FEASIBLE UNCERTAIN REASONING FOR MULTI AGENT ONTOLOGY MAPPING

Miklos Nagy[1], Maria Vargas-Vera[2], Enrico Motta[1]
Knowledge Media Institute[1], Computing Department[2]
The Open University
Walton Hall, Milton Keynes
United Kingdom

ABSTRACT

One of the main disadvantages of using Dempster-Shafer theory for uncertain reasoning is the computational complexity of the belief combination. Large number of variables can easily make the applicability unfeasible due to the exponential growth of the problem space. Semantic Web applications like ontology mapping usually exploits different kind of background knowledge in order to augment the available information which increases the number of variables considerably in the reasoning process. Therefore optimalisation is necessary in order to provide a feasible uncertain reasoning for ontology mapping with large number of variables. In this paper we introduce a novel genetic algorithm solution which is based on distributed junction tree optimalisation for a multi agent system.

KEYWORDS

Uncertain reasoning, Multi-agent systems, Semantic Web, Genetic Algorithm

1. INTRODUCTION

Dempster-Shafer theory offers an alternative to traditional Bayesian theory for the mathematical representation of uncertainty. The advantages of this framework are that it allows for the allocation of subjective probability masses to sets of variables and it allows proper representation of ignorance. Additionally, probability masses do not need to be assessed a priory for all variables as for traditional Bayesian frameworks. In spite of the fact that the theory is well-founded and general model of human reasoning under uncertainty, Dempster-Shafer theory is rarely used in Semantic Web applications. One of the most significant obstacles that discourage practical implementations is the relatively high computational complexity, especially in comparison with methods based on classical probability theory. In fact, combining belief functions using Dempster's rule of combination is known to be #P-complete (Orponen 1990) in the number of evidential sources.

The computational complexity of the combination can be tackled by approximation (Bauer-1996, Harmanec 1999) methods or optimalisation based on local computation (Bissig et.al. 1997, Shenoy 1988) which uses different message passing schemes in a graph structure called junction tree. The junction tree has a remarkable advantage from the distributed computing point of view because in theory it can easily be split into different parts which make it ideal for sharing the computational complexity between processors.

In this paper we propose a distributed belief function combination which can be applied in multi agent architecture in order to split up the problem space into manageable portions.

2. PROBLEM DESCRIPTION

Ontology mapping can be defined as a process of finding related concepts in different semantic metadata such as ontologies on the Semantic Web. It is the starting point for the semantic data integration where the domain knowledge needs to be aligned or merged. Ontology mapping algorithms can use different kind of background knowledge e.g. WordNet, Semantic Web (Nagy et.al. 2006, Sabou et.al. 2006) in order to

provide better ontology mappings. From the reasoning point of view this additional knowledge increases the number of variables in the system. As a consequence these additional variables increase the computational complexity of the reasoning process, which is important if the mapping system need to provide response in real time (Nagy et.al. 2006).

Using the Dempster Shafer theory for managing the uncertain aspects (Nagy et.al. 2006) of ontology mapping can improve the mapping precision but certain limitation needs to be introduced in order to keep the response time of the mapping algorithm within acceptable limits. One of these limitations as stated in (Nagy et.al. 2006) is to limit the number of additional variables that stem from the utilisation of the background knowledge in order to avoid the problem of large problem space which grows exponentially as the number of variable increases. However the introduction of any limitation can negatively affect the mapping precision because it implies that some background knowledge cannot be completely utilised. In order to make the belief combination and reasoning feasible for large number of variables we need to apply optimalisation during the computation. Junction tree based optimalisations are ideal for distributed architecture because the problem space can be split up into several portions. In practice however a near optimal junction tree does not guarantee that the split will result in equal portions in terms of variable size which hinders the practical distributed implementation because the computational load cannot be distributed equally. The main contribution of this paper is the introduction of a distributed joint tree construction algorithm which makes it possible to carry out uncertain reasoning with large number of variables using Dempster Shafer theory of evidence for ontology mapping on the Semantic Web.

3. FORMALISATION

3.1 Valuation Network

Valuation algebra framework has been proposed by Shenoy (Shenoy 1989) in order to represent knowledge about variables on a specific domain. In a valuation based system the knowledge is represented by functions called valuations and inferences are made using two operators

1. marginalisation
2. combination

A graphical depiction of a valuation-based system is called a valuation network where each valuation node is connected by an undirected edge to each variable node in its domain.

$$\theta_D = \prod_{x \in D} \theta_x$$

θ_D represents the state space for x which denotes a configuration for sets of variables on the domain D. By definition valuation φ, γ , etc represents some knowledge about the possible values of a set x of variables. So, each valuation refers to some defnite set of variables from the domain D.

The main advantage of the valuation algebra is that it is abstract enough to include many different probabilistic formalisms like Bayesian networks or Dempster-Shafer theory. Inference can be carried out by using two operations, combination and marginalisation.

The marginalisation corresponds to coarsening the knowledge.

$$m_\varphi \downarrow_x (A) = \sum_{C \subseteq \theta_D, C \downarrow x \cap D} m_\varphi (C)$$

where m_φ is the belief mass function and C is the subset of θ_D marginalised to the variables x. The combination operator is commutative and associative and corresponds to the aggregation of the knowledge.

$$m_{\varphi \otimes \gamma} (A) = \sum_{A_1 \cap A_2} m_\varphi \downarrow_x (A_1) * m_\gamma \downarrow_x (A_2)$$

Further advantage of the valuation algebra is that it allows computing marginals of a combination of valuations without explicitly computing the combination. This is called local computation because through message passing in the junction tree we compute marginals of the joint distribution without actually computing the joint distribution.

3.2 Junction Tree

Junction tree is a graphical structure which facilitates parallel computation of multiple queries. In practice it can be represented by an undirected tree τ where each node in τ is a cluster (nonempty set) of variables. In order to form a junction tree from these clusters the following conditions should be satisfied:

1. Singly connected: there is exactly one path between each pair of clusters.
2. Covering: for each clique *A* of *G* there is some cluster *C* such that $A \subseteq C$.
3. Running intersection: for each pair of clusters *B* and *C* that contains *i*, each cluster on the unique path between *B* and *C* also contains *i*.

As an example assume that the ontology mapping problem is represented by the following variables $O_1\{S\}$, $O_2\{A,B,C,D,E,F\}$ where *S* is the source concept and *A*, *B*, C, *D*, *E*, *F* represents potential target concepts that can be mapped to our source. The variables represent the following concepts form the ontology:

- *S*=publication
- *A*=paper
- *B*=communication
- *C*=article
- *D*=document
- *E*=abstract
- *F*=issue

Once similarities have been assigned the target variable combinations have been identified as candidate mappings as depicted on Fig 1a.

$$O_1\{S\} \Rightarrow O_2\{A,B\}$$
$$O_1\{S\} \Rightarrow O_2\{A,C\}$$
$$O_1\{S\} \Rightarrow O_2\{B,D\}$$
$$O_1\{S\} \Rightarrow O_2\{B,F\}$$
$$O_1\{S\} \Rightarrow O_2\{E,F\}$$
$$O_1\{S\} \Rightarrow O_2\{C,E\}$$
$$O_1\{S\} \Rightarrow O_2\{D\}$$

Figure 1a.

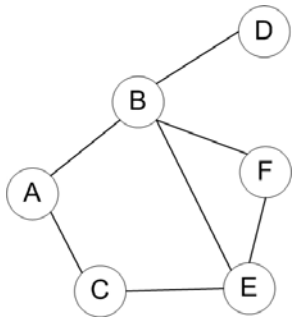

Figure 1b.Undirected graph representation

For each candidate mapping a subjective probability represents the strength of the evidence which supports that the source concept is equivalent with one or all of the target concepts. These evidences are the result of an observation as defined by the Dempster-Shafer theory. The dependencies between the target variables can be represented by an undirected graph as depicted on Fig. 1b.

Creating a junction tree involves different graph transformation operations which includes

1. Moralise the undirected graph
2. Triangulate the moralised graph
3. Transform the triangulated graph to junction graph and tree using variable elimination.

Moralising the graph means that we take the undirected graph, add a link between any pair of variables with a common child. The resulting graph is the moral graph. From the moral graph one can find the clusters, namely the cliques in the graph. Moral graph can contain several loops which violates the running intersection property. In order to resolve multiple loops we need to triangulate the graph which results in a graph where every loop of length 4 or more has a chord. Triangulation is a method which will guarantee that we can always form a junction tree from the resulting junction graph by increasing the minimum clique size, sometimes substantially. Creating the junction graph between any two clusters with a non-empty intersection can be achieved by adding a link with the intersection as the separator. The resulting graph is called a junction graph. All separators consist of a single variable, and if the junction graph contains loops, then all separators on the loop contain the same variable. Therefore any of the links can be removed to break the

loop, and by removing links until you have a tree, you get a junction tree. One possible junction tree for the ontology mapping example is depicted on Fig. 2.

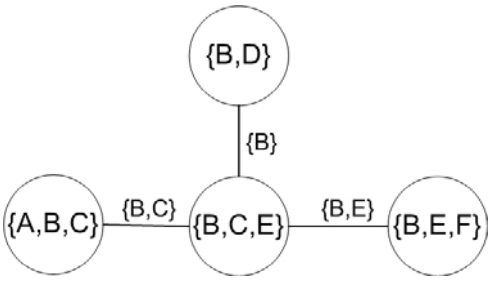

Figure 2. Junction tree.

In practice the junction tree creation process is not deterministic i.e. it is possible to built different junction trees for the same problem with the same variables. As an example different heuristics have been proposed by the research community for determining different variable elimination sequences which can lead to different triangulation and junction tree of the same graph. Additionally creating an optimal junction tree is known to be an NP complete problem (Arnborg et.al.1987). Further problem is the time constraint which is especially important in the context of our ontology mapping system where the user should expect some kind of interaction with the system. Nevertheless besides the above mentioned drawbacks there is one considerable advantage of using junction trees is the fact that once it is created it can easily be distributed hence the computation load can be divided. In order to overcome the before mentioned problems we propose a distributed genetic algorithm which finds good enough junction tree that makes uncertain reasoning feasible with large number of variables under time constraint.

3.3 Junction Tree Algorithm

In order to create a junction tree we propose using genetic algorithm which works under limited time constraint. This is especially important because in our multi agent ontology mapping framework we need to create mappings in real time and interact with the users if necessary. The main advantages of using genetic algorithm for finding a good junction tree are as follows:

1. Evolution can be distributed between agents
2. Once a time limit is reached the most fittest candidate can be selected from the population
3. Our objective is to create an algorithm which finds a good junction tree that can be distributed into our different agents so the reasoning process for the mapping selection can be carried out within acceptable time frame.

Using genetic algorithm for determining variable elimination for the triangulation of the undirected graph has been shown to be effective (Gomez et.al. 2004, Hao et.al. 2006) for creating optimal junction tree. However the ongoing research effort is aimed at creating an optimal junction tree for serial processing environment. Our approach on the junction tree creation differs from the existing approaches in terms of how the problem is represented by the genetic algorithm.

3.4 Junction Tree Chromosome

The chromosome or genome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve. Each individual is characterized by its chromosome S_k, which determines the individual fitness $f(S_k)$; $k = 1,...,n$; n is a population size. The chromosome is a string of symbols, $S_k = (S_{k1}, S_{k2}, S_{kN})$, N is a string length. The symbols S_{k1} are interpreted as genes of chromosome S_k. In order to represent the junction tree we use composite genes (Fig. 3) where each gene represents a clique in the tree.

Figure 3. Chromosome representation.

The size of the chromosome depends on the problem we are trying to solve and is equivalent to number of agents we have in the system. As an example assume we have 5 agents. In this case the chromosome size should be 5 as well because each agent is going to reason on the domain which is represented by the clique in the junction tree. As a consequence our objective is to create a junction tree which contains more or less equivalent number of variables in their cliques because in this case the computation of the reasoning process can be distributed equally. This is of course an ideal scenario which cannot always be created in practice.

3.5 Junction Tree Fitness

Fitness evaluation involves defining an objective or fitness function against which each chromosome is tested for suitability for the environment under consideration. As the algorithm proceeds we would expect the individual fitness of the best chromosome to increase as well as the total fitness of the population as a whole. For evaluating the chromosome our fitness function describes how well the junction tree is balanced and if it fulfils the junction tree criteria or not.
Assume as an example consider the following chromosomes and their fitness value as calculated by the function (Fig. 4).

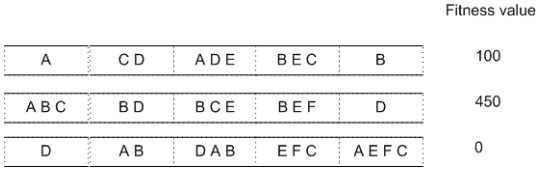

Figure 4. Chromosome fitness

The value of the fitness depends on different criteria like the number of evidences covered by the different cliques or the difference between the number of variables per clique. Our fitness function assigns 0 fitness to the chromosomes even if there is a possibility that the chromosome becomes acceptable after a certain number of evolutions because our mutation rate has to be high in order to replace the wrong chromosomes with random once again. Additionally we do not define maximum fitness for the chromosome because the good junction tree can be different from case to case. In our algorithm the fitness function is used to determine if it is possible to achieve improvement in the population.

3.6 Junction Tree Algorithm

Once a suitable representation has been decided upon for the chromosomes, it is necessary to create an initial population to serve as the starting point for the genetic algorithm. This initial population is usually created randomly. We need to select chromosomes from the current population for reproduction. If we have a population of size 2N, the selection procedure picks out two parent chromosomes, based on their fitness values, which are then used by the crossover and mutation operators (described below) to produce two offspring for the new population. This selection/crossover/mutation cycle is repeated until the new population contains 2N chromosomes i.e. after cycles. The higher the fitness value the higher the probability of that chromosome being selected for reproduction.

```
    input : Similarities,Variables
    output: List of Junction tree cliques
1 Initialise random population ;
2 for i = 1 to maxEvolution do
3         Get the fittest chromosome;
4         if CurrentFitness > LatestCurrentFitness then
5                 Store the fittest chromosome;
6         else
7                 Finish evolution;
8         Execute Best chromosome natural selector;
9         Execute Crossover operator;
10        Execute Mutation operator;
```

Algorithm 1. Junction tree algorithm

4. EXPERIMENTS

We have carried out experiments with randomly generated scenarios in order to evaluate our junction tree algorithm. Our main objective was to evaluate the average computational effort for creating the junction tree for different agents and measure the average fitness that is produced by our algorithm.

Our first test set aimed to measure how much time is necessary in order to create a junction tree. For this test we have defined 25 variables which represent the possible solutions in the ontologies i.e. concepts or properties. Using Dempster Shafer combination 25 distinct variables would result in a possible problem space with the size $2^{25} = 33,554,432$. For our experiments we have defined two sets of experiment where we consider using 5 agents which is represented by a chromosome with 5 genes. We have simulated 1000 junction creation cycles where we have measured the number of evolutions that was necessary to evolve a chromosome which describes a junction tree for the agents. In every cycle we have evolve the population till the point where the evolution was able to produce a fitter individual compared to the previous generation. The maximum number of evolution was set to 100 because this is the limit that represents a time constraint that can be used in order to make our algorithm responsive from the user point of view. The initial population size in every cycle was set to 50 and the initial populations were generated randomly. In each cycle we have simulated a random number of similarity assignments hence evidences with a random number of variables in each set. The result of our first experiment is depicted on Fig. 5a.

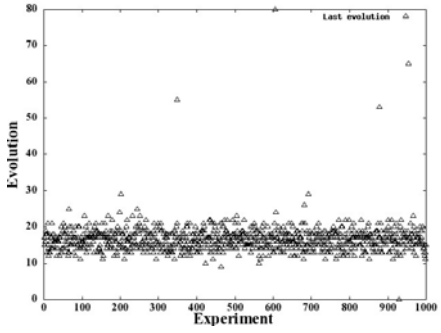

Figure 5a. Number of evolutions

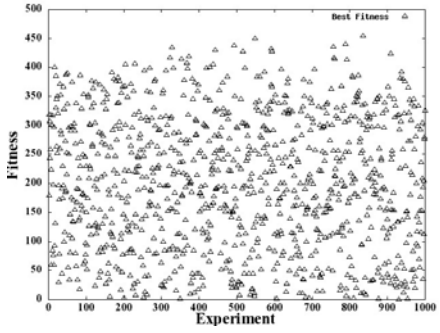

Figure 5b. Fitness at last evolution

Results have shown that the average number of necessary evolutions to create the junction tree was relatively stable for all tests which are encouraging considering our original objective that our process should operate under time constraint. However acknowledging this fact is not enough to deduct that our system will perform relatively equally with different variables. In order to evaluate the fitness of these junction trees we have measured the how well these junction trees meet our criteria. Therefore our second experiment aimed to evaluate how good junction trees we could produce at each cycle (Fig. 5b).

The fitness results could be interpreted as far less encouraging. As depicted on Fig. 6 the best values which could be reached during the experiments varies a lot and was not stable during our experiment. One reason for this instability can be explained with our experimental setup. For this test we have randomly generated 1000 scenarios. In each scenario the number of evidences that is available for the fitness evaluation of the chromosomes is different from scenario to scenario as depicted on Fig. 6a.

To prove our assumption we need to examine if there is a relation between the randomly generated evidences and the fitness functions that was reached at the evolution when the algorithm terminated. This relation is depicted on Fig. 6b.

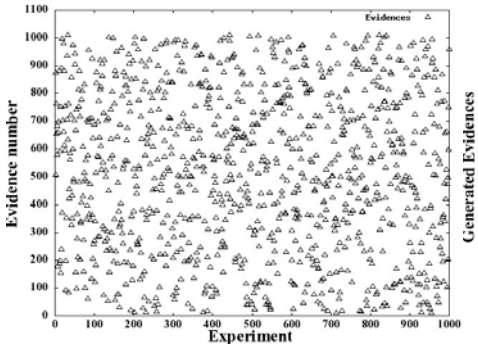

Figure 6a. Evidence number for experiments

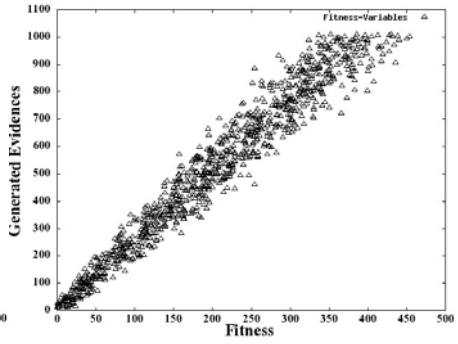

Figure 6b. Generated evidences and fitness

The dependency between the available evidences and the fitness function is a necessary condition however it results in an unfortunate instability throughout the randomly generated scenarios. Nevertheless it is necessary to assume that in real ontology mapping scenarios the number of evidences can vary from mapping to mapping. Nevertheless the experiments have shown the overall approach is promising however a number of questions remained unanswered. This will be addressed in our future research.

5. RELATED WORK

Probabilistic inference methods based on graphical representations are widely used and investigated in spite of the fact that it requires excessive computational resources. The popularity of these solutions can be explained that the problem space can be decomposed into smaller subspaces corresponding to cliques in the triangulated graph. Nevertheless different methods have been proposed for the junction tree creation. First solutions were based on different heuristics like minimum degree and deficiency (Rose 1976) or lexicographic search (Rose 1972). Experiments have proved that using heuristics can provide fairly satisfactory triangulations hence junction trees however optimal result is not guaranteed. As a consequence other researchers have proposed stochastic algorithms like simulated annealing (Kjaerulff 1992) which provides optimal solutions especially for sparse and dense graphs but with increased computational time. In order to address the computational time problem of the stochastic methods genetic algorithms (Larranaga et.al. 1997, Hao et.al. 2006) have been proposed. Using different crossover and mutation techniques experimental results have proved that considering the optimal solution the genetic algorithms can improve the results of the heuristic methods and were comparable to the simulated annealing. Additionally reasonable time constraint can also be imposed by setting the evolution number to a reasonable limit.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a method based on genetic algorithm which is a prerequisite for the efficient reasoning that can be distributed between agents by creating a junction tree using genetic algorithm. Our solution is especially suitable for a multi agent ontology mapping where the mapping system should interact with the users and where the number of possible variables for the mapping selection is large. Our solution for the junction tree creation differs from existing approaches in a way that we represent the problem differently in the chromosomes. We have carried out experiments with standard genetic operators which provided

reasonable evolution in terms of response time because the number of evolutions is relatively low. In our future research we will investigate using different genetic operators and fitness function that can be used to create a junction tree with more equal cliques which can lead to equal computation time for the agents. Further we intend to investigate the message passing strategy for the agents in order to create the reasoning process efficiently.

REFERENCES

Arnborg, S., et al.,1987. Complexity of finding embeddings in a k-tree, *SIAM J. Algebraic Discrete Methods*, 8(2), 277–284.

Bauer, M., 1996. Approximations for decision making in the Dempster- Shafer theory of evidence, In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pp. 73–80,

Bissig, R., et al., 1997. Fast-division architecture for Dempster-Shafer belief functions, In *Proceedings of the 1st International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU-FAPR'97)*, volume 1244 of Lecture Notes in Computer Science, pp. 198–209. Springer,

Gomez, M., Bielza, C., 2004, Node deletion sequences in influence diagrams using genetic algorithms, *Statistics and Computing*, 14(3), 181–198, (2004).

Harmanec, D.,1999, Faithful approximations of belief function, In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 271–27, San Francisco, CA, Morgan Kaufmann.

Kjaerulff, U., 1992. Optimal decomposition of probabilistic networks by simulated annealing, *Statistics and Computing*, 2, 7–17.

Larranaga, P., et al., 1997. Decomposing bayesian networks: triangulation of the moral graph with genetic algorithms, *Statistics and Computing*, 7(1), 19–34.

Nagy, M., et al., 2006. Dssim-ontology mapping with uncertainty, In *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006)*, volume 225.

Nagy, M., et al., 2006. Uncertainty handling in the context of ontology mapping for question answering, In *Proceedings of AAAI-2006 Symposium on Semantic Web for Collaborative Knowledge Acquisition*.

Orponen, P., 1990. Dempsters rule of combination is #p-complete, *Artificial Intelligence*, volume 44, pp. 245–253.

Rose, D., 1972. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, *Graph Theory and Computing*, pp. 183–217. Academic Press, New York.

Rose, D., Tarjan, E., 1975. Algorithmic aspects of vertex elimination, In *STOC '75: Proceedings of seventh annual ACM symposium on Theory of computing*, pp. 245–254. ACM.

Sabou, M., et al., 2006. Using the semantic web as background knowledge for ontology mapping, In *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006)*, volume 225.

Shenoy, P., 1989. A valuation-based language for expert systems, *International Journal for Approximate Reasoning*, volume 3, pp. 383–411.

Shenoy, P., Shafer, G., 1988. Axioms for probability and belief function propagation', In *Proceedings of the 4th Annual Conference on Uncertainty in Artificial Intelligence (UAI-88)*, pp. 169–198, New York, NY, Elsevier Science.

Wang, H., et al., 2006. Triangulation of Bayesian networks using an adaptive genetic algorithm, *Foundations of Intelligent Systems*, 4203, 127–136.