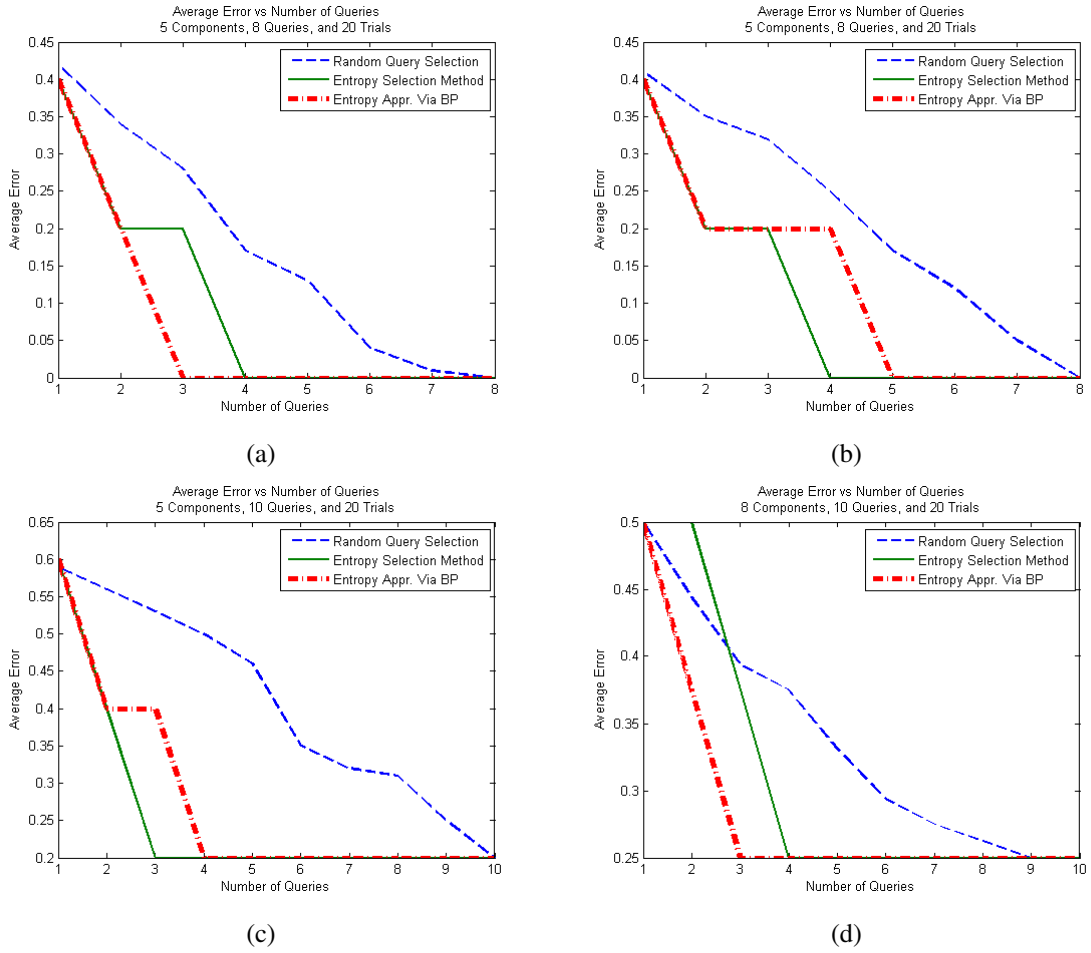

Figure 2: Comparison of performance when entropy is calculated with BP or Max–Product

selection are shown in FIgure 3. Overall, we can say that our proposed method is effective since it consistently performs better than random query selection.

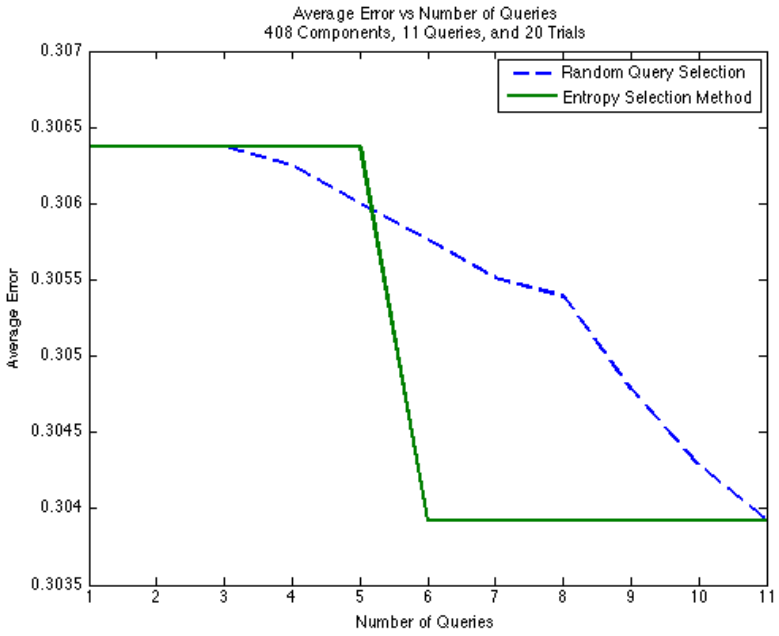

Figure 3: Performance of Max–Product entropy based query selection compared to random query selection on data from the WISER chemicals database.

6 Conclusion and Future Work

In this project we presented an effective method for active query selection for the problem of multiple object identification. As we have seen from the results presented, our method performs consistently well under different settings. The main drawback of this method is the computation time when we deal with fully connected Bayesian networks and also the accuracy of the posterior approximation using the marginals. Another issue was the poor performance in networks with many loops.

As part of future work we will try to improve on the efficiency of the Max–Product algorithm or find a more efficient method for approximating the posteriors. Also we will investigate further the accuracy of the approximation of the posterior density and try to find a more accurate approximation. Finally we will try to modify the algorithm to handle loopy networks more efficiently.

References

[1] Gowtham Bellala, Suresh K. Bhanvani, and Clayton D. Scott. Generalized shannon-fano coding for group-based query learning. *preprint*, 2009.

[2] Tung Le and C. N. Hadjicostis. Max-product algorithms for the generalized multiple-fault diagnosis problem. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 37(6):1607–1621, December 2008.

[3] Tung Le and C. N. Hadjicostis. Low-complexity max-product algorithms for problems of multiple fault diagnosis. *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 470–475, December 2007.

[4] Alice X. Zheng, Irina Rish, and Alina Beygelzimer. Efficient test selection in active diagnosis via entropy approximation. *Proceedings of UAI-2005 (Uncertainty in AI)*, March 2005.

[5] T. M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, Hoboken, NJ, 2006.

Description of Individual Effort

- **Dan**: Code for Max–Product algorithms. Equal part of progress report. Section 4 in final report. Simulations for final report.
- **Joyce**: Code for query selection and information metrics. Equal part of progress report. Part of methods section §3 in final report. Simulations for final report.
- **Antonis**: Proposal. Equal part of progress report. Most part of final report. Editing of all report documents.
- **Josh**: Code for belief propagation algorithm. Equal part of progress report. Part of methods section §3 in final report. Simulations for final report.