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

Xiaoxun Sun*& Marek J. Druzdzel & Changhe Yuan
Decision Systems Laboratory
School of Information Sciences & Intelligent Systems Program
University of Pittsburgh, Pittsburgh, PA 15260
{sxun,marek,cyuan}@sis.pitt.edu

Abstract

Maximum a *Posteriori* assignment (MAP) is the most probable instantiation of a set of variables given a partial evidence on the remaining variables in a Bayesian network. Finding MAP has been proved to be an NP-hard problem [15], and it is not only exponential in the network treewidth, but also in the *constrained treewidth* [10]. Exact approaches often fail to yield any results for MAP problems in very large Bayesian networks, and even approximate approaches may not yield efficient solutions.

We introduce the Dynamic Weighting A^* (DWA^*) search algorithm for solving MAP. By exploiting asymmetries in the distribution of MAP variables, the algorithm is able to greatly reduce the search space yielding very good quality MAP solutions. Experimental results demonstrate that our algorithm finds solutions generally faster and with a lower variance in search time than existing algorithms.

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 networks. One specialization of the MAP that has been paid 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 of the remaining variables. MAP turns out to be a very difficult problem even when compared to MPE or computing the probability of evidence. Particularly, the decision problem for MPE is NP-complete while the corresponding MAP problem is NP^{PP}-complete [10]. MAP is more useful than MPE for providing explanations. For instance, in diagnosis, generally we are only interested in the configuration of fault variables given some observations. There may be many other variables that have not been observed and are outside the scope of our interest.

Several researchers have proposed algorithms for solving the MAP problem. A very efficient approximate search-based algorithm based on local search, proposed by Park and Darwiche [10], is capable of solving MAP efficiently which is based on local search. An exact method, based on branch-and-bound depth-first search, proposed by Park and Darwiche [12], performs quite well when the search space is not too large. Another approximate proposed more recently by Yuan et al. [16] is a Reheated Annealing MAP algorithm. It is somewhat slower on simple networks but it is able to handle very hard cases on which the exact algorithm can not solve.

In this paper, we introduce the Dynamic Weighting A^* (DWA^*) Search algorithm for solving MAP that is faster than any of the existing algorithms. The algorithm explores the asymmetries among all possible assignments in the joint probability distributions. Typically, a small fraction of assignments can be expected to cover a large portion of the total probability space with the remaining assembles having practically negligible probability [5].

Previous research and simulation results have shown that the *greedy guess* [11, 16], which is represented as follows:

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

is quite close to the optimal solution of the MAP problems. In other words, it offers a very tight lower bound

*On leave from the Department of Media & Knowledge Engineering, Delft University of Technology, The Netherlands on the optimal solution. While it is theoretically not admissible (admissible heuristic should offer an upper bound on the MAP), with a simple extension it offers ϵ-admissibility and excellent performance.

The remainder of this paper is structured as follows. Section 2 defines the MAP problem and summarizes the main results on its complexity. It also outlines several methods for solving MAP. Section 3 describes the Dynamic Weighting A^* Search algorithm. Section 4 presents the results of applying the algorithm to several real complex Bayesian networks.

2 MAP and Previous Research

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

$$map = \max_{M} \sum_{S} p(M, S \mid E) \ . \tag{2}$$

In general, in Bayesian networks, we use the Conditional Probability Table (CPT) ϕ as the *potential* over a variable and its parent nodes. A potential over all the states of one variable after updating beliefs is called *marginal*. 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:

$$map = \max_{M} \sum_{S} \prod_{\phi \in \Phi} \phi_e \ . \tag{3}$$

We will introduce the algorithm of *Variable Elimination* [12] here in order to compute MAP. The name of the algorithm is just because it sums or maximizes out variables from a list of variables one by one, and this order is named 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 which neither the set $\mathbf{S}$ nor the $\mathbf{M}$ is empty, the order is constrained. Then the constrained variable order is known as the *constrained treewidth*. Generally, the constrained treewidth is much higher than treewidth, leading the problem beyond the limit of feasibility. Specifically, for some MAP problems, the variables elimination on polytrees is subject to the constrained treewidth, which requires exponential time, while MPE problems can be computed in linear time [12].

Consider Equation 3. When eliminating variables in MAP problems, summation commutes with summation, and maximization commutes with maximization. However, summation never commutes with maximization and vise versa. What is more strict, the order are obligatory to do summation before any maximization.

There are several proposed algorithms for solving the MAP problems in Bayesian networks. Park and Darwiche [10] proposed an approach using *local search* for solving the MAP problems. This algorithm starts from an initial guess and then iteratively improves the solution by moving to a better neighbor. More recently, Park and Darwiche [12] proposed a *branch-and-bound depth-first search algorithm* which provided the guarantee on the optimality of the obtained solution. All of these algorithms could provide very efficient solutions for most of the MAP problems when the networks are not too large or complex. Another approximate algorithm proposed by Yuan et al. [16] is a Reheated Annealing MAP algorithm. It uses Markov Chain Monte Carlo methods to sample from the target distribution, and applies the reheated simulated annealing technique to simulate a nonhomogeneous Markov chain. It is somewhat slower on simple networks but it is able to handle very hard cases that the exact algorithm can not solve.

3 Solving MAP using Dynamic Weighting A^* Search

We present in this section an algorithm for solving MAP using Dynamic Weighting A^* search, which incorporates the *dynamic weighting* [13] in the heuristic function, *relevance reasoning* [6] and *dynamic ordering* in the search tree.

The remainder of this section is organized as follows. First, we introduce the A^* Search. Second, we compare the A^* search algorithm with the branch-and-bound algorithm which is also suitable for solving the MAP problem, and show why it is superior. Third, we introduce our method of composing the heuristic function $h(n)$. Forth, we analyze the efficiency and accuracy of the dynamic weighting A^* search algorithm and the situation when over-estimate happened. Finally, we discuss two techniques for improving the efficiency of the algorithm.

3.1 A^* search

The MAP problems 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 $\mathbf{M}$. The root node represents an empty assignment. Each MAP variable will be instantiated in a certain order. If a variable $\mathbf{x}$ in the set of the set of MAP variables $\mathbf{M}$ is intantiated 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 *assignments* 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 into 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 $\mathbf{n}$ 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 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:

$P(\mathrm{M} \mid E) =$

$$\begin{aligned} &\overbrace{P(M_{ni} \mid M_{1j} \ldots M_{(n-1)t}, E) \ldots P(M_{(x+1)z} \mid M_{xy} \ldots E)}^{b} \\ &\quad \cdot \underbrace{P(M_{xy} \mid M_{1j}, M_{2k} \ldots M_{(x-1)q}, E) \ldots P(M_{1j} \mid E)}_{a} \end{aligned}$$

The general idea of the Dynamic Weighting A^* Search algorithm is that during the search, in each inner node of the probability tree, we can compute the value of item (a) in the function above *exactly*. 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 intantiated as the new evidence. In order to fit the typical format of the cost function of A^* Search, we just 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 intantiated MAP variables as its elements.

3.2 Efficiency Comparison between A^* and Branch-Bound Search for MAP Problems

A^* Search is closely related to the *branch-and-bound* techniques. For the MAP problems, the efficiency of the search algorithm is dominated by the number of nodes in the probability tree that are instantiated. In order to compare the efficiency of the two search algorithms for the MAP Problems,we will first introduce some definitions.

Definition 1 *: An algorithm A_1 is said to* dominate *an algorithm A_2 if every node expanded by A_1 is also expanded by A_2. We will also use the phrase* more efficient than *interchangeably with* dominates *[13].*

Let s_{max} be the most probable assignment and its probability be P_{max}. Let $P_{BestSoFar}$ denote the probability of the best assignment that we have found so far, which is less or equal to the P_{max}.

Theorem 1 *Given the same cost function f(x), A^* dominates branch and bound on MAP problems.*

Proof: The condition for cutting the current search path in "branch and bound" search is $f(x) < P_{BestSoFar}$. So the whole search space can be denoted by:

$$\begin{aligned} S_1 \;=\; \{x : f(x) \;\geq\; & P_{BestSoFar} \\ \cup\; x \in & \textit{ optimal assignment}\} . \end{aligned}$$

With the admissibility of the f(y), which guarantees that f(y) an upper-bound on the probability of the optimal solution, we have $f(y) \geq P_{max}$. For all nodes x that on the search path other than the optimal assignment, we have $f(x) \geq f(y)$. The whole search space can be denoted by:

$$\begin{aligned} S_2 \;=\; \{x : f(x) \;\geq\; & f(y) \\ \cup\; x \;\in\; & \textit{optimal assignment}\} \end{aligned}$$

Given that $P_{BestSoFar} \leq P_{max}$ and $f(y) \geq P_{max}$, we have $f(y) \geq P_{BestSoFar}$, which implies $S_2 \subseteq S_1$, i.e., that the A^* search dominates the branch-and-bound search. □

3.3 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 2 *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 2 *If h_2 is more informed than h_1 then A_2^** dominates *A_1^* (Nilsson). [13]*

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 for MAP.

3.3.1 Greedy Guess

If each variable in the MAP set $\mathbf{M}$ 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

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

The requirement of exhaustive independence is too strict for most of the MAP problem 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 [16]. This suggests using it as an ϵ-admissible heuristic function [13].

The curve *Greedy Guess Estimate* in Figure 1 shows that with the increase of the MAP variables, the ratio between the greedy guess and the accurate estimate of the optimal probability diverges from the ideal ratio **1.0** although not always monotonically.

3.3.2 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 do this by adding a weight to the greedy guess such that the product of them is equal or larger than the optimal probability for each inner node in the search tree. This yields an *ϵ-admissible* heuristic function that we need in order to find the optimal solutions. This assumption can be represented as follows:

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

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 the shallow layer of the search tree, we get more MAP variables than the deeper layer for estimate. 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:

$$\begin{aligned} h(x) &= P_{GreedyGuess} \cdot (1 + \alpha \frac{n-(x+1)}{n}) \\ &\quad (\alpha \geq \epsilon) \ . \end{aligned}$$

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 estimate of the probability. We see that the dynamic weighting heuristic becomes more informed than constant weighting. In our experiments, we set α to be 1.0.

3.4 Searching with Nonadmissible Heuristics for MAP Problem

Let us give a closer look and analyze the conditions when the algorithm fails to achieve optimality. 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

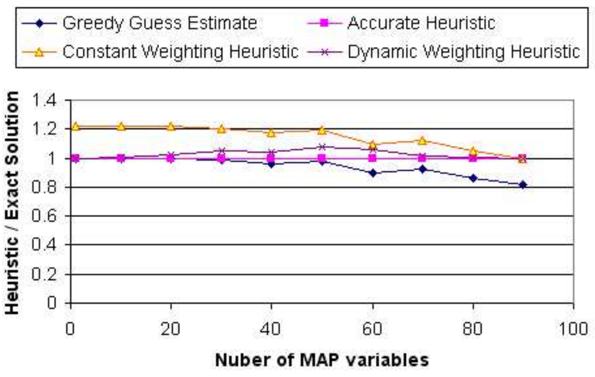

Figure 1: Constant Weighting Heuristic and Dynamic Weighting Heuristic based on Greedy Guess.

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 a non-admissible 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 3.3.2). 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 nonadmissible 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 [5], we can expect that for most of 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 proved the robustness of the algorithm in finding optimal solutions.

3.5 Improvements to the Algorithm

There are two main techniques that we used to improve the efficiency of the basic A^* algorithm.

3.5.1 Relevance Reasoning

The main problem faced by the decision-theoretic approach is the complexity of probabilistic reasoning. The critical factor in exact inference schemes for Bayesian networks is the topology of the underlying graph and, more specifically, its connectivity. The framework of relevance reasoning ([6] is an accessible summary of the relevant techniques) is based on *d*-separation and other simple and computational efficient techniques for pruning irrelevant parts of a Bayesian networks and can yield sub-networks that are smaller and less densely connected than the original network. Relevance reasoning is an integral part of the SMILE library on which the implementation of our algorithm is based.

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.

3.5.2 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 marginal probability distribution leads 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 the DWA^* algorithm, we studied its performance on many MAP problems in real Bayesian networks. We compare our results against these of current state of the art MAP algorithms: the P-Loc [11], P-Sys [12] and AnnealedMAP [16] algorithms respectively. We implemented the DWA^* algorithm in C++ and performed our tests on a 2.4 GHz Pentium IV Windows XP computer with 750MB memory.

4.1 Experimental Design

The Bayesian networks that we used include Alarm [4], Barley [9], CPCS179 and CPCS360 [14], Diabetes [2], Hailfinder [1], Munin [3], Pathfinder [7], P223layout, and Win95pts [8], some of which are 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 these networks are summarized in Table 1. We divided the networks into three groups: (1) small and middlesized, (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
	P223layout	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 are using.

For each network, we randomly generated 20 cases. For each case, we randomly chose 20 MAP variables among the root nodes or all the them if root nodes were less than 20. We chose the same number of evidence nodes from among 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 of the four algorithms generate results within the time limit. The P-Sys algorithm reported that it found all the optimal solutions. Table 3 reports the number of MAP problems that are solved correctly by the P-Loc AnnealedMAP and DWA^* algorithms. They all performed well on these networks. The DWA^* was able to find all the optimal solutions. The P-Loc algorithm missed only one case on the P223layout network and the AnnealedMAP missed one on Haifinder and two cases on P223layout.

Since both AnnealedMAP and P-Loc failed to find all the optimal solution in P223layout, in each of the 20 cases we studied the performance of the 4 algorithms as a function of the number of MAP variables (we randomly generated 20 cases for each number of MAP variables).

Because the search time of P-Sys increased very fast with the number of MAP variables, and it failed to generate any result when the number of MAP variables reached 40, while the DWA^* Search found all

	P-Loc	A-MAP	A^*
Alarm	20	20	20
CPCS179	20	20	20
CPCS360	20	20	20
Hailfinder	20	19	20
Pathfinder	20	20	20
P223layout	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 are solved correctly out of 20 random cases for the first and second group of networks.

No. of 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 that the other 3 algorithms found smaller probabilitis than A^* Search in network P223layout.

the largest probabilities, we compared all the other 3 algorithms with DWA^* Search. With the increase of the number of MAP variables, both the P-Loc and AnnealedMAP turned to be less accurate for P223layout. 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^*.

In addition to the precision of the results, we also compared the efficiency of the algorithms. Table 4 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 P223layout networks. The DWA^* search generated solutions within the shortest time on average. The small variance of the search time indicates that DWA^* is more stable across different networks.

For the second group, which consists of large Bayesian networks, P-Sys, AnnealedMAP and DWA^* are all efficient. DWA^* search still spent shortest search time on average, while the P-Loc was much slower on the HRL1 network.

	P-Sys	P-Loc	A-MAP	A^*
Alarm	0.011	0.019	0.076	0.006
CPCS179	0.030	0.134	0.250	0.019
CPCS360	0.057	90.202	0.820	0.123
Hailfinder	3.910	0.118	0.452	0.239
Pathfinder	0.054	0.061	0.050	0.001
P223layout	32.370	1.376	12.166	2.507
Win95pts	0.031	0.041	0.292	0.030
Munin	3.382	5.353	19.620	2.996
HRL1	1.287	224.968	7.157	0.418
HRL2	0.087	5.45	4.071	0.384

Table 4: Average running time in seconds of the P-Sys, P-Loc, AnnealedMAP and DWA^* algorithms on the first and second group of networks.

4.3 Results for the Third Group

The third group consists of two complex Bayesian networks: Barley and Diabetes, many nodes of which have more then 10 different states. As the P-Sys algorithm did not produce any results within the time limit, the only available measure of accuracy was a relative one: which of the algorithms found an assignment with higher probability. Table 5 lists the number of cases that were solved differently between P-Loc, AnnealedMAP, and the DWA^* algorithm and the number of cases that the DWA^* algorithm found a more probable assignment. 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 5: The numbe of cases that are solved differently from P-Loc, AnnealedMAP and DWA^*.

For Barley, the accuracy of the three algorithms is quite similar. However, for Diabetes DWA^* is 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	101.47	34.67	199.16
Diabetes	TimeOut	369.35	315.79	185.89

Table 6: Average running time in seconds of the P-Sys, P-Loc, AnnealedMAP and DWA^* algorithms on the third groups of Bayesian networks.

DWA^* turned out to be slower than P-Loc and AnnealedMAP on Barley but more efficient on Diabetes (see Table 6).

4.4 Results for Incremental Evidence Test

Out last experiment focused on the robustness of the four algorithms to the number of nodes in the MAP set and the evidence set. In this experiment, we generated MAP problems with an increasing number of MAP and evidence nodes and ran four algorithms on these cases . We chose the Munin network, as it looks as the hardest network among the group 1 & 2 . The P-Sys was able to solve only cases with fewer than 140 MAP and evidence variables. The times for each of the cases are shown in Figure 2.

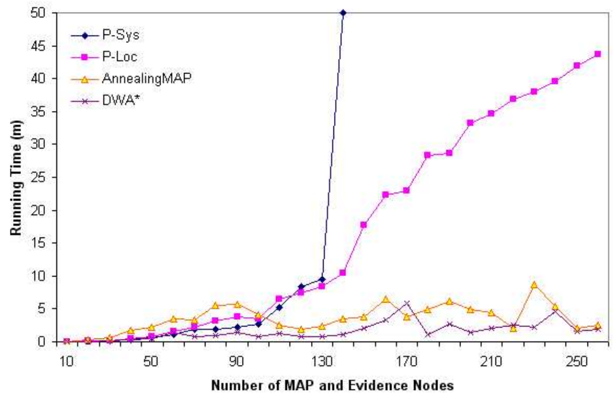

Figure 2: Plot of the running time of the P-Sys, P-Loc, AnnealedMAP and DWA^* algorithms when increasing the number of evidence nodes on the Munin network.

5 Discussion

Finding MAP in Bayesian networks is hard. By exploiting asymmetries among the probabilities of possible assignments properties of Joint Probability Distributions among all the possible assignments, the Dynamic Weighting A^* Search is able to greatly reduce the search space and lead to efficient and accurate solution of the MAP problem. Our experimental result also show that generally, the Dynamic Weighting A^* 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, the Dynamic Weighting A^* Search can still provide accurate solutions efficiently.

Further extension of this research is to apply the Dynamic Weighting A^* Search algorithm 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 linear in k.

In sum, the Dynamic Weighting A^* Search algorithm enriches the approaches for solving MAP problem and extends the scope of MAP problems that can be solved.

Acknowledgements

This research was supported by the Air Force Office of Scientific Research grant F49620–03–1–0187. We thank Adam Zagorecki and Tomek Sowinski for insightful comments that led to improvements in the paper. We thank James D. Park for answering the question regarding his systematic search algorithm and we thank Adnan Darwiche and Keith Cascio for providing us with the latest version of the P-Sys and P-Loc algorithms within the SamIam software. All experimental data have been obtained using SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://www.sis.pitt.edu/~genie`.

References

[1] 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.

[2] 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.

[3] S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. R. Sørensen, A. Rosenfalck, and F. Jensen. MUNIN — an expert EMG assistant. In John E. Desmedt, editor, *Computer-Aided Electromyography and Expert Systems*, chapter 21. Elsevier Science Publishers, Amsterdam, 1989.

[4] 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. 2'nd European Conf. on AI and Medicine*, pages 38:247–256, Springer-Verlag, Berlin, 1989.

[5] 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.

[6] 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 (An extended version of this paper is in preparation.), 4–6 November 1994.

[7] D. Heckerman. Probabilistic similarity networks. *Networks*, 20(5):607–636, August 1990.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] J. Pearl. *Heuristics : intelligent search strategies for computer problem solving.* Addison-Wesley Publishing Company, Inc., 1988.

[14] 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.

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

[16] 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.