A Learning-Based Algorithm Selection Meta-Reasoner for the Real-Time MPE Problem

Haipeng Guo[1] and William H. Hsu[2]

[1] Department of Computer Science,
Hong Kong University of Science and Technology
hpguo@cs.ust.hk
[2] Department of Computing and Information Sciences,
Kansas State University
bhsu@cis.ksu.edu

Abstract. The algorithm selection problem aims to select the best algorithm for an input problem instance according to some characteristics of the instance. This paper presents a learning-based inductive approach to build a predictive algorithm selection system from empirical algorithm performance data of the Most Probable Explanation(MPE) problem. The learned model can serve as an algorithm selection meta-reasoner for the real-time MPE problem. Experimental results show that the learned algorithm selection models can help integrate multiple MPE algorithms to gain a better overall performance of reasoning.

1 Introduction

Uncertain reasoning under bounded resources is crucial for real-time AI applications. Examples of these include online diagnosis, crisis monitoring, real-time decision support systems, etc. In these tasks the correctness of a computation depends not only on its accuracy but also on its timeliness. Some mission-critical applications require a hard computation deadline to be strictly enforced where the utility drops to zero instantly if the answer to the query is not returned and a control is not produced. Other soft real-time domains only admit a soft deadline where the utility degrades gradually after the deadline is passed.

Researchers have broadly developed two types of methods to address real-time inference in Bayesian Networks(BNs). The first is to use anytime algorithms (Zilberstein [19]), or flexible computation (Horvitz et al. [7]). These are iterative refinement algorithms that can be interrupted at "any" time and still produce results of some guaranteed quality. Most stochastic simulation and partial evaluation inference algorithms belong to this category. The second method is to combine multiple different inference algorithms where each of these may be more or less appropriate for different characteristics of the problems. The architecture unifying various algorithms often contains a key meta-reasoning component which partitions resources between meta-reasoning and reasoning in

G.I. Webb and Xinghuo Yu (Eds.): AI 2004, LNAI 3339, pp. 307–318, 2004.

order to minimize the overall runtime of problem solving and gain a better overall performance. Work in this category include intelligent reformulation (Breese and Horvitz [2]), algorithm portfolio (Gomes and Selman [5]), cooperative inference (Santos et al. [17]), etc.

This paper is concerned with a specific type of meta-reasoning, namely *algorithm selection*, for the real-time MPE problem with a soft deadline. We use a learning-based approach to induce an MPE algorithm selection model from the training data. The learning needs to be done only once and it takes only a few minutes. Then the learned model is available to anyone as an MPE algorithm selection meta-reasoner. For an input MPE instance, the meta-reasoner(decision trees) can select the best algorithm in only a few seconds and achieve the best overall performance of reasoning. In the following sections we shall first introduce the MPE problem and the algorithm selection problem. Then we shall describe the proposed approach and present the main experimental results. Finally we shall draw the conclusions and discuss future directions.

2 Algorithm Selection for the MPE Problem

2.1 Bayesian Networks and the MPE Problem

A Bayesian network (Pearl [13]) is a pair $(\mathbf{G}, \mathbf{P})$ where $\mathbf{G}$ is a directed acyclic graph whose nodes represent random variables, and $\mathbf{P}$ is a set of Conditional Probability Tables(CPTs) — one for each node in $\mathbf{G}$. An evidence $\mathbf{E}$ is a set of instantiated nodes. An explanation is a complete assignment of all node values consistent with $\mathbf{E}$. The probability of each explanation can be computed in linear time using the chain rule:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \pi(X_i)), \tag{1}$$

where $\pi(X_i)$ denotes the parents of node X_i.

MPE is an explanation such that no other explanation has higher probability. It provides the most likely state of the world given the observed evidence. MPE has a number of applications in diagnosis, prediction, and explanation. It has been shown that both exact and approximate MPE are NP-hard (Shimony [15]; Abdelbar and Hedetniemi [1]). Exact MPE algorithms all share a worst-case complexity exponential in the maximal clique size of the underlying undirected graph. So approximate algorithms are necessary for large and complex networks. There are two basic classes of approximate MPE algorithms: stochastic sampling and search-based algorithms. Most of them are anytime algorithms. However, each algorithm may work well on some but poorly on other MPE instances. Under real-time constraints, it would be very helpful if we could know in advance which algorithm is the best for what instances.

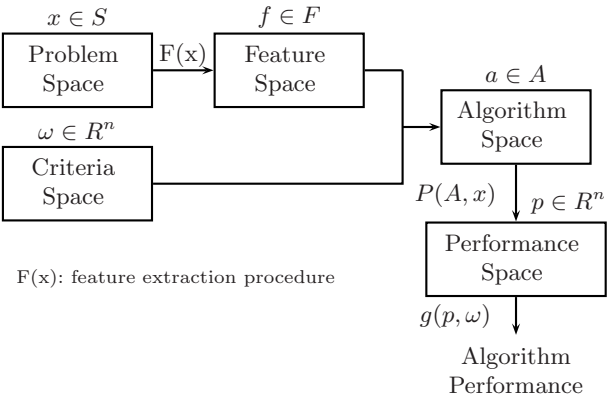

Fig. 1. The abstract model of algorithm selection

2.2 The Algorithm Selection Problem

The algorithm selection problem is to select one among a candidate set of algorithms that solves the input instance the best in some sense. It was first formulated in (Rice [14]). An abstract model of the algorithm selection problem is shown in Fig. 1, where the input instance x in the problem space S is represented as a feature vector f in the feature space. The task is to build a selection mapping between S and the algorithm space A that provides a good (measured by w) algorithm to solve x subject to the constrains that the performance of the algorithm is optimized.

From the point of view of computability theory, the general problem of algorithm selection asks to design a program, or a Turing machine, that takes as inputs the descriptions of two candidate algorithms, and outputs the best one according to some performance criteria such as problem-solving time and solution quality. By applying Rice's theorem it can be shown that the general algorithm selection problem is undecidable (Guo [6]). It implies that, in general, there can be no hope of finding a pure analytical means of automatic algorithm selection only from the descriptions of these algorithms. This general result should not be surprising because the HALTING PROBLEM basically states that in general you can not even tell whether a Turing machine (algorithm) can halt or not given arbitrary input.

3 A Machine Learning-Based Approach for Algorithm Selection

In this paper we turn to a more feasible direction: applying inductive, rather than analytical approach. Our proposed inductive approach relies significantly on experimental methods and machine learning techniques. We are partly motivated by the observation that some easy-to-compute problem features can be used as good indicators of some algorithm's performance on the specific class

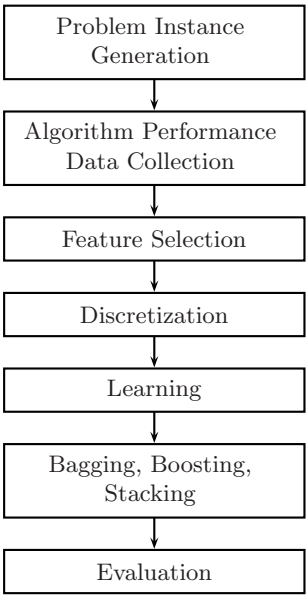

Fig. 2. The machine learning-based approach for algorithm selection

of instances. This knowledge can help select the best algorithm to gain more efficient overall computation. In NP-hard problem-solving, researchers have long noticed that algorithms exploiting special problem instance features can perform on the particular class of instances better than the worst-case scenario. In light of this, two of the main directions of this work are to study different instance features in terms of their goodness as a predictive measure for some algorithm's performance and to investigate the relationships between instance characteristics and algorithms' performance.

Another motivation comes from the inspiration of automating and mimicking human expert's algorithm selection process. In many real world situations, algorithm selection is done by hand by some experts who have a good theoretical understanding to the algorithms and are also very familiar with their runtime behaviors. The automation of the expert's algorithm selection process thus has two aspects: analytical and experimental. We have already known that the first aspect is hard to be automated and compiled into a program. In contrast, automating the experimental aspect is more feasible because of the advancements that have been made in experimental algorithmic (Johnson [11]), machine learning (Witten and Frank [18]), and uncertain reasoning techniques (Horvitz et al. [8]).

The difficulty of automatic algorithm selection is largely due to the uncertainty in the input problem space, the lack of understanding to the working mechanism of the algorithm space, and the uncertain factors of implementations and runtime environments. From the viewpoint of expert systems and machine learning, the algorithm selection system acts as an "intelligent meta-reasoner"

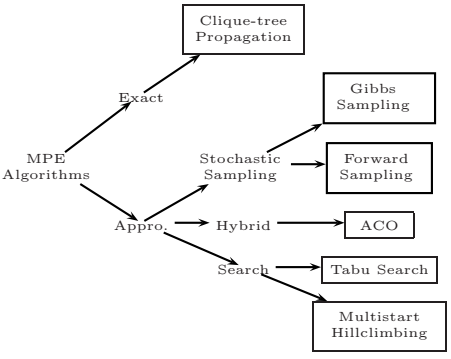

Fig. 3. Candidate MPE algorithms

that is able to learn the uncertain knowledge of algorithm selection from its past experiences and use the learned knowledge (models) to reason on algorithm selection for the input instance in order to make the right decision. This can be formulated as the following machine learning problem:

The algorithm selection learning problem
Task T: selecting the best algorithm, $Best\_Algm(f)$, for instance f.
Measure P: percent of correct selections.
Training data E: algorithm performance data collected from experiments.

Since the target function, $Best\_Algm(f)$, has discrete values, this is indeed a classification problem. An overview of the procedure is shown in Fig. 2. The first two steps, including instance generation and algorithm performance data collection, prepare the training data. The next two steps preprocess the data, including discretization and feature selection. Then in the learning step, machine learning algorithms are applied to induce the predictive algorithm selection model. Also, some meta-learning methods — such as bagging, boosting, and stacking (Witten and Frank [18]) — can be used here to improve the learned predictive models. Finally, the best learned model is evaluated on test data.

4 The Algorithm Space And The Feature Space

4.1 The Algorithm Space

Our candidate MPE algorithms include one exact algorithm: Clique-Tree Propagation(CTP) (Lauritzen and Spiegelhalter [12]); two sampling algorithms: Gibbs Sampling (Pearl [13]) and Forward Sampling (also called Likelihood Weighting) (Fung and Chang [3]) ; two local search-based algorithms: Multistart Hillclimbing and Tabu Search (Glover and Laguna [4]); and one hybrid algorithm combining both sampling and search: Ant Colony Optimization(ACO). These algorithms are chosen because currently they are among the most commonly used MPE algorithms. A classification of these representative algorithms is shown in Fig. 3.

Because of the lack of space, we refer interested readers to (Guo [6]) for detailed descriptions.

4.2 The Instance Feature Space

An MPE instance consists of three components: the network structure, the CPTs, and the evidence.

Network characteristics include network *topological type* and *connectedness*. We distinguish three topological types: polytrees, two-level networks(Noisy-OR), and multiply connected networks. Network connectedness *conn* is simply calculated as $conn = \frac{n\_arcs}{n\_nodes}$. These two characteristics have a direct influence on the exact inference algorithm's performance. In contrast, sampling algorithms' performance is rarely affected by them.

CPT characteristics include *CPT size* and *CPT skewness*. Since we only consider binary nodes, the maximum number of parents of a node, $max\_parents$, can be used to bound the CPT size. The skewness of the CPTs is computed as follows (Jitnah and Nicholson [10]): for a vector (a column of the CPT table), $v = (v_1, v_2, \ldots, v_m)$, of conditional probabilities,

$$skew(v) = \frac{\sum_{i=1}^{m} |\frac{1}{m} - v_i|}{1 - \frac{1}{m} + \sum_{i=2}^{m} \frac{1}{m}}. \quad (2)$$

where the denominator scales the skewness from 0 to 1. The skewness of a CPT is the average of the skewness of all columns, whereas the skewness of the network is the average of the skewness of all CPTs. We will see that CPT skewness has a significant influence on the relative performance of sampling and search-based algorithms.

Evidence characteristics includes the *proportion* and the *distribution type* of evidence nodes. Evidence proportion is simply the number of evidence nodes, n_evid, divided by n_nodes: $\frac{n\_evid}{n\_nodes}$. Usually, more evidence nodes implies more unlikely evidence. Hence, the MPE will also be quite unlikely and the probability that it is hit with any sampling scheme is not very high. The distribution of evidence nodes also affects the hardness of MPE instances. If most evidence nodes are "cause" nodes, the problem is called *predictive reasoning*. If most evidence nodes are "effect" nodes, it is called *diagnostic reasoning*. It has been proven that predictive reasoning is easier than diagnostic reasoning (Shimony and Domshlak [16]). In our experiments, we will consider three types of evidence distributions: strictly predictive, strictly diagnostic, and randomly distributed. An inference problem is called "strictly predictive" if the evidence nodes have no non-evidence parents; it is called "strictly diagnostic" if the evidence nodes have no non-evidence children.

We are aware that there might exist some other features that could work as well or even better. These particular features are chosen mainly because domain knowledge, previous literature(Jitnah and Nicholson [10]; Ide and Cozman [9]; Shimony and Domshlak [16]), and our initial experimental experience all

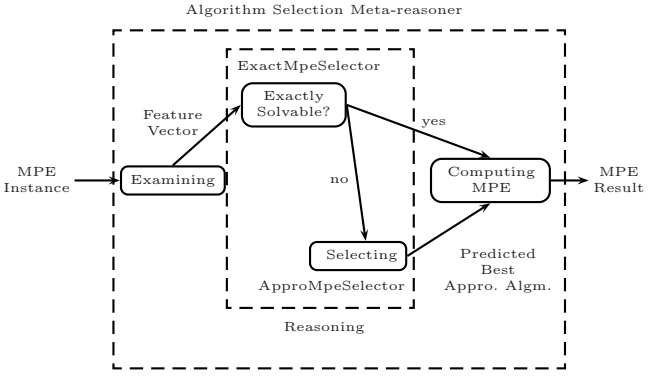

Fig. 4. The algorithm selection meta-reasoner

suggest that they are good indicators of MPE instance hardness and algorithm performance. The other reason is that they are all easy to compute.

5 Experiments and Results

Our first goal is to identify the class of MPE instances for which the exact inference algorithm is applicable. When the exact algorithm is not applicable (most probably due to an out-of-memory error in practice), we need to look at various approximate algorithms. Thus our second goal is to learn the predictive model that can determine which approximate algorithm is the best. Therefore, the algorithm selection meta-reasoner to be learned will consist of two classifiers as shown in Fig. 4: the *ExactMpeSelector* for exact algorithm selection, and the *ApproMpeSelector* for approximate algorithm selection.

5.1 Data Preparation

In the data preparation phase, we first generate MPE instances with different characteristics uniformly at random. The random generation of MPE instances with controlled parameter values is based on a Markov chain method (Ide and Cozman [9]). It is reasonable and necessary to consider only a subset of all possible MPE instances , i.e. the set of "Real World Problems" (RWP). In order to simulate RWP BNs, we first extract the ranges of all characteristic parameter values from a collection of 13 real world samples, call it D_{RWBN}, and then generate networks and MPE instances based on the extracted distributions. The ranges of their characteristic values are as follows: $30 \leq n\_nodes \leq 1,000$; $conn \in [1.0, 2.0]$; $maxParents < 10$; $0.25 < skewness 0.87$. These characteristics information are used to guide the generation of our training datasets.

The first training dataset for learning *ExactMpeSelector*, D_{MPE1}, is generated as follows: we first randomly generate networks with connectedness varying from 1.0 to 2.0(with a step of 0.2) and maximum number of parents varying from 3 to 10. The number of nodes used are $\{30, 50, 80, 100, 120, 150, 200\}$. We then

Table 1. Experiment 1: learning *ExactMpeSelector*

	C45	NaiveB.	BN	Bagg.	Boost.	Stack.
c.a. (%)	94.80	82.79	90.06	94.75	94.81	94.56
s.d. (%)	0.27	0.36	0.24	0.25	0.23	0.45

run exact algorithm CTP on these randomly generated networks and record the performance. To perform inference, CTP first compiles the network into a clique tree. We record the maximum clique size and label the network as "yes" instance if the compilation is successful. Otherwise, if it throws out an out-of-memory error or takes longer than 5 minutes, we label the instance as "no". D_{MPE1} has four numeric attributes: *n_node*, *topology*, *connectedness*, and *maxParents*. The target class, *ifUseExactAlgorithm*, takes boolean values representing whether exact algorithm is applicable or not. We also include these 13 real world networks into D_{MPE1}. The final D_{MPE1} contains a total of 1,893 instances.

The second training dataset for learning *ApproMpeSelector*, D_{MPE2}, only contains two-level and multiply networks. We generate a set of networks with different characteristic values and then run all 5 approximate algorithms on them with different evidence settings. We give each algorithm a fixed number of samples or search points and label the instance using the best algorithm that returns the best MPE value. The total number of samples was 300, 1000, or 3,000. D_{MPE2} has 8 attributes: *n_node*, *topology*, *connectedness*, *maxParents*, *skewness*, *evidPercent*, *evidDistri*, and *n_samples*. The target class is the best algorithm for this instance. D_{MPE2} contains 5,184 instances generated from 192 networks.

5.2 Model Induction

We now apply various machine learning algorithms to induce the predictive algorithm selection models. We consider three different kinds of models: decision tree learning (C4.5), naive Bayes classifier, and Bayesian network learning (K2). We also consider three meta-learning methods to combine multiple models: bagging, boosting and stacking, which all use C4.5 as their base learner. So, total, we have six different learning schemes. Before learning, we also conduct data preprocessing such as discretization and/or feature selection if necessary.

In experiment 1, we run all 6 learning schemes on D_{MPE1}. Table 1 shows the classification accuracies of each learned model. We use the best model out of these 6, i.e. the one that has both high classification accuracy and efficient reasoning mechanism. We can see that boosting(94.81%), C4.5(94.80%), and bagging(94.75%) all have a high classification accuracy. We also notice that NaiveBayes has the worst performance of only 82.79%, which verifies that the features in D_{MPE1} are not independent of each other. Since C4.5 is much simpler and more efficient on reasoning, we use the decision tree learned by C4.5 as the best model for exact MPE algorithm selection: *ExactMpeSelector*.

In experiment 2, we look at feature selection for approximate MPE algorithm selection using D_{MPE2}. Each data case has 9 attributes. The first 8 are MPE

Table 2. Experiment 3: learning *ApproMpeSelector*

	C4.5	NaiveB.	BN	Bagg.	Boost.	Stack.
c.a (%)	77.75	72.77	76.08	75.44	77.16	77.36
s.d. (%)	0.23	0.03	0.01	0.27	0.26	0.32

instance features and the last one is the target class labelling the best approximate MPE algorithm. We apply a GA-wrapped C4.5 feature selection classifier to search for the best feature subset. The wrapper uses C4.5 as the evaluation classifier and a simple genetic algorithm to search the attribute space. The GA's population size and number of generations are 20. The crossover probability is 0.6 and the mutation probability 0.033. The feature subset selected is {*n_node*, *skewness*, *evidPercent*, *evidDistri*, *n_samples*}. Note that all network structure features are filtered out. The result agrees with our domain knowledge that network structure does not affect approximate algorithms' performance very much. From now on, we will use this selected subset rather than D_{MPE2} itself.

In experiment 3, we apply all 6 machine learning algorithms on the selected feature subset of D_{MPE2} to induce *ApproMpeSelector*. The experimental results(Table 2) show that C4.5 has the highest classification accuracy(77.75%). Because of this and the fact that C4.5 has a much faster reasoning mechanism, we choose it as the best model for *ApproMpeSelector*.

In experiment 4, we study the influences of each individual feature on the relative performance of different algorithms. We partition the training dataset used in experiment 3 by each feature's values and record the number of times of each algorithm being the best at each feature value level. The results are summarized as follows: (1)*Number of Nodes*. *n_nodes* affects the relative performance of two search algorithms, but forward sampling and ACO are almost not affected. When *n_nodes* increases from 50 to 100 multi-start hillclimbing becomes the best algorithm more frequently and the chances for tabu search being the best drops significantly. This can be explained by the constant size of the tabu list used. When network becomes larger while the tabu list remain the same size, the tabu list's influence becomes weaker. This makes it lose its best algorithm position to multi-start hillclimbing. (2)*Number of Samples*. Again, the relative performances of two search algorithms are affected, but forward sampling and ACO's are not. When the given number of samples increases from 300 to 1,000 to 3,000, tabu search becomes the best algorithm more often and multi-start hillclimbing loses its top rank. It seems that Tabu search can utilize available number of search points better than multi-start hillclimbing. (3) *CPT skewness*. Skewness has the most significant influence on the relative performance of these algorithms as shown in Table 3. When the skewness is low, the search space is flat and search algorithms perform much better than sampling algorithms. Multi-start hillclimbing wins the best algorithm two times more than tabu search. When the skewness is around 0.5, ACO outperforms all other algorithms almost all the time. When the skewness increases to 0.9, forward sampling and ACO are the winners and perform equally well. We also notice that forward sampling works better only for highly skewed networks and ACO works for both highly-skewed networks

Table 3. Partitioning D_{MPE2} by CPT Skewness

skewness	Number of Times of Being Best Algorithm				
	gibbs.	forward.	multiHC	tabu	aco
0.1	0	0	1059	512	157
0.5	0	4	9	174	1677
0.9	0	858	9	28	942

and medium-skewed networks. (4)*Evidence Proportion.* The result shows that changing evidence percentage does not affect two search algorithms' relative performance, but it affects forward sampling and ACO. ACO is out-performed by forward sampling as the percentage of evidence nodes increases from 10% to 30%. We should also note that evidence percentage's influence is much weaker than that of skewness. (5)*Evidence Distribution.* The relative performance of multi-start hillclimbing is not affected. Tabu search is only slightly affected. It shows that diagnostic inference is relatively hard for forward sampling but is easy for ACO. In contrast, random distributed evidence is relatively hard for ACO but is easy for forward sampling.

5.3 Model Evaluation

Finally, we evaluate the learned algorithm selection meta-reasoner to verify that it does achieve a better overall performance of reasoning. For a given MPE instance, the meta-reasoner first examines the instance and extracts the feature vector. Then *ExactMpeSelector* is called to determine whether it is exactly solvable. If the classification result is "yes", the system then executes the exact inference algorithm. If it is "no", *ApproMpeSelector* will be used to select the best approximate algorithm. The selected algorithm is then executed and the final MPE value returned. This procedure is shown in Fig. 4.

We first test *ExactMpeSelector* on $D_{MpeTest}$, which contains 405 instances generated in the same way as previous experiments. *ExactMpeSelector* identifies 243 "yes" instances correctly. Then we apply *ApproMpeSelector* on the rest 162 "no" instances. The result shows that there are 123 correctly classified instances and 39 incorrectly classified instances. The classification accuracy is 75.93%.

To show that the algorithm selection system outperforms any single algorithm, we partition these 162 "no" instances into three groups according to their skewness. There are 27 unskewed, 54 medium-skewed, and 81 highly-skewed instances. For each group, we compare the total approximate MPE values returned by each individual algorithm with the total values returned by the algorithm selection system. On medium-skewed and highly-skewed instances, the algorithm selection system returns the largest total MPE values of 6.0×10^{-6} and 0.16. On unskewed instances, the system returns the second largest MPE value of 2.1×10^{-26}. But the largest total, computed by multi-start hill climbing, is only 2.2×10^{-26}. The algorithm selection system's result is almost as good as that. Adding them all together, the algorithm selection system returns the largest total MPE value on all 162 instances.

We also test the system on real BNs. First, all 13 real world networks are correctly classified by ExactMpeSelector. There are 11 "yes" networks. On these networks, all predicted best approximate algorithms also agree with actual best algorithms. The two "no" networks are *link* and *munin1*. *ApproMpeSelector* selects ACO as the best approximate algorithm for both. The actual running of all algorithms on *link* returns all 0, given 5,000 samples. This is due to its huge state space(724 nodes, 5.77×10^{277} states) and low skewness(13,715 out of 20,502 numbers are 0). *munin1* has 189 nodes and 3.23×10^{123} states. Given 5,000 samples, ACO returns the best MPE of 5.93×10^{-8}. Forward sampling finds the second best MPE of 6.61×10^{-9}. All other algorithms just return 0.

In summary, the test results on both artificial and real Bayesian networks verify that the learned algorithm selection meta-reasoner can make reasonable decisions on selecting exact and best approximate MPE algorithms for the input MPE instance and provides a better overall performance.

6 Conclusions and Discussions

We have reported a machine learning-based approach to build an algorithm selection meta-reasoner for the real-time MPE problem. The system consists of two predictive models (classifiers). For an input MPE instance, the first one decides if exact algorithm is applicable. And the second one determines which approximate algorithm is the best. Different MPE instance characteristics have different properties and affect different algorithms' performance. Our experimental results show that CPT skewness is the most important feature for approximate MPE algorithm selection. It reveals that in general search-based algorithms work better on unskewed networks and sampling algorithms work better on skewed networks. Other features, such as *n_nodes*, *n_samples*, *evidPercent* and *evidDistri*, all affect these algorithms' relative performance to some degree, although not as strong as *skewness* does. The learned algorithm selection system uses some polynomial time computable instance characteristics to select the best algorithm for the NP-hard MPE problem and gains the best overall performance in terms of the returned solution quality given the same computational resources. The time of computing features and selecting the best algorithm are negligible comparing to the actual problem solving time. The most important and difficult task in this scheme is to identify the set of candidate features. The main limitation of this method is that the size of training data grows exponentially in the number of features used. The fact that training data are generated from a specific set of real world instances may also limit the learned system's applicable range. In the future this scheme could be applied to algorithm selection of other NP-hard problems and help to build more efficient real-time computation systems.

Acknowledgements

Thanks anonymous reviewers for their valuable comments. This work was partially supported by the HK Research Grants Council under grant HKUST6088/01E.

References

Abdelbar, A. M., Hedetniemi, S. M.: Approximating MAPs for belief networks in NP-hard and other theorems. Artificial Intelligence. **102** (1998) 21–38

Breese, J. S., Horvitz, E.: Ideal reformulation of belief networks. In UAI90. (1990) 129–144

Fung, R., Chang, K. C.: Weighting and integrating evidence for stochastic simulation in Bayesian networks. In UAI89. (1989) 209–219

Glover, F., Laguna, M.: Tabu search. Kluwer Academic Publishers, Boston. (1997)

Gomes, C. P., Selman, B.: Algorithm portfolio design: theory vs. practice. In UAI97. (1997) 190–197

Guo, H.: Algorithm selection for sorting and probabilistic inference: a machine learning-based approach. PhD thesis, Kansas State University. (2003)

Horvitz, E.: Computation and action under bounded resources. PhD thesis, Stanford University. (1990)

Horvitz, E., Ruan, Y., Kautz, H., Selman, B., Chickering, D. M.: A Bayesian approach to tackling hard computational problems. In UAI01. (2001) 235–244

Ide, J. S., Cozman F. G.: Random generation of Bayesian networks. In Brazlian Symposium on Artificial Intelligence, Pernambuco Brazil. (2002)

Jitnah, N., Nicholson, A. E.,: Belief network algorithms: A study of performance based on domain characterization. In Learning and Reasoning with Complex Representations. **1359** Springer-Verlag (1998) 169–188

Johnson, D.: A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser and D. S. Johnson and C. C. McGeoch, editors, Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges. (2002) 215–250

Lauritzen, S. L., Spiegelhalter, D. J.: Local computations with probabilities on graphical structures and their application to expert systems (with discussion). J. Royal Statist. Soc. Series B **50** (1988) 157-224

Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA, Morgan-Kaufmann. (1988)

Rice, J. R.: The algorithm selection problem. In M. V. Zelkowitz, editors, Advances in computers. **15** (1976) 65–118

Shimony, S. E.: Finding MAPs for belief networks is NP-hard. Artificial Intelligence. **68** (1994) 399–410

Shimony, S. E., Domshlak, C.: Complexity of probabilistic reasoning in directed-path singly connected Bayes networks. Artificial Intelligence. **151** (2003) 213–225

Santos, E., Shimony, S. E., Williams, E.: On a distributed anytime architecture for probabilistic reasoning. Technique Report AFIT/EN/TR94-06. Department of Electrical and Computer Engineering, Air Force Institute of Technology. (1995)

Witten, I. H., Frank, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann. (1999)

Zilberstein, S.: Operational rationality through compilation of anytime algorithms. PhD Thesis. University of California at Berkeley. (1993)