Dynamic Weighting A* Search-based MAP Algorithm for Bayesian Networks

Xiaoxun Sun
Computer Science Department
University of Southern California
Los Angeles, CA 90089
xiaoxuns@usc.edu

Marek J. Druzdzel
Decision Systems Laboratory
School of Information Sciences
& Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
marek@sis.pitt.edu

Changhe Yuan
Department of Computer Science
and Engineering
Mississippi State University
Mississippi State, MS 39762
cyuan@cse.msstate.edu

Abstract

In this paper we propose the Dynamic Weighting A* (DWA*) search algorithm for solving MAP problems in Bayesian networks. By exploiting asymmetries in the distribution of MAP variables, the algorithm is able to greatly reduce the search space and offer excellent performance both in terms of accuracy and efficiency.

1 Introduction

The Maximum a *Posteriori* assignment (MAP) is the problem of finding the most probable instantiation of a set of variables given partial evidence on the remaining variables in a Bayesian network. One specialization of MAP that has received much attention is the Most Probable Explanation (MPE) problem. MPE is the problem of finding the most probable assignment of a set of variables given full evidence on the remaining variables. MAP turns out to be a much more difficult problem compared to MPE and computing the probability of evidence. Particularly, the decision problem for MPE is NP-complete while the corresponding MAP problem is NP^{PP}-complete [Park, 2002]. MAP is more useful than MPE for providing explanations. For instance, in diagnosis, we are generally only interested in the configuration of fault variables given some observations. There may be many other variables that have not been observed that are outside the scope of our interest.

In this paper, we introduce the Dynamic Weighting A* (DWA*) search algorithm for solving MAP that is generally more efficient than existing algorithms. The algorithm explores asymmetries among all possible assignments in the joint probability distribution of the MAP variables. Typically, a small fraction of the assignments can be expected to cover a large portion of the total probability space with the remaining assignments having practically negligible probability [Druzdzel, 1994]. Also, the DWA* uses dynamic weighting based on *greedy guess* [Park and Darwiche, 2001; Yuan *et al.*, 2004] as the heuristic function. Although it is theoretically not admissible (admissible heuristic should offer an upper bound on the MAP), the heuristic significantly reduces the size of the search tree, while rarely pruning away the optimal solutions.

2 MAP

The MAP problem is defined as follows: Let **M** be the set of MAP variables, the configuration of which is what we are interested in; **E** is the set of evidence, namely the variables whose states we have observed; The remainder of the variables, denoted by **S**, are variables whose states we neither know nor care about. Given an assignment **e** of variables **E**, the MAP problem is that of finding the assignment **m** of variables **M** which maximizes the probability $P(m \mid e)$, while the MPE problem is the special case of MAP with **S** being empty, i.e.,

$$MAP = \max_{M} \sum_{S} p(M, S \mid E) \ . \tag{1}$$

In general, in Bayesian networks, we use the Conditional Probability Table (CPT) ϕ as the *potential* over a variable and its parent nodes. The notation ϕ_e stands for the potential in which we have fixed the value of $e \in E$. Then the probability of MAP with Φ as its CPTs turns out to be a real number:

$$MAP = \max_{M} \sum_{S} \prod_{\phi \in \Phi} \phi_e \ . \tag{2}$$

In Equation 2, summation commutes with summation, and maximization commutes with maximization. However, summation does not commute with maximization and vice versa. Therefore, it is obligatory to do summation before any maximization. The order is called the *elimination order*. The size of the largest clique minus 1 in a jointree constructed based on an elimination order is called the *induced width*. The induced width of the best elimination order is called the *treewidth*. However, for the MAP problems in which the set **S** and the **M** are both non-empty, the order is constrained. Then the constrained elimination order is known as the *constrained treewidth*. Generally, the constrained treewidth is much larger than treewidth, leading the problem beyond the limit of feasibility for complex models. Specifically, for some MAP problems, variable elimination on polytrees is subject to the constrained treewidth, which requires exponential time, while MPE problems can be computed in linear time [Park and Darwiche, 2003].

A very efficient approximate search-based algorithm based on local search, proposed by Park and Darwiche [2001], is capable of solving MAP efficiently. An exact method, based on branch-and-bound depth-first search, proposed by Park

and Darwiche [2003], performs quite well when the search space is not too large. Another approximate algorithm proposed more recently by Yuan et al. [2004] is a Reheated Annealing MAP algorithm. It is somewhat slower on simple networks but it is capable of handling difficult cases that exact methods cannot tackle.

3 Solving MAP using Dynamic Weighting A*

We propose in this section an algorithm for solving MAP using Dynamic Weighting A* search, which incorporates the *dynamic weighting* [Pohl, 1973] in the heuristic function, *relevance reasoning* [Druzdzel and Suermondt, 1994] and *dynamic ordering* in the search tree.

3.1 A* search

MAP can be solved by A* search in the probability tree that is composed of all the variables in the MAP set. The nodes in the search tree represent partial assignments of the MAP variables **M**. The root node represents an empty assignment. The MAP variables will be instantiated in a certain order. If a variable **x** in the set of MAP variables **M** is instantiated at the i*th* place using its j*th* state, it will be denoted as $\mathbf{M}_{ij}$. Leaves of the search tree correspond to the last MAP variable that has been instantiated. The vector of instantiated states of each MAP variable is called an *assignment* or a *scenario*.

We compute the probability of assignments while searching the whole probability tree using chain rule. For each inner node, the newly instantiated node will be added to the evidence set, i.e., the evidence set will be extended to $\mathbf{M}_{ij} \cup \mathbf{E}$. Then the probability of the MAP problem which consists of **n** MAP variables can be presented as follows:

$$\begin{aligned} P(\mathrm{M} \mid E) &= P(M_{ni} \mid M_{1j}, M_{2k}, \ldots M_{(n-1)t}, E) \\ &\quad \ldots P(M_{2k} \mid M_{1j}, E) P(M_{1j} \mid E) \ . \end{aligned}$$

Suppose that we are in the x*th* layer of the search tree and preparing for instantiating the x*th* MAP variables. Then the function above can be rewritten as follows:

$$P(\mathrm{M} \mid E) = \overbrace{P(M_{ni} \mid M_{1j} \ldots M_{(n-1)t}, E) \ldots P(M_{(x+1)z} \mid M_{xy} \ldots E)}^{b} \cdot \underbrace{P(M_{xy} \mid M_{1j}, M_{2k} \ldots M_{(x-1)q}, E) \ldots P(M_{1j} \mid E)}_{a} . \quad (3)$$

The general idea of the Dynamic Weighting A* search is that during the search, in each inner node of the probability tree, we can compute the *exact* value of item (a) in the function above. We can estimate the heuristic value of the item (b) for the MAP variables that have not been instantiated given the initial evidence set and the MAP variables that have been instantiated as the new evidence. In order to fit the typical format of the cost function of A* Search, we can take the logarithm of the equation above, which will not change its monotonicity. Then we get $f(n) = g(n) + h(n)$, where $g(n)$ and $h(n)$ are obtained from the logarithmic transformation of items (a) and (b) respectively. $g(n)$ gives the exact cost from the start node to node in the n*th* layer of the search tree, and h(n) is the estimated cost of the best search path from the n*th* layer to the leaf nodes of the search tree. In order to guarantee the optimality of the solution, $h(n)$ should be *admissible*, which in this case means that it should be an upper-bound on the value of any assignment with the currently instantiated MAP variables as its elements.

3.2 Heuristic Function with Dynamic Weighting

The A* Search is known for its completeness and optimality. For each search step, we only expand the node in the frontier with the largest value of $f(n)$.

Definition 1 *A heuristic function h_2 is said to be* more informed than *h_1 if both are admissible and h_2 is closer to the optimal cost. For the MAP problem, the probability of the optimal assignment $P_{opt} < h_2 < h_1$.*

Theorem 1 *If h_2 is more informed than h_1 then A_2^** dominates *A_1^* (Nilsson). [Pearl, 1985]*

The *power* of the heuristic function is measured by the amount of pruning induced by $h(n)$ and depends on the accuracy of this estimate. If $h(n)$ estimates the completion cost precisely ($h(n) = P_{opt}$), then A* will only expand nodes on the optimal path. On the other hand, if no heuristic at all is used (for the MAP problem this amounts to $h(n) = 1$), then a uniform-cost search ensues, which is far less efficient. So it is critical for us to find an $admissble$ and $tight$ $h(n)$ to get both accurate and efficient solutions.

Greedy Guess

If each variable in the MAP set **M** is conditionally independent of all the rest of MAP variables (this is called *exhaustive independence*), then the MAP problem amounts to a simple computation based on the *greedy* chain rule. We instantiate the MAP variable in the current search layer to the state with the largest probability and repeat this for each of the remaining MAP variables one by one. The probability of MAP is then

$$P(M|E) = \prod_{i=1}^{n} \max_j P(M_{ij} | M_{(i-1)k} \ldots M_{1m}, E) \ . \quad (4)$$

The requirement of exhaustive independence is too strict for most of the MAP problems to be calculated by using the function above. Simulation results show that in practice, when this requirement is violated, the product is still extremely close to the MAP probability [Yuan *et al.*, 2004]. This suggests that it can be potentially used as a heuristic function for MAP.

The curve *Greedy Guess Estimate* in Figure 1 shows that with the increase of the number of MAP variables, the ratio between the greedy guess and the accurate estimate of the optimal probability diverges from the ideal ratio *one* although not always monotonically.

Dynamic Weighting

Since the greedy guess is a tight lower bound on the optimal probability of MAP, it is possible to compensate for the error between the greedy guess and the optimal probability. We can achieve this by adding a weight to the greedy guess such that their product is equal or larger than the optimal probability

for each inner node in the search tree under the following assumption:

$$\exists\epsilon\{\forall P_{GreedyGuess} * (1+\epsilon) \geq P_{opt} \wedge$$
$$\forall\epsilon'(P_{GreedyGuess} * (1+\epsilon^{'}) \geq P_{opt}) \Rightarrow \epsilon \leq \epsilon^{'}\},$$

where ϵ is the minimum weight that can guarantee the heuristic function to be admissible. Figure 1 shows that if we just keep ϵ constant, neglecting the changes of the estimate accuracy with the increase of the MAP variables, the estimate function and the optimal probability can be represented by the curve *Constant Weighting Heuristic*. Obviously, the problem with this idea is that it is less informed when the search progresses, as there are fewer MAP variables to estimate.

Dynamic Weighting [Pohl, 1973] is an efficient tool for improving the efficiency of A* Search. If applied properly, it will keep the heuristic function admissible while remaining tight on the optimal probability. For MAP, in shallow layers of the search tree we get more MAP variables than in deeper layers. Hence, the greedy estimate will be more likely to diverge from the optimal probability. We propose the following Dynamic Weighting Heuristic Function for the x*th* layer of the Search tree of n MAP variables:

$$h(x) = GreedyGuess \cdot (1 + \alpha\frac{n-(x+1)}{n}), (\alpha \geq \epsilon)\ . \quad (5)$$

Rather than keeping the weight constant throughout the search, we dynamically change it, so as to make it less heavy as the search goes deeper. In the last step of the search $(x = n-1)$, the weight will be zero, since the Greedy Guess for only one MAP variable is exact and then the cost function f(n-1) is equal to the probability of the assignment. Figure 1 shows an empirical comparison of greedy guess, constant, and dynamic weighting heuristics against accurate estimates of the probability. We see that the dynamic weighting heuristic is more informed than constant weighting.

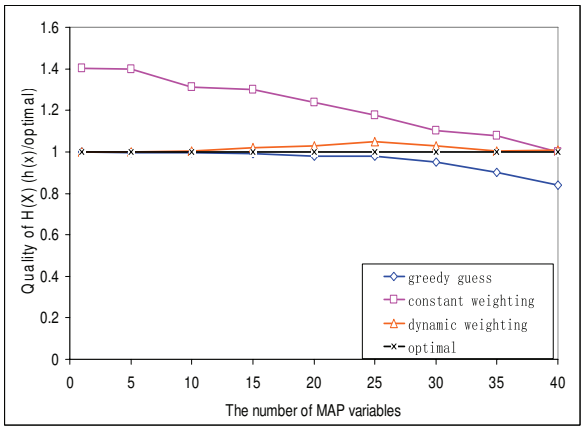

Figure 1: Constant Weighting Heuristic and Dynamic Weighting Heuristic based on Greedy Guess.

3.3 Searching with Inadmissible Heuristics

Since the minimum weight ϵ that can guarantee the heuristic function to be admissible is unknown before the MAP problem is solved, and it may vary in different cases, we normally set α to be a safe parameter which is supposed to be larger than ϵ (In our experiments, we set α to be 1.0). However, what if α is accidentally smaller then ϵ leading the weighted heuristic to be inadmissible? Suppose there are two candidate assignments: s_1 and s_2 with probability p_1 and p_2 respectively, among which s_2 is the optimal assignment that the algorithm fails to find. And s_1 is now in the last step of search, which will lead to a suboptimal solution. We skip the logarithm in the function for the sake of clarity here (then the cost function f is a product of transformed g and h instead of their sum).

$$f_1 = g_1 \cdot h_1 \text{ and } f_2 = g_2 \cdot h_2.$$

The error introduced by an inadmissible h_2 is $f_1 > f_2$. The algorithm will then find s_1 instead of s_2, i.e.,

$$f_1 > f_2 \Rightarrow g_1 \cdot h_1 > g_2 \cdot h_2.$$

Since s_1 is now in the last step of search, $f_1 = p_1$ (Section 3.2). Now suppose that we have an *ideal* heuristic function $h_2^{'}$, which leads to $p_2 = g_2 \cdot h_2^{'}$. Then we have:

$$\frac{g_1 \cdot h_1}{p_2} > \frac{g_2 \cdot h_2}{g_2 \cdot h_2^{'}} \Rightarrow \frac{p_1}{p_2} > \frac{g_2 \cdot h_2}{g_2 \cdot h_2^{'}} \Rightarrow \frac{p_1}{p_2} > \frac{h_2}{h_2^{'}}.$$

It is clear that only when the ratio between the probability of suboptimal assignment and the optimal one is larger than the ratio between the inadmissible heuristic function and the ideal one, may the algorithm find a suboptimal solution.

Because of large asymmetries among probabilities that are further amplified by their multiplicative combination [Druzdzel, 1994], we can expect that for most of the cases, the ratios between p_1 and p_2 are far less than **1**. Even though the heuristic function will sometimes break the rule of admissibility, if only the greedy guess is not too divergent from the ideal estimate, the algorithm will still not diverge from the optimal probability. Our simulation results also confirm the robustness of the algorithm in finding optimal solutions.

3.4 Improvements to the Algorithm

There are several techniques that can improve the efficiency of the basic A* algorithm.

Relevance Reasoning

The main problem faced by our approach is the complexity of probabilistic inference. The critical factor in exact inference schemes for Bayesian networks is the topology of the underlying graph and, more specifically, its connectivity. Relevance reasoning [Druzdzel and Suermondt, 1994] is a technique based on *d*-separation and other simple and computational efficient techniques for pruning irrelevant parts of a Bayesian network and can yield sub-networks that are smaller and less densely connected than the original network. For MAP, our focus is the set of variables **M** and the evidence set **E**. Parts of the model that are probabilistically independent from the nodes in **M** given the observed evidence **E** are computationally irrelevant to reasoning about the MAP problem.

Dynamic Ordering

As the search tree is constructed dynamically, we have the freedom to order the variables in a way that will improve the efficiency of the DWA* search. Expanding nodes with

the largest asymmetries in their marginal probability distributions lead to early cut-off of less promising branches of the search tree. We use the entropy of the marginal probability distributions as a measure of asymmetry.

4 Experimental Results

To test DWA*, we compared its performance on MAP problems in real Bayesian networks against those of current state of the art MAP algorithms: the P-LOC and P-SYS algorithms [Park and Darwiche, 2001; 2003] in SamIam, and ANNEALEDMAP [Yuan *et al.*, 2004] in SMILE. We implemented DWA* in C++ and performed our tests on a 3.0 GHz Pentium D Windows XP computer with 2GB RAM. We used the default parameters and settings for all the three algorithms above during comparison, unless otherwise stated.

4.1 Experimental Design

The Bayesian networks that we used in our experiments included Alarm [Beinlich *et al.*, 1989], Barley [Kristensen and Rasmussen, 2002], CPCS179 and CPCS360 [Pradhan *et al.*, 1994], Hailfinder [Abramson *et al.*, 1996], Munin, Diabetes [Andreassen *et al.*, 1991], Andes [Conati *et al.*, 1997], Pathfinder, and Win95pts [Heckerman *et al.*, 1995], some of which were constructed for diagnosis. We also tested the algorithms on two very large proprietary diagnostic networks built at the HRL Laboratories (HRL1 and HRL2). The statistics for all networks are summarized in Table 1. We divide the networks into three groups: (1) small and middle-sized, (2) large but tractable, and (3) hard networks.

Group	Network	#Nodes	#Arcs
1	Alarm	37	46
	CPCS179	179	239
	CPCS360	360	729
	Hailfinder	56	66
	Pathfinder	135	195
	Andes	223	338
	Win95pts	76	112
2	Munin	1,041	1,397
	HRL1	1,999	3,112
	HRL2	1,528	2,492
3	Barley	48	84
	Diabetes	413	602

Table 1: Statistics for the Bayesian networks that we use.

For each network, we randomly generated 20 cases. For each case, we randomly chose 20 MAP variables from the root nodes or all of them if there were fewer than 20 root nodes. We chose the same number of evidence nodes from the leaf nodes. To set evidence, we sampled from the prior probability distribution of a Bayesian network in its topological order and cast the states of the sample to the evidence nodes. Following previous tests of MAP algorithms, we set the search time limit to be $3,000$ seconds (50 minutes).

4.2 Results for the First and Second Group

In the first experiment, we ran the P-LOC, P-SYS, ANNEALEDMAP and DWA* on all the networks in the first and second group, and all four algorithms generated results within the time limit. The P-SYS is an exact algorithm. So Table 3 only reports the number of MAP problems that were solved optimally by the P-LOC, ANNEALEDMAP and DWA*. The DWA* found all optimal solutions. The P-LOC missed only one case on Andes and the ANNEALEDMAP missed one on Hailfinder and two cases on Andes.

	P-LOC	A-MAP	DWA*
Alarm	20	20	20
CPCS179	20	20	20
CPCS360	20	20	20
Hailfinder	20	19	20
Pathfinder	20	20	20
Andes	19	18	20
Win95pts	20	20	20
Munin	20	20	20
HRL1	20	20	20
HRL2	20	20	20

Table 2: The number of cases that were solved optimally out of 20 random cases for the first and second groups of networks.

Since both ANNEALEDMAP and P-LOC failed to find all optimal solutions in Andes, we studied the performance of the four algorithms as a function of the number of MAP variables (we randomly generated 20 cases for each number of MAP variables).

#MAP	P-SYS	P-LOC	A-MAP
10	0	0	0
20	0	1	2
30	0	1	0
40	TimeOut	4	4
50	TimeOut	6	2
60	TimeOut	5	2
70	TimeOut	6	5
80	TimeOut	6	1

Table 3: The number of cases for which the existing algorithms found smaller probabilities than A* Search in network Andes.

Because the search time of P-SYS increased quickly with the number of MAP variables, and it failed to generate any result when the number of MAP variables reached 40, while DWA* found all largest probabilities, we compared all the other three algorithms against DWA*. With the increase of the number of MAP variables, both P-LOC and ANNEALEDMAP turned out to be less accurate than DWA* on Andes. When the number of MAP variables was above 40, there were about 25% cases of P-LOC and 15% cases in which ANNEALEDMAP found smaller probabilities than DWA*. We notice from Table 5 that P-LOC spent less time than DWA* when using its default settings for Andes, so we increased the search steps of P-LOC such that it spent the same amount of time as DWA* in order to make a fair comparison. However, in practice the search time is not continuous in the number of search steps, so we just chose parameters

for P-LOC such that it spent slightly more time than DWA*. Table 4 shows the comparison results. We can see that after increasing the search steps of P-LOC, DWA* still maintains better accuracy.

#MAP	P-LOC<DWA*	P-LOC>DWA*
10	0	0
20	0	0
30	0	0
40	1	0
50	2	0
60	2	1
70	3	2
80	5	0

Table 4: The number of cases that the P-LOC found larger/smaller probabilities than DWA* in network Andes when spending slightly more time than DWA*.

In addition to the precision of the results, we also compared the efficiency of the algorithms. Table 5 reports the average running time of the four algorithms on the first and the second groups of networks. For the first group, the ANNEALEDMAP, P-LOC and P-SYS algorithms showed similar efficiency on all except the CPCS360 and Andes networks. DWA* generated solutions within the shortest time on the average. Its smaller variance of the search time indicates that DWA* is more stable across different networks.

	P-SYS	P-LOC	A-MAP	A*
Alarm	0.017	0.020	0.042	0.005
CPCS179	0.031	0.117	0.257	0.024
CPCS360	0.045	75.20	0.427	0.072
Hailfinder	2.281	0.109	0.219	0.266
Pathfinder	0.052	0.056	0.098	0.005
Andes	14.49	1.250	4.283	2.406
Win95pts	0.035	0.041	0.328	0.032
Munin	3.064	4.101	19.24	1.763
HRL1	0.493	51.18	2.831	0.193
HRL2	0.092	3.011	2.041	0.169

Table 5: Average running time in seconds of the P-SYS, P-LOC, ANNEALEDMAP and DWA* algorithms on the first and second group of networks.

For the second group, which consists of large Bayesian networks, P-SYS, ANNEALEDMAP and DWA* were all efficient. DWA* search still spent shortest time on the average, while the P-LOC was much slower on the HRL1 network.

4.3 Results for the Third Group

The third group consisted of two complex Bayesian networks: Barley and Diabetes, many nodes of which have more than 10 different states. Because the P-SYS algorithm did not produce results within the time limit, the only available measure of accuracy was a relative one: which of the algorithms found an assignment with a higher probability. Table 6 lists the number of cases that were solved differently between the P-LOC, ANNEALEDMAP, and DWA* algorithms. P_L, P_A and P_* stand for the probability of MAP solutions found by P-LOC, ANNEALEDMAP and DWA* respectively.

	$P_*>P_L/P_*<P_L$	$P_*>P_A/P_*<P_A$
Barley	3/2	5/3
Diabetes	5/0	4/0

Table 6: The number of cases that are solved differently from P-LOC, ANNEALEDMAP and DWA*.

For Barley, the accuracy of the three algorithms was quite similar. However, for Diabetes DWA* was more accurate: it found solutions with largest probabilities for all 20 cases, while P-LOC failed to find 5 and ANNEALEDMAP failed to find 4 of them.

	P-SYS	P-LOC	A-MAP	A*
Barley	TimeOut	68.63	31.95	122.1
Diabetes	TimeOut	338.4	163.4	81.8

Table 7: Average running time in seconds of the P-SYS, P-LOC, ANNEALEDMAP and DWA* on the third groups.

DWA* turned out to be slower than P-LOC and ANNEALEDMAP on Barley but more efficient on Diabetes (see Table 7).

4.4 Results for Incremental MAP Test

Our last experiment focused on the robustness of the four algorithms to the number of MAP variables. In this experiment, we set the number of evidence variables to be 100 and generated MAP problems with an increasing number of MAP nodes and ran four algorithms on these cases. We chose the Munin network, because it seemed the hardest network among the group 1 & 2 and had sufficiently large sets of root and leaf nodes. The running time for each of the cases are shown in Figure 2. Typically, P-SYS and P-LOC need more running time in face of more complex problems, while ANNEALEDMAP and DWA* seem more robust in comparison.

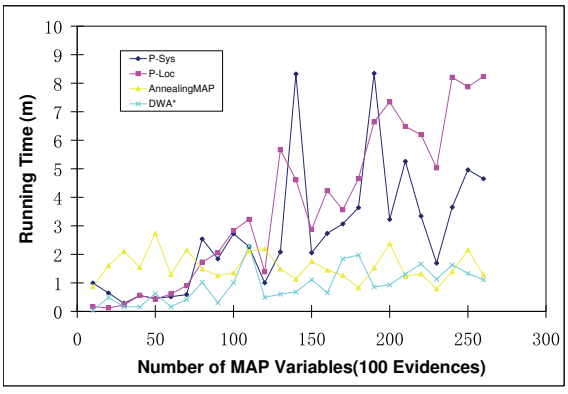

Figure 2: Plot of the running time of the P-SYS, P-LOC, ANNEALEDMAP and DWA* algorithms when increasing the number of MAP nodes on the Munin network.

5 Discussion

Finding MAP in Bayesian networks is hard. By exploiting asymmetries among the probabilities of possible assignments of Joint Probability Distributions of MAP variables, the DWA* is able to greatly reduce the search space and lead to efficient and accurate solution of the MAP problem. Our experiments show that generally, DWA* Search is more efficient than the existent algorithms. Especially for large and complex Bayesian networks, when the exact algorithm fails to generate any result within a reasonable time, DWA* can still provide accurate solutions efficiently. Further extension of this research is to apply the DWA* to the k-MAP problem, which is to find k most probable assignments for MAP variables. It is very convenient for the DWA* algorithm to achieve that, since after finding the most probable assignment, the algorithm keeps all the candidate assignments in the search frontier. We can expect that the additional search time will be sublinear in k. Solving the k-MAP problem gives additional insurance against missing the optimal solutions, as there is a very good chance that if it is missed at first, it will show up among the following k-1 solutions.

Acknowledgements

This research was supported by the Air Force Office of Scientific Research grants F49620–03–1–0187 and FA9550–06–1–0243 and by Intel Research. We thank Leon Rothkrantz for his support during the research. We thank Adnan Darwiche and Keith Cascio for providing us with the latest version of the P-SYS and P-LOC algorithms. All experimental data have been obtained using SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`.

References

[Abramson *et al.*, 1996] B. Abramson, J. Brown, W. Edwards, A. Murphy, and R. Winkler. Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–72, 1996.

[Andreassen *et al.*, 1991] S. Andreassen, R. Hovorka, J. Benn, K. G. Olesen, and E. R. Carson. A model-based approach to insulin adjustment. In M. Stefanelli, A. Hasman, M. Fieschi, and J. Talmon, editors, *Proceedings of the Third Conference on Artificial Intelligence in Medicine*, pages 239–248. Springer-Verlag, 1991.

[Beinlich *et al.*, 1989] I. Beinlich, G. Suermondt, R. Chavez, and G. Cooper. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In *In Proc. 2nd European Conf. on AI and Medicine*, pages 38:247–256, Springer-Verlag, Berlin, 1989.

[Conati *et al.*, 1997] C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel. On-line student modeling for coached problem solving using Bayesian networks. In *Proceedings of the Sixth International Conference on User Modeling (UM–96)*, pages 231–242, Vienna, New York, 1997. Springer Verlag.

[Druzdzel and Suermondt, 1994] Marek J. Druzdzel and Henri J. Suermondt. Relevance in probabilistic models: "Backyards" in a "small world". In *Working notes of the AAAI–1994 Fall Symposium Series: Relevance*, pages 60–63, New Orleans, LA, 4–6 November 1994.

[Druzdzel, 1994] M. J. Druzdzel. Some properties of joint probability distributions. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 187–194, Morgan Kaufmann Publishers San Francisco, California, 1994.

[Heckerman *et al.*, 1995] D. Heckerman, J. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, 38:49–57, 1995.

[Kristensen and Rasmussen, 2002] K. Kristensen and I.A. Rasmussen. The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33:197–217, 2002.

[Park and Darwiche, 2001] J. D. Park and A. Darwiche. Approximating MAP using local search. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI–01)*, pages 403–410, Morgan Kaufmann Publishers San Francisco, California, 2001.

[Park and Darwiche, 2003] J. D. Park and A. Darwiche. Solving MAP exactly using systematic search. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI–03)*, pages 459–468, Morgan Kaufmann Publishers San Francisco, California, 2003.

[Park, 2002] J. D. Park. MAP complexity results and approximation methods. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI–02)*, pages 388–396, Morgan Kaufmann Publishers San Francisco, California, 2002.

[Pearl, 1985] J. Pearl. *Heuristics : intelligent search strategies for computer problem solving*. Addison-Wesley Publishing Company, Inc., 1985.

[Pohl, 1973] I. Pohl. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. In *Proc. of the 3rd IJCAI*, pages 12–17, Stanford, MA, 1973.

[Pradhan *et al.*, 1994] M. Pradhan, G. Provan, B. Middleton, and M. Henrion. Knowledge engineering for large belief networks. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–94)*, pages 484–490, San Mateo, CA, 1994. Morgan Kaufmann Publishers, Inc.

[Yuan *et al.*, 2004] C. Yuan, T. Lu, and M. J. Druzdzel. Annealed MAP. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI–04)*, pages 628–635, AUAI Press, Arlington, Virginia, 2004.