An ACO Algorithm for the Most Probable Explanation Problem

Haipeng Guo[1], Prashanth R. Boddhireddy[2], and William H. Hsu[2]

[1] Department of Computer Science,
Hong Kong University of Science & Technology
hpguo@cs.ust.hk
[2] Department of Computing & Information Sciences,
Kansas State University
{pbo8844,bhsu}@cis.ksu.edu

Abstract. We describe an Ant Colony Optimization (ACO) algorithm, ANT-MPE, for the most probable explanation problem in Bayesian network inference. After tuning its parameters settings, we compare ANT-MPE with four other sampling and local search-based approximate algorithms: Gibbs Sampling, Forward Sampling, Multistart Hillclimbing, and Tabu Search. Experimental results on both artificial and real networks show that in general ANT-MPE outperforms all other algorithms, but on networks with unskewed distributions local search algorithms are slightly better. The result reveals the nature of ACO as a combination of both sampling and local search. It helps us to understand ACO better, and, more important, it also suggests a possible way to improve ACO.

1 Introduction

Bayesian networks (BNs) (Pearl 1988) are the currently dominant method for uncertain reasoning in AI. They encode the joint probability distribution in a compact manner by exploiting conditional independencies. One of the main purposes of building a BN is to conduct probabilistic inference - i.e. to compute answers to users' queries, given exact values of some observed evidence variables. This paper is concerned with a specific type of Bayesian network inference: finding the Most Probable Explanation (MPE). MPE is the problem of computing the instantiation of a Bayesian network that has the highest probability given the observed evidence. It is useful in many applications including diagnosis, prediction, and explanation. However, MPE is NP-hard (Littman 1999).

Ant Colony Optimization (ACO) is a recently developed approach that takes inspiration from the behavior of real ant colonies to solve NP-hard optimization problems. The ACO meta-heuristic was first introduced by Dorigo(1992), and was recently defined by Dorigo, Di Caro and Gambardella(1999). It has been successfully applied to various hard combinatorial optimization problems.

In this paper we present the first application of ACO to the MPE problem. In section 2 we briefly introduce MPE and the related work. Then we describe our ANT-MPE algorithm in section 3. In section 4 we present the experimental

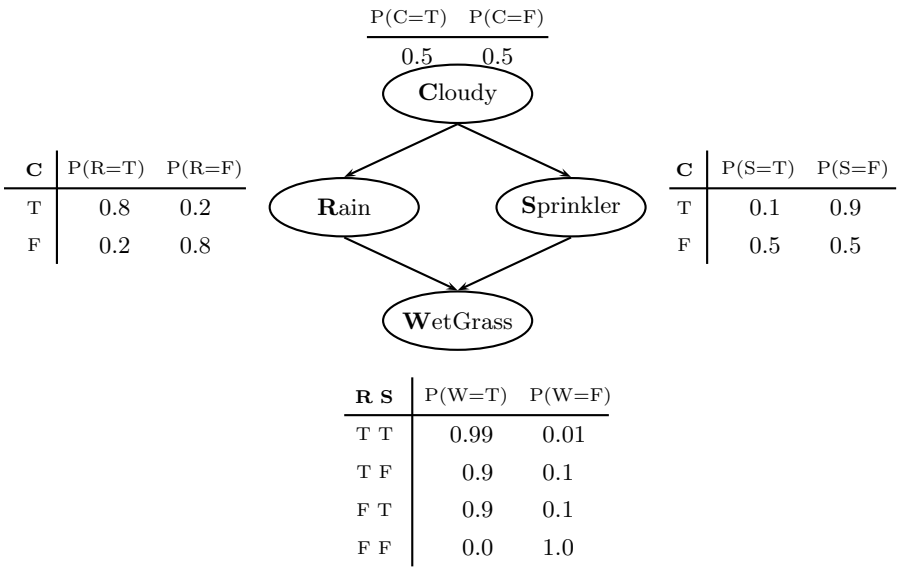

Fig. 1. The Sprinkler Network.

results, including tuning ANT-MPE's parameters and comparing it with four other sampling and local search-based approximate MPE algorithms. Finally we summarize our findings and conclude with some discussions.

2 The MPE Problem

2.1 Bayesian Networks and The MPE Problem

A Bayesian network (Fig.1) is a Directed Acyclic Graph (DAG) where nodes represent random variables and edges represent conditional dependencies between random variables. Attached to each node is a Conditional Probability Table (CPT) that describes the conditional probability distribution of that node given its parents' states. Distributions in a BN can be discrete or continuous. In this paper we only consider discrete ones. BNs represent joint probability distributions in a compact manner. Let $\{X_1, \ldots, X_n\}$ be the random variables in a network. Every entry in the joint distribution $P(X_1, \ldots, X_n)$ can be calculated using the following 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 parent nodes of X_i. Figure 1 shows a simple BN with 4 nodes, the Sprinkler network (Russell and Norvig 2003).

Let (G, P) be a Bayesian network where G is a DAG and P is a set of CPTs, one for each node in G. An evidence E is a set of instantiated nodes. An

explanation is a complete assignment of all node values consistent with E. Each explanation's probability can be computed in linear time using (1). For example, in the Sprinkler network (Fig. 1), suppose we have observed that the grass is wet, i.e. the $E = \{\mathrm{W} = \mathbf{T}\}$. One possible explanation of this is: { C = **T**, R = **T**, S = **F**, W = **T**}. Its probability is:

$$\begin{aligned} & P(C = \mathbf{T}, R = \mathbf{T}, S = \mathbf{F}, W = \mathbf{T}) \\ &= P(C = \mathbf{T})P(R = \mathbf{T}|C = \mathbf{T})P(S = \mathbf{F}|C = \mathbf{T})P(W = \mathbf{T}|S = \mathbf{F}, R = \mathbf{T}) \\ &= 0.5 \times 0.8 \times 0.9 \times 0.9 = 0.324. \end{aligned}$$

MPE is an explanation with the highest probability. It provides the most likely state of the world given the observed evidence. MPE has a number of applications in diagnosis, abduction and explanation. Both exact and approximate MPE are NP-hard (Littman 1999, Abdelbar and Hedetniemi 1998). Therefore approximate and heuristic algorithms are necessary for large and dense networks.

2.2 Related Work

Clique-tree propagation is the most popular exact inference algorithm(Lauritzen and Spiegelhalter 1988). It is efficient for sparse networks but can be very slow and often runs out-of-memory for dense and complex ones. The same is true for other exact algorithms such as variable elimination and cutset conditioning. In fact, all exact inference algorithms share a worst-case complexity exponential in the induced treewidth (same as the largest clique size) of the underlying undirected graph,

Approximate MPE algorithms trade accuracy for efficiency so that they can at least find a near-optimal explanation in a reasonable amount of time on some large instances where exact algorithms fail. There are two basic categories of approximate algorithms: stochastic sampling and search-based algorithms. Their main advantage is that the running time is fairly independent of the topology of the network and linear in the number of samples or search points.

Stochastic sampling algorithms can be divided into importance sampling algorithms (Fung and Chang 1989) and Markov Chain Monte Carlo (MCMC) methods (Pearl 1988). They differ from each other in whether samples are independent or not. Both can be applied to a large range of network sizes. But with a large network and unlikely evidence, the most probable explanation can also be very unlikely. Thus the probability of it being hit by any sampling schemes will be rather low. This is the main weakness of sampling algorithms.

Search algorithms have been studied extensively in combinatorial optimization. Researchers have applied various search strategies to solve MPE, for example, the best first search (Shimony and Charniack 1999), linear programming (Santos 1991), stochastic local search (Kask and Dechter 1999), genetic algorithms (Mengshoel 1999), etc. More recently, Park (2002) tried to convert MPE to MAX-SAT, and then use a MAX-SAT solver to solve it indirectly. Other local search algorithms often use some heuristics to guide the search in order to avoid

getting stuck into local optimal. The most popular heuristics include Stochastic Hillclimbing, Simulated Annealing (Kirkpatrick et al. 1983), Tabu Search (Glover et al. 1997), etc.

3 Ant Algorithms to Solve MPE

Ant algorithms were inspired by the foraging behavior of real ant colonies, in particular, by how ants can find the shortest paths between food sources and nest. Ants deposit on the ground a chemical substance called pheromone while walking from nest to food sources and vice versa. This forms pheromone trails through which ants can find the way to the food and back to home. Pheromone provides indirect communications among ants so that they can make use of each other's experience. It has been shown experimentally (Dorigo, Di Caro and Gambardella 1999) that this foraging behavior can give rise to the emergence of shortest paths when employed by a colony of ants.

Based on this ant colony foraging behavior, researchers have developed ACO algorithms using artificial ant systems to solve hard discrete optimization problems. In an ant system, artificial ants are created to explore the search space simulating real ants searching their environment. The objective values to be optimized usually correspond to the quality of the food and the length of the path to the food. An adaptive memory corresponds to the pheromone trails. Also, the artificial ants can make use of some local heuristic functions to help make choose among a set of feasible solutions. In addition, a pheromone evaporation mechanism is usually included to allow the ant colony to slowly forget its past history. By doing so it can direct the search towards new directions that have not been explored in the past.

ACO was first used on the Travelling Salesman Problem (Dorigo and Gambardella 1997). From then on it has been applied to the Job-Shop Scheduling Problem (Colorni et al. 1994), to the Graph Coloring Problem (Costa and Hertz 1997), to the Quadratic Assignment Problem (Gambardella et al. 1999), to the Vehicle Routing Problem (Bullnheimer 1999), etc.

In the following of this section we describe how to apply ACO to solve the MPE problem.

3.1 An Ant System for MPE

The Ants. In an MPE ant system, artificial ants build MPE solutions (explanations) by moving on the Bayesian network from one node to another. Ants must visit all nodes in the *topological order* defined by the network, i.e. before a node X_i is visited all its parents, $\pi(X_i)$, must be visited. When an ant visit X_i, it must take a *conditional branch* which is a number in the CPT. For evidence nodes E, ants are only allowed to take the branches that agree with E. The memory of each ant contains the nodes it has visited and the branches selected.

The pheromone tables, the heuristic function tables, and the ant decision tables. Each node has 3 tables: *the Pheromone Table (PT)*, *the Heuristic Function Table (HFT)*, and *the Ant Decision Table (ADT)*. All three tables have the same structure as the CPTs. The PTs store pheromone values accumulated on each conditional branch. HFTs represent heuristics used by ants. They are exactly the same as CPTs and are kept unchanged. ADTs are used by ants to make the final decision of choosing which branch to take.

How to update these tables and build the tour. The ADT, $A_i = [a_{ijk}]$, of node X_i is obtained by the composition of the local pheromone trail values τ_{ijk} with the local heuristic values η_{ijk} as follows:

$$a_{ijk} = \frac{[\tau_{ijk}]^{\alpha}[\eta_{ijk}]^{\beta}}{\sum_j [\tau_{ijk}]^{\alpha}[\eta_{ijk}]^{\beta}} \tag{2}$$

where j is the jth row and k the kth column of the corresponding ADT at the ith node. α and β are two parameters that control the relative weight of pheromone trails and heuristic values.

The probability with which an ant chooses to take a certain conditional branch while building its tour is:

$$p_{ij} = \frac{a_{ij\pi_i}}{\sum_j a_{ij\pi_i}} \tag{3}$$

where π_i is the column index of the ADT and its value is conditioned on the values of parent nodes of ith node. This is equivalent to randomly simulate the ADT.

After ants have built their tour (an explanation), each ant deposits pheromone $\Delta\tau_{ijk}$ on the corresponding pheromone trails (the conditioned branches of each node on the tour). The pheromone value being dropped represents solution quality. Since we want to find the most probable explanation, we use the probability of the selected tour as the pheromone value. Suppose the generated tour is $\{x_1, \ldots, x_n\}$, the pheromone value is as follows:

$$\Delta\tau_{ijk} = \begin{cases} P(x_1, \ldots, x_n) & \text{if } j = x_i, k = \pi(x_i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where the $P(x_1, \ldots, x_n)$ is computed by the chain rule in (1).

Updating the PTs is done by adding a pheromone value to the corresponding cells of the old PTs. Each ant drops pheromone to one cell of each PT at each node, i.e., the jth row, kth column of the PT at ith node. After dropping the pheromone, an ant dies. The pheromone evaporation procedure happens right after the ants finish depositing pheromone. The main role of pheromone evaporation is to avoid stagnation when all ants end up selecting the same tour. In summary, PTs are updated by the combination of pheromone accumulation and pheromone evaporation as follows:

$$\tau_{ijk} = (1 - \rho)\tau_{ijk} + \Delta\tau_{ijk} \tag{5}$$

Algorithm 1 ANT-MPE

1: **Input** — an MPE instance (G, P, E);
2: **Output** — an explanation $u = (u_1, \ldots, u_n)$;
3: **Begin** ANT-MPE(G, P, E)
4: **Initialization**: initialize α, β, ρ, *n_iterations*, *n_ants*, *best_trail*, PTs, HFTs, ADTs;
5: **while** *n_iterations* > 0 **do**
6: updating ADTs using PTs and HFTs;{Equation 2}
7: generating *n_ants* ant trails by random sampling from ADTs according to the topological order;{Equation 3}
8: computing each trail's probability and updating *best_trail*;{Equation1}
9: updating PTs by dropping pheromone, pheromone evaporation;{Equation 4 & 5}
10: *n_iterations* $--$;
11: **end while**
12: **Return** *best_trail*;
13: **End** ANT-MPE.

where $\tau_{ijk} = \sum_{l=1}^{m} \Delta\tau_{ijk}$, m is the number of ants used at each iteration, and $\rho \in (0, 1]$ is the pheromone trail decay coefficient.

3.2 The ANT-MPE Algorithm

Given the above ant system, we design an ACO algorithm, ANT-MPE, for MPE. It is listed in Algorithm 1. In the initialization step, we set pheromone values to a small positive constant on all pheromone tables, set ADTs to 0, and set HFTs to the same as CPTs. After initialization, we generate a batch of ants for several iterations. At each iteration, we first update the ant decision tables from the current pheromone tables and the heuristic function tables. Then ants use the ant decision tables to build tours, evaluate them by CPTs, and save the best trail. Then pheromone is dropped and the pheromone tables are updated. The pheromone evaporation is triggered right after. This procedure stops when the number of iterations runs out. The best solution so far is returned as the approximate MPE.

4 Results

4.1 Test Datasets

The CPT skewness of a network is computed as follows (Jitnah and Nicholson 1996): for a vector (a column of the CPT table), $v = (v_1, v_2, \ldots, v_m)$, of conditional probabilities,

$$skew(v) = \frac{\sum_{i=1}^{m} |\frac{1}{m} - v_i|}{1 - \frac{1}{m} + \sum_{i=2}^{m} \frac{1}{m}}. \quad (6)$$

where the denominator scales the skewness from 0 to 1. The skewness for one 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 used both real world and randomly generated networks to test ANT-MPE. CPT skewness was used as the main control parameter when generating random networks because we knew from domain knowledge that it would affect the instance hardness for sampling and local search algorithms. We had collected 11 real world networks. The size and skewness of these real world networks are listed in Table 6. We can see that on average most real world networks are skewed. In fact only one network's skewness(*cpcs54*) is less than 0.5 and the average skewness is about 0.7. In our experiment, we considered three different levels of skewness: {skewed(0.9), medium(0.5), unskewed(0.1)}.

The number of nodes we used for random network generation were 100 and 200. All networks were too dense to be solved exactly. All nodes were binary variables. Another factor was the evidence. In all experiments, 10% nodes were randomly selected as the evidence nodes and their values were also randomly selected. The default number of ants for each experiment was set to 3,000.

4.2 Experiment 1: Tuning α, β, and ρ in ANT-MPE

In experiment 1 we used 100 randomly generated multiply connected networks to tune the parameter values in ANT-MPE. These networks were divided into 5 groups by their skewness and number of nodes: {*skewed*100, *medium*100, *unskewed*100, *medium*200, *unskewed*200}. Each group contained 20 networks.

The weight of pheromone trails, α, and the weight of local heuristic function, β, are two most important parameters for ant algorithms. We first ran ANT-MPE on all 100 networks with 5 different combinations of (α, β) values: {(0, 5), (1, 0), (1, 5), (3, 5), (5, 1)}. The pheromone evaporation rate ρ was set to 0.01 for all runs. We gave each parameter setting 3,000 ants and compared the approximate MPE values returned. When a parameter setting returned the highest value, we counted one "win" for it. When it returned the lowest value, we counted one "loss" for it. Note that they could tie with each other.

The result is listed in Table 1. We can see that: (1) When $\beta = 0$, the local heuristic function was not being used, it never won and lost 97 out of 100 times. When $\beta = 5$, it never lost and the number of wins increased to around 60. This indicates the importance of local heuristic function, i.e. the conditional probability tables. (2) When we set β to its best value 5 and let α be 0, 1 and 3, number of wins peaked at $\alpha = 1$ as 65. This can be explained as follows: when $\alpha = 0$, the communications between ants are not exploited so the search performance will be correspondingly affected; when $\alpha = 3$, the communications are overemphasized and the search can be trapped into local optima too early. (3) Different parameter settings tied with each other more frequently on skewed networks than on unskewed networks. This was because skewed networks' solution spaces were also skewed thus making them easier for ant algorithms comparing to those unskewed ones. Basically most parameter settings were able to find the

Table 1. Different (α, β) values on skewed, medium, unskewed networks with 100 and 200 nodes.

	skewed100		**medium100**		**unskewed100**		**medium 200**		**unskewed 200**		**total**	
(α, β)	**win**	**loss**	**win**	**loss**	**win**	**loss**	**win**	**loss**	**win**	**loss**	**win**	**loss**
(0, 5)	17	0	16	0	6	0	14	0	6	0	59	0
(1, 0)	0	20	0	20	0	18	0	20	0	19	0	97
(1, 5)	17	0	15	0	8	0	15	0	10	0	65	0
(3, 5)	17	0	15	0	6	0	16	0	4	0	58	0
(5, 1)	16	0	0	0	0	2	0	0	0	1	16	3

same best MPE. Also note that on these more difficult unskewed networks, $(1, 5)$ always got the best performance.

So we took (1, 5) as the best (α, β) values. This result also agreed with Dorigo's finding (1997) on the TSP problem. We used it as our default (α, β) values in all other experiments. We also tuned ρ in the same way using 5 different values: {0, 0.001, 0.01, 0.05, 0.5}. But the results did not show the dominance of any ρ value over others excepted that 0.1 was slightly better than others. So we just used $\rho = 0.1$ in all other experiments. Because of the lack of space, we do not list the detail results here.

4.3 Experiment 2: Algorithm Comparison on Randomly Generated Networks

In experiment 2, we compared ANT-MPE with four other sampling and local search-based approximate MPE algorithms: Forward Sampling, Gibbs Sampling, Multi-Start Hillclimbing, and Tabu Search on two randomly generated test datasets. Again, all networks were exactly intractable. On the first test dataset, we ran all algorithms and counted the number of times each algorithm "won" the competition. So far, the most effective way to fairly compare different heuristic algorithms is to allow all algorithms to consume the same amount of computation resources, with distinctions being based on the quality of solutions obtained (Rardin 2001). In our experiments, we gave each algorithm a given number of samples(or equivalently, ants and search-points) and then compared the quality of solutions returned. The algorithm returned the highest MPE was labelled as "winner". We also record when the highest MPE was found by each algorithm. If two algorithms returned the same value, the one that used less resources was labelled as "winner". On the second test dataset, we compared the total approximate MPE values returned by each algorithm.

Experiment 2.1: Algorithm Comparison by Number of WINs. The test dataset here contained 2,592 randomly generated MPE instances. Number of nodes was set to 100. The skewness had three levels: skewed(0.9), medium(0.5), or unskewed(0.1). Each level contained 864 instances. Number of samples had

Table 2. Experiment 2.1: number of times of each algorithm being the best.

	Best Algorithm				
	Gibbs Sampling	Forward Sampling	Multi-start HC	Tabu Search	ANT-MPE
counts	0	366	697	139	1,390
percentage	0%	14.12%	26.89%	5.36%	53.63%

Table 3. Experiment 2.1: grouped by #samples.

#samples	**Number of Times of Being the Best Algorithm**				
	Gibbs Sampling	Forward Sampling	Multi-start HC	Tabu Search	ANT-MPE
300	0	106	283	0	475
1,000	0	124	262	3	475
3,000	0	136	152	136	440

Table 4. Experiment 2.1: grouped by skewness.

skewness	**Number of Times of Being the Best Algorithm**				
	Gibbs Sampling	Forward Sampling	Multi-start HC	Tabu Search	ANT-MPE
0.1	0	0	694	135	35
0.5	0	0	1	1	862
0.9	0	366	2	3	493

three values as well: 300, 1,000, or 3,000. Each group also contained 864 instances. The results are summarized in Table 2, Table 3 and Table 4.

Table 2 basically shows that in general ANT-MPE outperforms all other algorithms. Table 3 shows that number of samples does not significantly affect ANT-MPE. It only slightly influences two search algorithms' relative performance. Table 4 gives the most interesting result. We can see that (1) on skewed networks ANT-MPE generally outperforms all other algorithms, while Forward Sampling still can compete; (2) on medium networks, ANT-MPE dominates; (3) on unskewed networks, search algorithms outperforms ANT-MPE. Fortunately, most real world networks are not unskewed. This is because skewness in fact indicates the structure of the probabilistic domain and real world distributions are all structured to some degree. Therefore we can expect that ANT-MPE would work well on most real world networks.

Experiment 2.2: Algorithm Comparison by the Returned MPE Probabilities. In this experiment we ran all algorithms on 162 randomly generated networks. They were divided into three groups: 27 unskewed networks, 54 medium networks, and 81 skewed networks. For each group, we collected the

Table 5. Experiment 2.2: Total MPE Probabilities Returned by Each Algorithm.

skewness	Total MPE Probabilities Returned by Each Algorithm				
	Gibbs Sampling	Forward Sampling	Multi-start HC	Tabu Search	ANT-MPE
0.1	4.7×10^{-29}	1.0×10^{-27}	2.2×10^{-26}	7.9×10^{-27}	1.1×10^{-26}
0.5	1.4×10^{-14}	2.8×10^{-8}	6.2×10^{-9}	1.6×10^{-7}	6.0×10^{-6}
0.9	1.9×10^{-46}	0.14	1.4×10^{-10}	2.5×10^{-14}	0.16

total approximate MPE probabilities returned by each algorithm. The result is shown in Table 5. It shows that in terms of the returned MPE probabilities, ANT-MPE outperforms all other algorithms on both skewed and medium networks. On unskewed networks, Multi-start Hillclimbing(2.2×10^{-26}) is only slightly better than ANT-MPE(1.1×10^{-26}). ANT-MPE is the second best and it is still at the same order of magnitude as Multi-start Hillclimbing. So we can draw the conclusion that in general ANT-MPE outperforms all other algorithms.

4.4 Experiment 3: Algorithm Comparison on Real Networks

Table 6. Test Results on 11 Real World Bayesian Networks.

Network	#Nodes	Skew-ness	Exact MPE	Best Appro. Algorithm	Returned by Best Algo.	Returned by ANT-MPE
alarm	37	.84	0.04565	Forward Sampling	0.04565	0.04565
barley	413	.87	3.67×10^{-37}	N/A	N/A	N/A
cpcs179	179	.76	0.0069	ANT-MPE	0.0069	0.0069
cpcs54	54	.25	1.87×10^{-11}	Multi-start HC	1.28×10^{-11}	5.78×10^{-12}
hailfinder	56	.50	1.44×10^{-12}	ANT-MPE	2.11×10^{-13}	2.11×10^{-13}
insurance	27	.70	0.002185	Forward Sampling	0.002185	0.002185
pigs	441	.55	5.03×10^{-88}	ANT-MPE	1.31×10^{-141}	1.31×10^{-141}
water	32	.75	3.08×10^{-4}	ANT-MPE	3.08×10^{-4}	3.08×10^{-4}
munin2	1003	.89	8.74×10^{-37}	ANT-MPE	1.23×10^{-37}	1.23×10^{-37}
munin3	1041	.55	2.49×10^{-37}	ANT-MPE	7.07×10^{-40}	7.07×10^{-40}
munin1	189	.88	N/A	ANT-MPE	5.93×10^{-8}	5.93×10^{-8}

In experiment 3 we ran on all algorithms on 11 real world networks. Each run was given 3,000 samples. We compared the exact MPE probability, the MPE probability returned by the best approximate algorithm, and the MPE probability returned by ANT-MPE. We used Hugin to compute the exact MPE. The result is listed in Table 6.

ANT-MPE was the best for 7 of 10 networks where the results were available. Forward Sampling were the best for *alarm* and *insurance* because they returned

the MPE earlier. But ANT-MPE was able to find the same MPE as well. Multi-start Hillclimbing outperformed ANT-MPE on *cpcs54*, whose skewness was only 0.25. But ANT-MPE was the second best on *cpcs54* and Multi-start Hillclimbing was only slightly better. We can say that in general ANT-MPE outperformed all other algorithms on the real world test dataset.

5 Concluding Remarks

We have described an ant algorithm, the ANT-MPE, for the MPE problem. To our knowledge, this is the first application of ACO to MPE. The empirical results show that in general ANT-MPE outperforms other sampling and search algorithms on both artificial and real networks. More specifically, on skewed networks ANT-MPE generally outperforms other algorithms, but Forward Sampling are competent; on medium networks ANT-MPE basically dominates; on unskewed networks, local search algorithms outperform ANT-MPE, but they are only slightly better and ANT-MPE is the second best.

This result is interesting because it reveals ant algorithms' nature as a combination of sampling and local search. The sampling part comes from the fact that each individual ant can use CPTs as heuristic functions to explore new trails. The search part is that a colony of ants can exchange information through pheromone trails so as to cooperatively "learn" how to find the best solution. Basically, if we set α to 0, then ACO becomes Forward Sampling, Because it only uses CPTs as the heuristic functions when generating ant trails(samples). With the use of pheromone trails($\alpha \neq 0$), ANT-MPE manages to outperform Forward Sampling on both skewed and medium networks. As the skewness decreases, the solution space becomes more "flat" and the number of local optima increases. It is well-known that as the number of local optima increases, most likely the search space becomes harder to explore. This makes it more difficult for sampling algorithms, while simple search heuristic like random restart will have more chances to explore new areas in the solution space. That is why search algorithms outperform ANT-MPE on unskewed networks. This result implies that as a combination of sampling and local search, ACO's search aspect is weaker than its sampling aspect. This can be verified by the importance of β values as shown in experiment 1. It also suggests a possible way to improve ACO. If we can detect that the solution space is flat, then we can change ants' strategy to favor exploration more than exploitation so as to gain a better overall performance.

Possible future work include conducting similar algorithm comparison experiments on other NP-hard problems to see if the same conclusion regarding to instance hardness and algorithm performance can be drawn there.

Acknowledgements

Thank 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. Artif. Intell. **102** (1998) 21–38

Bullnheimer, B.: Ant Colony Optimization in Vehicle Routing. Doctoral thesis, University of Vienna. (1999)

Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M.: Ant system for Job-Shop Scheduling. Belgian Journal of Operations Research, Statistics and Computer Science. **34(1)** (1994) 39–53

Costa, D., Hertz, A.: Ants can colour graphs. Journal of the Operational Research Society. **48** (1997) 295–305

Dorigo, M.: Optimization, Learning and Natural Algorithms. Ph.D.Thesis, Politecnico di Milano, Italy. (1992)

Dorigo, M., Di Caro, G., Gambardella, L. M.: Ant algorithms for discrete optimization. Artificial Life, **5(2)** (1999) 137-172

Dorigo, M., Gambardella, L. M.: Ant Colonies for the Traveling Salesman Problem BioSystems. **43** (1997) 73-81

Fung, R., Chang, K. C.: Weighting and integrating evidence for stochastic simulation in Bayesian networks. In Uncertainty in Artificial Intelligence 5. (1989) 209–219

Gambardella, L. M., Taillard, E., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society. **50** (1999) 167–176.

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

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

Kask, K., Dechter R.: Stochastic local search for Bayesian networks. In Workshop on AI and Statistics 99. (1999) 113–122

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P.: Optimization by simulated annealing. Science, Number 4598. **220** (1983) 671–680

Littman, M.: Initial experiments in stochastic search for Bayesian networks. In Proceedings of the Sixteenth National Conference on Artificial Intelligence. (1999) 667–672

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

Mengshoel, O. J.: Efficient Bayesian Network Inference: Genetic Algorithms, Stochastic Local Search, and Abstraction. Computer Science Department, University of Illinois at Urbana-Champaign. (1999)

Park, J. D.: Using weighted MAX-SAT engines to solve MPE. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI). (2002) 682–687

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

Rardin, R. L., Uzsoy, R.: Experimental evaluation of heuristic optimization algorithms: a tutorial. Journal of Heuristics. **7** (2001) 261–304

Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, NJ. (2003)

Santos, E.: On the generation of alternative explanations with implications for belief revision. In UAI91. (1991) 339–347

Shimony, S. E., Charniak, E.: A new algorithm for finding MAP assignments to belief network. In UAI 99. (1999) 185–193