Competing Fusion for Bayesian Applications

Michael Abramovici, Manuel Neubach
University of Bochum
Chair of Information Technology in
Mechanical Engineering, Bochum, Germany
{abr,manuel.neubach}@itm.rub.de

Madjid Fathi, Alexander Holland
University of Siegen
Institute of Knowledge Based Systems
Siegen, Germany
{fathi,alex}@informatik.uni-siegen.de

Abstract

In this paper we address and discuss the problem of learning graphical models like Bayesian networks using structure learning algorithms. We present a new parameterized structure learning approach. A competing fusion mechanism to aggregate expert knowledge stored in distributed knowledge bases or probability distributions is also described. Experimental results of a medical case study show that our approach can improve the quality of the learned graphical model.

Keywords: Bayesian Networks, Competing Fusion, Structure Learning.

1 Introduction

Knowledge representation applications need a powerful instrument as formal graphical language requiring reasoning under uncertainty. Bayesian networks represent knowledge under conditions of uncertainty. We introduce a structure learning algorithm for Bayesian networks, when conditional probability information is missing or imperfect. The parameterized approach describes a complete class of algorithms including as special case standard hill climbing. Knowledge fusion is an important field of knowledge science and engineering [1], which can transform and integrate distributed knowledge resources to generate new knowledge representations or visualizations [2]. Experimental results of a medical case study concerning a logical alarm reduction mechanism for intensive care patients show that our approach is very applicable for knowledge fusion to improve the efficiency of working in groups by combining distributed knowledge items.

2 Bayesian Inference and Probabilistic Models

Bayesian networks are graphical models to represent knowledge under conditions of uncertainty. They have been used in many fields like logistic applications [3], expert systems [4] or classification systems as powerful tools for the knowledge representation and inference under uncertainty. Next, we review Bayesian networks and Bayesian inference and proceed to learn such network structures and finally combine network structures built from different experts via competing fusion using sampling techniques and LinOP aggregation.

Bayesian networks and the use of such probabilistic models is based on direct acyclic graphs (DAG) with a probability table associated with each node. The nodes in a Bayesian network represent propositional variables in a domain, the edges between the nodes represent the dependency relationship among the variables. Each node has a conditional probability table (CPT) $P(X \mid X_1,\ldots,X_n)$ attached that quantifies the effects that the parents $X_1,\ldots,X_n$ have on the node. We could say that the conditional probabilities encode the strength of dependencies among the variables. For each variable a conditional probability distribution is defined that specifies the probability of node X being in a certain state given the values of the parents of X.

A decision maker makes decisions by combining his own knowledge, experience and intuition with that available from other sources. Given a learned network structure like Bayesian networks the decision maker can derive

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 378–385
Torremolinos (Málaga), June 22–27, 2008

additional information by applying an inference algorithm. We use the learned Bayesian network to calculate new probabilities when particular information is achieved. For instance let A have n states with $P(A) = (x_1, ..., x_n)$ and assume that we get the information e that A can only be in state i or j. This statement expresses that all states except for i and j are impossible, so next we can illustrate the probability distribution as $P(A, e) = (0,...,0, x_i, 0,...,0, x_j, 0, ...,0)$. Note that $P(e)$ is the sum of $P(a,e)$. Assume a joint probability table $P(U)$ where $\underline{e}$ is the preceding finding (n-dimensional table of zeros and ones). Using the chain rule for a Bayesian network [5] over the universe U and let $\underline{e_i}$ be findings we can express the following

$$P(U,e) = \prod\nolimits_{A\in U} P\big(A|pa(A)\big) \cdot \prod\nolimits_i \underline{e_i} \qquad (1)$$

and for $A \in U$ we have

$$P(A,e) = \frac{\sum_{U\setminus\{A\}} P(U,e)}{P(e)}. \qquad (2)$$

On constructing Bayesian networks from skill data as application area we use nodes to represent database attributes. Different Bayesian network structure learning algorithms have been developed. A good overview demonstrating general approaches to graphical probabilistic model learning from data is introduced by [6], [7] and [8]. In general we can distinguish between search and score methods and the dependency analysis approach.

In the first case the algorithm views the skill learning problem as searching for a structure that best fits the data. The methods starts as graphical representation without any edges, using some search method to add an edge to the representation. In the next step they can use score methods to compare the new with the older structure. The main problem to learn Bayesian networks using search and scoring methods is the NP-hard complexity. Representative algorithms belonging to the search and scoring method are polytree construction algorithms, the K2 algorithm applying a Bayesian scoring method or the Lam-Bacchus algorithm applying the minimal description length principle.

Using the second dependency analysis method is a different approach. These algorithms try to discover the dependencies from the data and next use these dependencies to infer the structure.

Our approach introduced in the following section belongs to the first family of algorithms. Based on the complexity examination we count the number of independent network parameters as

$$|\theta_m| = \sum_i (|X_i| - 1) \cdot |pa(X_i)| \qquad (3)$$

with $|pa(X_i)|$ as number of (joint) states of all parent nodes of X_i to obtain a complexity boundary as model complexity.

3 Learning Bayesian Network Structures

The revelation of conditional independence relationships plays a fundamental role within the medical diagnostic and information processing. When for instance learning network structures from skill data, one applies information theoretic measures to detect conditional independence relations and afterwards uses the well known d-separation concept [9] to infer the structures of networks [10]. Measure the volume of the information flow between two nodes to conclude if a group of valves corresponding to a condition set can reduce and eventually block the information flow.

In Bayesian networks information can be determined about the value of a node knowing the value of the other node when both nodes are dependent. The mutual information between two nodes can therefore provide information in the case of two nodes dependency. The degree of their relationship is also important. The mutual information of two nodes X_i and X_j is defined as

$$Inf(X_i, X_j) = \sum_{x_i,x_j} P(x_i, x_j) log_2 \frac{P(x_i, x_j)}{P(x_i)P(x_j)} \qquad (4)$$

and the conditional mutual information is defined as

$$Inf(X_i, X_j|C) = \sum_{x_i,x_j,c} P(x_i, x_j, c) log_2 \frac{P(x_i, x_j|c)}{P(x_i|c)P(x_j|c)}. \qquad (5)$$

Based on model complexity and evaluation measures we built up parameterized scoring functions as follows:

$$f_1(m,D) = \Big(\sum_i \sum_{x_i \in dom(X_i)} \sum_{Y_j \in pa(X_i)} \sum_{y_j \in dom(Y_j)} P\big(X_i = x_i, Y_j = y_j\big) log_2 \frac{P\big(X_i = x_i, Y_j = y_j\big)}{P(X_i = x_i) \cdot P\big(Y_j = y_j\big)}\Big) - \beta \cdot \sum_i (|X_i| - 1) \cdot \prod_{X \epsilon pa(x_i)} |X| \,. \quad (6)$$

Formula (6) takes the following form applying maximum likelihood:

$$f_2(m,D) = \Big(N \cdot \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} log_2 \frac{N_{ijk}}{N_{ij}}\Big) - \beta \cdot |\theta_m| \quad (7)$$

where m represents the network structure, D the training data, N the total number of cases, $r_i (1 \le i \le n)$ the cardinality of the random variable X_i, $q_i (1 \le i \le n)$ the number of joint states of all parent nodes of X_i and $|\theta_m|$ stands for the number of independent network parameters, expressed also as $|\theta_m| = \sum_{i=1}^{n} (r_i - 1) \cdot q_i$. In special cases regarding the scoring function well known quality measures can be present for specific β allocations like $\beta=1$ (AIC metric) or $\beta = \frac{1}{2} log_2 N$ (minimum description length metric). Different additional quality measure instances are discussed in detail in [11] and [12].

3.1 Searching Strategies

In several well known searching problems we have the property that the state description itself contains all the information needed to find a solution. The path by which the solution is reached is in this case irrelevant. The calculation of the optimal Bayesian network structure includes browsing through the complete network search space. The naive approach is to measure for any search order each graph structure and give back the best valued DAG. The main problem is the huge number of different directed acyclic graphs, which depends on the node number n:

$$g(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} \,. \quad (8)$$

The following table 1 gives an impression about the growing number of directed acyclic graphs depending on the number of network nodes n. If n exceeds the value 7 machine learning applications need further searching strategies like heuristic strategies.

Table 1: Number of directed acyclic graphs $g(n)$ depending on the number of nodes n

n	5	6	7	8	10
$g(n)$	29281	$3.78 \cdot 10^6$	$2.46 \cdot 10^8$	$7.84 \cdot 10^{11}$	$4.18 \cdot 10^{18}$

Iterative improvement algorithms often provide the most practical approach. Consider all states laid out on the surface of a landscape. The height of any point on the landscape corresponds to the evaluation function of the state at that point. The idea is to move around the landscape by trying to find the highest peaks which stand for the optimal solutions [5].

Hill Climbing algorithms always try to make changes that improve the current state. They continually move into the direction of increasing values. Hill Climbing does not maintain a search tree, so the node data structure need only record the state and its evaluation by value. When there is more than one best successor to choose from, the algorithm can select among them via random.

Instead of starting randomly in the case of stuck in a local minimum, we could also allow the search to take some downhill steps to escape the local minimum. This is the idea of simulated annealing. Instead of picking the best move, it picks a random move. If the move actually improves the situation, it is always executed. Otherwise, the algorithm makes the move with some probability degree less than one. The probability decreases exponentially with the badness value of the move. A second parameter T is also used to determine the probability. At higher values of T, bad moves are more likely to be allowed. As T tends to zero, they become more and more unlikely, until the simulated annealing algorithm behaves more or less like Hill Climbing. Hill Climbing search contains useful optimization potential evaluated in the following subsection.

3.2 Learning Structures Using LAGD Hill Climbing

Based on appropriate scoring functions like (6) or (7) a local search strategy using greedy hill climbing can be executed to compare the directed acyclic graphs m_{old} and m_{new} using $\Delta = f(m_{new}) - f(m_{old})$. Look ahead in good directions (LAGD) hill climbing as new generalization approach calculates in advance k

steps in regard to the chosen scoring function. Another parameter l stands for the number of best evaluated network structures per look ahead step. LAGD Hill Climbing offers a new class of parameterized algorithms including the parameters:

- number of look ahead steps k
- number of calculated good operations l per each look ahead step

The LAGD hill climbing algorithm using local search can be expressed in pseudo code as follows:

1. Start the algorithm using an arbitrary DAG m_{old}.
2. Calculate the difference value Δ well-defined as $\Delta = f(m_{new}) - f(m_{old})$ between the current DAG candidate m_{old} and all its neighbours m_{new}. Store the l best valued operations in a specific set B_1.
3. $\forall b \in B_1$: Execute operation b and calculate according to 2. a specific set B_2 with the l best operations based on the current network structure. Continue step 3 until reaching look ahead depth k and choose on the last recursion level the operation greedy.
4. Choose a transition sequence to maximize Δ based on all considered sequences and set m_{old} as one of the DAGs with the greatest score-win rate in regard to Δ.
5. GOTO 2. UNTIL $\Delta \leq 0$ for all allowed transition sequences to graph structures in a distance of k steps.
6. Deliver the outcoming DAG m_{old}.

As testbed for the implementation we used the WEKA environment (*Waikato Environment for Knowledge Analysis),* a collection of machine learning algorithms for data mining and knowledge discovery tasks [17]. We were able to integrate our developed LAGD hill climbing in the Waikato Environment and today LAGD is an integral part of WEKA. The LAGD Hill Climbing algorithm spans a whole class of structure learning algorithms parameterized by the number of good operations *l*, the number of look ahead steps *k*, the maximal number of parents per node, the score type as instance of quality measures like bayes, entropy, akaike information criterion or minimum description length, the initiation possibility as naive bayes classifier and the final application of a markov blanket correction (compare figure 1).

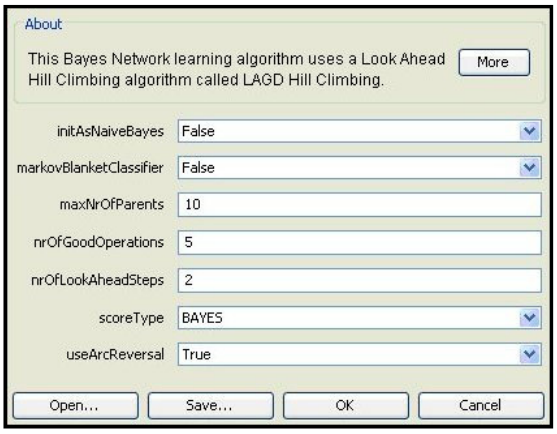

Figure 1: LAGD spans a whole class of structure learning algorithms

The special case *k=1* (nrOfLookAheadSteps=1) results in standard greedy hill climbing. To reduce the calculating time adjust the parameter l to regard only the l best valued operations per look ahead step. The lower bound concerning the network structure quality agrees with standard greedy hill climbing results based on greedy k-step operation sequences. The computational complexity per iteration step (k-step sequence) with n Bayesian network nodes, k look ahead steps and l good operations per look ahead step can be determined as $O(\sum_{i=0}^{k-1} l^i \cdot n^2)$. This term may be assessed by $O(l^{k-1} \cdot n^2)$, as can be seen by the following induction:

We want to show that $\sum_{i=0}^{k-1} l^i \cdot n^2 \leq 2 \cdot l^{k-1} \cdot n^2$.

Initial induction step *k=1*:

$\sum_{i=0}^{1-1} l^i \cdot n^2 = n^2 \leq 2 \cdot l^{1-1} \cdot n^2 = 2 \cdot n^2$ (true)

Induction step $k \rightarrow k+1$:

$$\sum_{i=0}^{(k+1)-1} l^i \cdot n^2 = \left(\sum_{i=0}^{k-1} l^i \cdot n^2\right) + l^k \cdot n^2 \underset{IH}{\leq} 2 \cdot l^{k-1} \cdot n^2 + l^k \cdot n^2 \underset{l \geq 2}{\leq} l \cdot l^{k-1} \cdot n^2 + l^k \cdot n^2 = 2 \cdot l^k \cdot n^2 = 2 \cdot l^{(k+1)-1} \cdot n^2$$

Obviously $\sum_{i=0}^{k-1} l^i \cdot n^2$ can be assessed by $2 \cdot l^{k-1} \cdot n^2$. The argument follows when neglecting the multiplier *2* according to Bachmann-Landau notation.

3.3 Experimental Results for Medical Data Using LAGD Hill Climbing

We have chosen a medical test dataset to compare our LAGD Hill Climbing algorithm with other classical well known algorithms like simulated annealing or standard greedy hill climbing used for different metrics as instances of quality measures (i.e. AIC, MDL).

The ALARM network [13] is a commonly used network which is a representative of a real life Bayesian network. It was originally described by Beinlich as a network for monitoring patients in intensive care. The ALARM network structure consists of 37 nodes and 46 edges with 8 diagnosis, 16 medical findings and 13 temporary variables.

A Monte Carlo technique named *Probabilistic Logic Sampling* was used to generate case databases consisting of 10.000 test cases based on the ALARM network. Probabilistic Logic Sampling generates each case or sample by orienting to the weak node order induced by the underlying directed acyclic graph [14].

The criterion to compare the quality or performance of different Bayesian networks learned according to different structure learning algorithms is the so called LogScore, which is based on the appropriate local score metric. When applying for instance the akaike information criterion (AIC with $\beta=1$) the LogScore AIC derives as follows:

$$LogScore_{AIC}(m,D) = -h_2(m,D) = \left(N \cdot \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} log_2 \frac{N_{ijk}}{N_{ij}} \right) - \beta \cdot |\theta_m|. \quad (9)$$

I. e. we derive the LogScore AIC by simply multiplying the AIC metric by minus one. It follows from formula (9) the equation $LogScore_{AIC}(m,D) = -h_2(m,D) = f_2(m,D)$.

Figure 2 illustrates the learning curve for the dataset ALARM using the parameter values *maxNrOfParents=5*, *nrOfLookAheadSteps=2* and *nrOfGoodOperations=5* with LogScore AIC ($\beta=1$).

The quotient $\frac{N_{ijk}}{N_{ij}}$ is always in the interval (0,1], i. e. the resulting logarithm of this quotient is obviously less than or equal to zero.

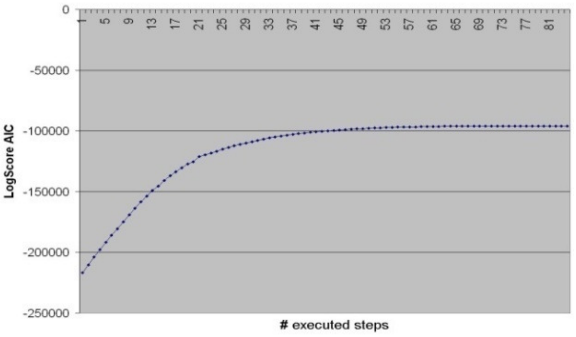

Figure 2: ALARM LAGD Hill Climbing learning curve with LogScore AIC

The LogScore starts strong negative and rises successive in finding better valued neighbour graphs. With increasing the number of iteration steps the model-complexity function reduces the total LogScore by the summand $\beta \cdot |\theta_m|$. Our experimental results evaluate different structure learning algorithms like simulated annealing, greedy hill climbing and LAGD based on LogScore metrics Bayes, AIC and MDL (see figure 3).

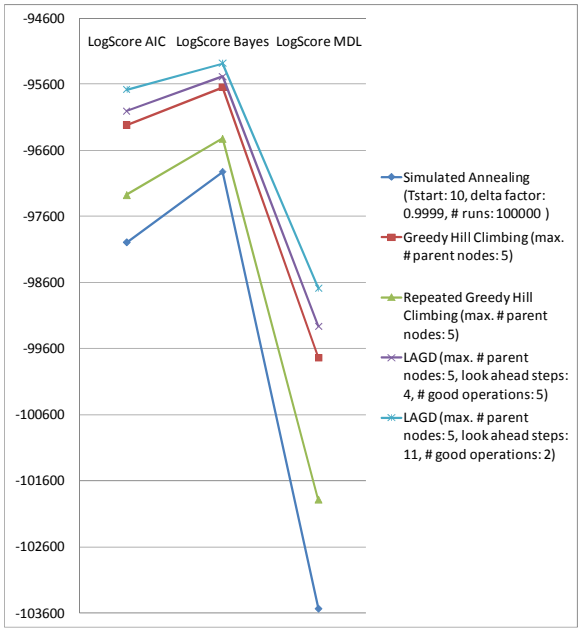

Figure 3: Structure learning results comparing different Bayesian network structure learning algorithms with LogScore metrics AIC, Bayes and MDL

Figure 4 displays in detail the dependency between the quality of the calculated network structure and the number of look ahead steps with fixed parameter value *l=2*. We have chosen the minimum description length as local score metric for an exemplary comparison of greedy hill climbing and LAGD since results were very similar using the other metrics AIC and Bayes.

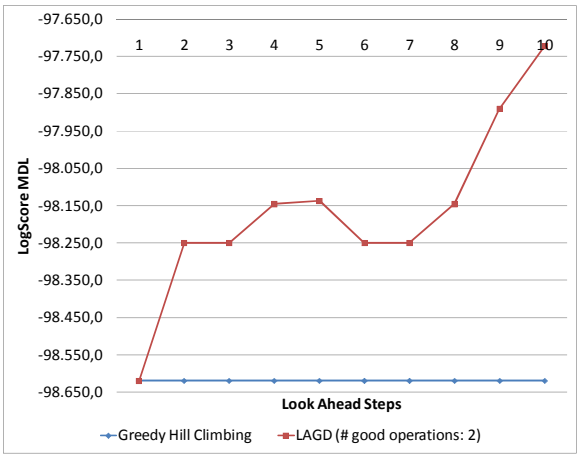

Figure 4: Comparison of greedy Hill Climbing and LAGD with varied look ahead steps using LogScore MDL

It is obvious that increasing LAGD look ahead steps causes a better network quality. The startling bending down of the curve when increasing the look ahead parameter from *5* to *6* may be explained as follows: With look ahead parameter $k=5$ LAGD finds a good local optimum, while with parameter $k=6$ LAGD falls in a trap, when seeing a better evaluated graph structure in the beginning. LAGD follows this "false" path, that finally results in a local optimum, which is unfortunately worse than in the case $k=5$.

4 Competing Fusion of Distributed Knowledge

The development of knowledge-based systems involves knowledge acquisition from a diversity of sources often geographically distributed. It is obviously difficult to bring together information from different knowledge sources about a subject of common interest. The sources include for instance written documents, interviews and application data stored in distributed knowledge bases disposed from different experts often specialized in a medical field like intensive care or a specialist in internal medicine.

Intuitively merging medical knowledge bases is to find a knowledge base that has at least as much information as each component medical knowledge base and is the smallest such medical knowledge base. Different experts working together are not in a position to generate the complete Bayesian network structure including conditional probability tables and all necessary random variables. An expert E_i (i=1,…,n) working in an specific hospital division has only access to specific medical data or medical knowledge to build a substructure of a complete Bayesian network structure. The main problem occurs when all included experts compose their individual medical knowledge to build up the Bayesian network structure representing the domain knowledge. We can integrate knowledge stored in different Bayesian networks BN_i through knowledge fusion. Researchers differentiate the aspects competing, complementary and cooperative knowledge fusion.

To merge different Bayesian networks (for example BN_1 and BN_2) we must first determine a measure for the approximation quality. A suitable measure is the Kullback-Leibler divergence between Bayesian network structures like BN_1 and BN_2. The Kullback-Leibler divergence [15] expresses the difference or distance between two probability distributions. Given the probability distributions p and q define $KL(p,q)$ as follows:

$$KL(p,q) = \sum_x p(x) log \frac{p(x)}{q(x)}. \qquad (10)$$

The Kullback-Leibler divergence is obviously not symmetric and can also be verbalised using the cross entropy $H(p,q)$ as follows:

$$\begin{aligned} KL(p,q) = & -\sum_x p(x) log\ q(x) \\ & + \sum_x p(x) log\ p(x) \\ & = H(p,q) - H(p). \end{aligned} \qquad (11)$$

The KL divergence values are not negative with $KL(p,q) = 0$ if and only if $p=q$.

Competing fusion includes combined expert knowledge from different fields like medical, financial or engineering problem scenarios. Each expert can generate a case database with embedded and not obvious inferable conditional probability table settings and network structures. The Bayesian network learning algorithm introduced in section 3 delivers Bayesian networks for each expert to fuse via sampling or via LinOP aggregation.

Competing fusion via sampling:

1. Synthesize for each expert network a case database using a Monte Carlo technique.
2. Aggregate the expert case databases.
3. Learn the aggregated Bayesian network structure based on the case database determined in 2. using an automatic learning algorithm.

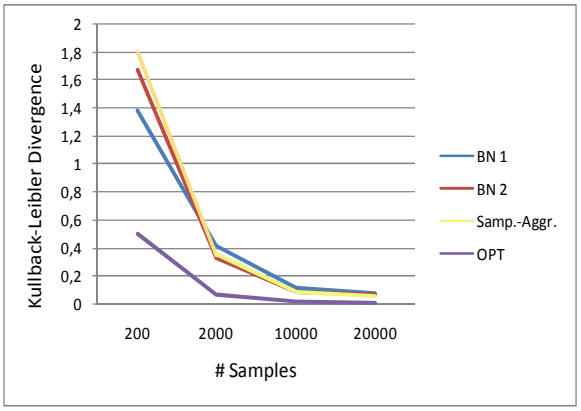

Figure 5: Sampling aggregation results with Kullback-Leibler divergence

The ALARM network [13] was splitted in two disjoint case databases based on a Monte Carlo technique named Probabilistic Logic Sampling. The respective Bayesian networks BN_1 and BN_2 were learned with LAGD. According to the described sampling algorithm the Bayesian network *Sampl.-Aggr.* was generated. The number of samples was varied between 200 and 20000. Results are illustrated in figure 5.

The use of Monte Carlo techniques and the hereby induced noise may be avoided by applying an aggregation operator for a common unified probability distribution. The aggregation of L expert probability distributions $p_1,\ldots,p_L$ using the LinOP operator [16] leads to the following algorithm:

1. Aggregate the probability distributions $p_1,\ldots,p_L$ of L Bayesian networks using the LinOP operator for a common probability distribution

$$p^* = \sum_{i=1}^{L} \frac{\alpha_i}{\sum_{j=1}^{L} \alpha_j} p_i, \qquad (12)$$

with LinOP

$$LinOP\ (\alpha_1, p_1, \ldots, \alpha_L, p_L) = \sum_{i=1}^{L} \alpha_i p_i. \qquad (13)$$

2. Learn the aggregated Bayesian network with a local score metric based learning algorithm using p^*.

The ALARM network results using competing fusion via LinOP aggregation demonstrates the following figure 6 [18]:

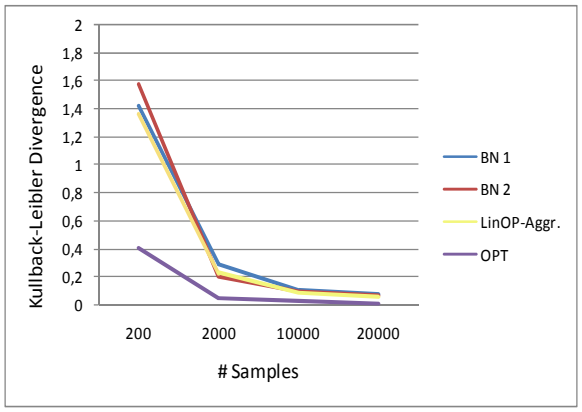

Figure 6: LinOP aggregation results with Kullback-Leibler divergence

When comparing sampling aggregation to LinOP aggregation in the case of small sample sizes (less than 2000) the results show that LinOP aggregation clearly outperforms sampling aggregation. Due to the introduced noise the aggregated network *Samp.-Aggr.* has an even higher Kullback-Leibler divergence compared to the original Bayesian networks BN_1 and BN_2. Increasing successively sample sizes causes in both cases an asymptotical alignment of Kullback-Leibler divergence of the aggregated network structure to the optimum.

5 Summary and Future Work

We presented a new local score metrics based structure learning approach called LAGD Hill Climbing. The number of look ahead steps and the number of operations considered for look ahead are configurable, which spans a whole class of structure learning algorithms. Both the time taken for computing and the quality of the calculated Bayesian network structure may be tuned by adjusting these parameters.

The test result for the ALARM network using quality measures like LogScore AIC, Bayes and MDL clarifies the advantages of LAGD in regard to the achievable network quality. LAGD is fully integrated in the Waikato Environment for Knowledge Analysis and may be downloaded via the WEKA website. In the future, the usage of a generalized structure

learning approach including dynamic interdependencies between look ahead steps and number of good operations is intended.

LAGD hill climbing may be applied in the field of mechanical engineering and especially product lifecyle management to learn from existing condition monitoring data and reveal relationships between sensor data, environmental parameters and failure events. In this context competing fusion improves the quality of aggregated knowledge models based on distributed knowledge sources. Aggregation and fusion methods like sampling aggregation and LinOP aggregation make the generated condition monitoring knowledge collected in the product use phase from various customers within a feedback cycle usable for product development and help to improve the next generation of a given product, which is topic of another paper [19].

References

[1] L. Ru-qian (2003). Knowledge Science and Computation Science. Tsighua University Press, Beijing, China.

[2] A.D. Preece, K.Y. Hui (2000). The Kraft Architecture for Knowledge Fusion and Transformation. In *Knowledge Based Systems,* volume 13, pages 113-120.

[3] A. Holland (2004). A Bayesian Approach to Model Uncertainty in Network Balanced Score Cards. In *Proceedings of the 10th International Conference on Soft Computing (Mendel Conference)*, Brno, Czech Republic, pages 134-138.

[4] G.D. Kleiter (1992). Bayesian Diagnosis in Expert Systems. In *Artificial Intelligence*, volume 54, pages 1-32.

[5] S. Russel, P. Norvig (2003). Artificial Intelligence. A Modern Approach. Prentice Hall, New Jersey, USA.

[6] D. Heckerman (1995). A Tutorial on Learning Bayesian Networks. In *Technical Report, MSR-TR-95-06*, Microsoft Research, USA.

[7] P. Krause (1996). Learning Probabilistic Networks. In *Technical Report, Philips Research Laboratories*, United Kingdom.

[8] C. Borgelt, R. Kruse (2002). Graphical Models. Methods for Data Analysis and Mining. John Wiley & Sons, New York, USA.

[9] J. Pearl (1988). Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Francisco, USA.

[10] F.V. Jensen (2001). Bayesian Networks and Decision Graphs. Statistics for Engineering and Information Science, Springer-Verlag, Berlin Heidelberg New York.

[11] C. Borgelt (2000). Data Mining with Graphical Models. Dissertation, University of Magdeburg, Germany.

[12] G. Grimmett, D. Stirzaker (2004). Probability and Random Processes. 3rd Edition, Oxford University Press, UK.

[13] I.A. Beinlich, H.J. Suermondt, R.M. Chavez, G.F. Chooper (1989). The ALARM Monitoring System. A Case Study with two Probabilistic Inference Techniques for Belief Networks. In *Proceedings of the Second European Conference on Artificial Intelligence in Medical Care*, pages 247-256.

[14] M. Henrion (1988). Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. In *Uncertainty in Artificial Intelligence 2*, pages 149-163, North-Holland.

[15] L.C. van der Gaag, S. Renooij (2001). On the evaluation of Probabilistic Networks. In *Proceedings of the 8th Conference on Artificial Intelligence in Medicine*, Lecture Notes in Computer Science, volume 2101, pages 457-461, Springer-Verlag, Berlin Heidelberg New York.

[16] M. Pradhan (1997). Focussing Attention in Anytime Decision Making. PhD thesis, Section on Medical Informatics, Stanford University, USA.

[17] I.H. Witten, E. Frank (2005). Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann, San Francisco.

[18] P. Maynard-Reid II, U. Chajewska (2001). Aggregating Learned Probabilistic Beliefs. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pages 354-361.

[19] M. Abramovici, M. Fathi, A. Holland, M. Neubach (2008). Integration of Product Use Information into PLM. In *Proceedings of the 15th CIRP International Conference on Life Cycle Engineering (LCE 2008)*, Sydney, Australia.