REVIEW OF COMPUTER ENGINEERING STUDIES
Vol.2, No.1, 2015, pp.1-8
http://dx.doi.org/10.18280/rces.020101

AIRCRAFT FAULT DIAGNOSIS BY SOLVING MAP EXACTLY

Wang Yao*and Sun Qin

School of Aeronautics, Northwestern Polytechnic University, P.R.China.

Email: wangyaorose@126.com

ABSTRACT

Bayesian networks (BNs) used for intelligence fault diagnosis have gotten lots of attention. This paper focuses on the application of maximum a posteriori hypothesis (MAP) in aircraft fault diagnosis. Combining with the aircraft system its own characteristics and technical background in the field of diagnosis, the paper both presents a systematic scheme used for solving MAP exactly together with the needed pseudo-code and devises a complete flow block diagram to identify the most possible cause of a fault by solving MAP exactly. Here MAP is solved by performing a binary depth-first search in a search tree on the basis of upper bounds on exact MAP solution. With the new scheme and flow block diagram, one can solve MAP problem systematically which helps people to decide the maintenance strategy.

Keywords: Bayes network, Fault diagnosis, MAP, Binary depth-first search, Upper bounds.

1. INTRODUCTION

Airplane consists of large-scale integrated control systems with a great amount of mechanical, electrical and hydraulic components; among them there exist complicated physical interactions as well as themselves internal. The physical relations between the components are complex, coupling and full of uncertainties [1, 2]. Apparently, the probability analysis of such uncertainty should be theoretical basis and key means in such complex system fault system diagnosis.

Bayesian network is a combination of probability theory and graph theory and it is a theory system that describes the relationships of random variables [3, 4, 5]. Based on Bayesian theory, BN is an effective way to get an accurate solution to uncertain probability reasoning [6, 7, 8]. In fault diagnosis, given the fault representation, a quantitative probability evaluation method is needed to predict the probability of the every failure mode of the each specific component. After that, by comparing the maximum probability of each component, maintenance strategy is given. That is just called MAP (maximum a posteriori hypothesis) in Bayer theory.

MAP appears to be much more difficult in probabilistic inference than other two typical problems, posterior probability (Pr) and most probable explanation (MPE). The problems for the latter two are NP-complete while MAP problem is NP^{PP} -hard [9~13]. Specifically, standard structure-based methods for finding exact solutions to MAP, such as variable elimination and jointree algorithm, have complexities that are exponential in the constrained treewidth of the network [14, 15]. Nevertheless, these methods can solve MPE and Pr in time which is exponential in treewidth [14, 15].

A recent algorithm proposed by James D. Park [15] stands for a notable advance of the state of the art in solving MAP exactly. It performs a depth-first search in the space of all instantiations of the MAP variables. The search can be effectively pruned using upper bounds that can be calculated by jointree algorithm only exponential in the treewidth, not limited by the constrained treewidth. However, for a BN, there are different jointrees according to different constructing methods [16, 17, 18]. However, James D. Park [15] just provides a general idea how to compute upper bound using a jointree neither provides an algorithm for constructing a concrete jointree nor gives an algorithm for computing upper bound.

The focus of this paper is on providing a complete scheme for solving MAP exactly, which consists of three main algorithm, one for constructing an efficient binary jointree, one used for computing upper bound and the last used for finding out the exact MAP solution by performing a binary depth-first search. Specifically, these three algorithms stem from fusion algorithm [18], CPT algorithm [5] and systematic algorithm [15], separately. For there exit gaps among these three algorithms raised by different authors, the paper not only modifies these three algorithms to become a systematic scheme for solving MAP but also provides the necessary pseudo-codes in detail. On the basis of the scheme, a complete flow block diagram for fault diagnosis is devised. Finally, two cases in the aircraft fault diagnose setting are analyzed. In this paper, all the variables in BN are binary in view of that any BN with multi-valued variables can be transformed in to a BN where all the variables are binary.

The paper is structured as follows. Section 2 is dedicated to provide the systematic scheme for solving MAP exactly, including studying the complexity of MAP and presenting three modified algorithms and their corresponding pseudo-code. Section 3 provides a complete flow block diagram for fault diagnose based on all fixed algorithms above. Section 4 discusses two airplane fault diagnose cases using the flow block diagram raised in Section 3. Section 5 closes with conclusions.

2. THE SCHEME FOR SOLVING MAP EXACTLY

2.1 Complexity of solving MAP

This section starts with a formal notation of MAP. Given a Bayesian network with conditional probability tables (CPTs), all the variables can be partitioned into three sets: $\boldsymbol{E}$, $\boldsymbol{S}$, $\boldsymbol{M}$. $\boldsymbol{E}$ stands for the set of variables whose values are known, $\boldsymbol{S}$ stands for the set which should be summed out, and $\boldsymbol{M}$ stands for the set of variables which should be maximized over. The MAP problem is that of finding and instantiation $\boldsymbol{m}$ of variables $\boldsymbol{M}$ which maximizes the probability of instantiation $\boldsymbol{e}$ [15].

Let Φ denotes the set of CPTs, and for each CPT $\phi \in \Phi$, let ϕ_e denote its restriction under the evidence $\boldsymbol{e}$. $\Pr(\boldsymbol{m}, \boldsymbol{e})$ for all $\boldsymbol{m}$ can computed by the following potential ψ over variables $\boldsymbol{M}$:

$$\psi = \sum_{\boldsymbol{S}} \prod_{\phi \in \Phi} \phi_e \qquad (1)$$

Hence, MAP problem can be solved by the following equation:

$$\psi^* = \max_{\boldsymbol{M}} \sum_{\boldsymbol{S}} \prod_{\phi \in \Phi} \phi_e \qquad (2)$$

Note that when $\boldsymbol{M}$ is empty, MAP problem becomes Pr problem while when $\boldsymbol{S}$ is empty, it turns into MPE problem. Both MPE and Pr are NP-complete while MAP is NP^{pp}-complete, which means that, in variable elimination, the elimination order of MPE and Pr are unconstrained while the elimination order of MAP is constrained. Constrained elimination order expresses that one can only choose among orders that put the MAP variables last to solve MAP problem.

The following theorem further illustrates the need for a constrained order for solving MAP problem [14].

Theorem 1: Let ϕ be a potential over disjoint variable X, Y, Z. Then,

1 $\sum_X \sum_Y \phi = \sum_Y \sum_X \phi$

2 $\max_X \max_Y \phi = \max_Y \max_X \phi$

3 $[\sum_X \max_Y \phi](z) \geq [\max_Y \sum_X \phi](z)$.

For all instantiations z of variables Z. Moreover, the equality holds only when there is some value y of variable Y such that $\max_Y \phi_X = \phi_{XY}$ for all values x of variable X. That is, the optimal value of variable Y is independent of variable X.

The first and second equations in Theorem 1 guarantee that using any elimination order, correct solution to Pr problem and MPE problem can be calculated. Nevertheless, the third inequality conveys that only the elimination order chosen from those that put MAP variables last can ensure the correct MAP solution. Moreover, Theorem 2 explains that an arbitrary elimination order can guarantee an upper bound (here upper bound means no less than) on the correct MAP solution, which is significant in finding out the exact MAP solution using a depth-first search.

Theorem 2: In Eq. (2), for any two subsets $\tilde{\boldsymbol{M}}$ $\tilde{\boldsymbol{S}}$ that satisfy $\tilde{\boldsymbol{M}} \subseteq \boldsymbol{M}$ and $\tilde{\boldsymbol{S}} \subseteq \boldsymbol{S}$, the following formula can arrive at a probability, which is an upper bound on the exact MAP solution.

$$\psi^* = \max_{\boldsymbol{M} \setminus \tilde{\boldsymbol{M}}} \sum_{\boldsymbol{S} \setminus \tilde{\boldsymbol{S}}} \max_{\tilde{\boldsymbol{M}}} \sum_{\tilde{\boldsymbol{S}}} \prod_{\phi \in \Phi} \phi_e \qquad (3)$$

Theorem 2 can be derived from Theorem 1 easily and the paper omits the formal proof. In the following, the paper uses MAP ($\boldsymbol{M}$, $\boldsymbol{e}$) to represent the exact MAP solution and uses BD ($\boldsymbol{M}$, $\boldsymbol{e}$) to represent the upper bound which can be computed by Theorem 2.

2.2 Constructing a binary jointree

Section 2.1 discusses the complexity of MAP, which is more difficult than Pr and MPE problem from the view of elimination order. To perform efficient computation, an undirected network called jointree is extracted from BN, which contain clusters and each cluster consists of several nodes from the original BN. The information of locally connected clusters, provided through CPT, is propagated in the jointree by message passing mechanism [19, 20].To increase the computational efficiency, a special kind of jointree named binary jointree is constructed, in which every node is connected no more than three neighbors. Binary jointree has many advantages compared with other kinds of jointree [18]. This part provides Algorithm 1 for constructing a binary jointree which is modified from fusion algorithm originally proposed by P. P. Shenoy [18].

The data declaration of Algorithm 1:

Λ : A set that contains all the variables of the BN.

Γ : A set that is a union set of the following two kinds of variables sets, which should be presented in the binary jointree.

The first kind is the ones that can denote the CPTs of each variable in BN. For example, for node X with its fathers $\pi(X)$, $\Pr(X \mid \pi(X))$ is the CPT of node X and set $\{X, \pi(X)\}$ is among this kind of sets.

The second kind is the ones whose probability distribution needs to be calculated. In the aircraft diagnosis setting of the paper, each variable in a MAP query forms such a set. For instance, if the MAP query is set $\{X, Y\}$, set $\{X\}$ and set $\{Y\}$ form the second kind of sets.

$\boldsymbol{N}$: A set that contains the nodes of the binary jointree. It is initially null.

$\boldsymbol{\varepsilon}$: A set that contains the edges of the binary jointree. It is initially null.

π : An order of set Λ. It can be gotten through some heuristic algorithm.

$|\Gamma_Y|$: The number of the elements in set Γ_Y.

2.3 Computing upper bounds using jointree algorithm

Jointree is a data structure that can calculate marginalization probability using Shenoy-Shafer local elimination algorithm with high efficiency [21, 22]. It works by the means of message passing mechanism. This part will explain two characters of jointree, how to initialize a jointree and message passing mechanism first. Then, combining with Section 2.1, it expounds upper bounds in detail. Finally, this part modifies CPT algorithm [5] to make it able to calculate upper bounds.

Two characters of Jointree: As a jointree, it comprises many clusters in geometry and each cluster is made up of a set of variables of BN. A jointree has two characters [5, 14].

No.1 A jointree must satisfy connectedness. In other words, if in a jointree, cluster A and cluster B both contain variable V, all the clusters on the roads that connect to A and B must contain V too.

No.2 A jointree must be able to cover the BN. Here, "cover" has two means. Firstly, for any variable V in BN, there exits at least one cluster that satisfies $X \in C$ and $\pi(X) \subseteq C$, in which $\pi(X)$ stands for the set of father nodes of X, and such a cluster is called family cluster. Secondly, in the jointree, the union set of all the clusters is just the complete set of all the variables in BN.

Algorithm 1 construct $(\Lambda, \Gamma, \boldsymbol{N}, \boldsymbol{\varepsilon}, \pi)$

```
1: while $|\Gamma|>1$ do
2: Choose a variable $Y$, $Y \in \Lambda$
3: $\Gamma_Y=\{\gamma_i \in \Gamma \mid Y \in \gamma_i\}$
4: while $|\Gamma_Y|>1$ do
5: choose $\gamma_i \in \Gamma_Y$ $\gamma_j \in \Gamma_Y$ ,and $|\gamma_i \cup \gamma_j| \le |\gamma_m \cup \gamma_n|$ for all $\gamma_m$, $\gamma_n \in \Gamma_Y$
6: $\gamma_k = \gamma_i \cup \gamma_j$
7: $\boldsymbol{N} = \boldsymbol{N} \cup \{\gamma_i\} \cup \{\gamma_j\} \cup \{\gamma_k\}$
8: $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon} \cup \{\{\gamma_i, \gamma_k\}, \{\gamma_j, \gamma_k\}\}$
9: $\Gamma_Y = \Gamma_Y - \{\gamma_i, \gamma_k\}$
10: $\Gamma_Y = \Gamma_Y \cup \{\gamma_k\}$
11: end while
12: if $|\Lambda|>1$ then
13: choose $\gamma_i$ ,and $\gamma_i = \Gamma_Y$
14: $\gamma_j = \gamma_i - \{Y\}$
15: $\boldsymbol{N} = \boldsymbol{N} \cup \{\gamma_i\} \cup \{\gamma_j\}$
16: $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon} \cup \{\{\gamma_i, \gamma_j\}\}$
17: $\Gamma = \Gamma \cup \{\gamma_j\}$
18: end if
19: $\Gamma = \Gamma - \{\gamma_i \in \Gamma \mid Y \in \gamma_j\}$
20: $\Lambda = \Lambda - \{Y\}$
21: end while
```

Initializing a jointree. After a jointree is extracted from BN, it will have to be initialized next. Specifically, each CPT of the original BN must be assigned to some cluster. That is, for every variable X in a BN, one should find a family cluster of variable X, then assigns $P(X \mid \pi(X))$ to the cluster. After assigning all the CPTs of the BN, the left clusters which still is not assigned a CPT to should be assigned "unit function=1" to.

Message passing mechanism: Message passing mechanism must follow two basic rules [14].

Rule 1: Before Cluster *r* deliver messages to its neighbor Cluster named Cluster *s*, Cluster *r* must receive messages from its own neighbor cluster(s) first.

Rule 2: When Cluster *r* will receive messages from neighbor Cluster s (assume Cluster *s* has received messages from its own neighbor(s)), the messages should be marginalized first. Here, marginalization means marginalizing off variables that are included in Cluster s and not included in Cluster *r*.

According two rules above, the corresponding mathematic expression is given as below.

Given a BN with three sets ***E***=***e***, ***S*** and ***M***, it has been extracted into a Jointree using any constructing algorithm. Function ϕ_i is a CPT of the original BN that is assigned to cluster *i* of the constructed Jointree now.

The process of message passing from cluster *i* to its neighbor cluster *j* is expressed as following:

$$M_{ij} = \max_{\boldsymbol{X}} \sum_{\boldsymbol{Y}} \phi_i \prod_{k \ne j} M_{ki} \tag{4}$$

In which, $\boldsymbol{X} \subseteq M, \boldsymbol{Y} \subseteq S$, all variables in ***X*** and ***Y*** only appear in cluster *i* do not appear in cluster *j* and M_{ki} represents the messages that Cluster *j* has received from its own neighbor cluster, cluster *k* [23].

Equation (2) has two special cases. When ***X*** is empty, the message passing mechanism leads to Pr(*e*). When ***Y*** is empty, the message passing mechanism can solve MPE. However, neither ***X*** nor ***Y*** is empty, the message passing mechanism leads to an incorrect MAP solution. Explanations are as below.

Algorithm 2-1 inward BD1 $(\mathrm{J}, \boldsymbol{E}, \boldsymbol{M}, \boldsymbol{e}, \boldsymbol{S})$

```
1: clear all the potential information stored in each edge of the tree, and set $\boldsymbol{E}=\boldsymbol{e}$;
2: choose a cluster in the tree which contains some variable in set $\boldsymbol{M}$ as the root;
3: for (every neighbor cluster $C_{ni}$ of $C_r$)
4: call collect1 $(\mathrm{J}, C_r, C_{ni})$ and return a function $\psi_i$;
5: end for
6: $h(C_r^{'}) = \prod_i^m \psi_i \bullet P$ , m is the number of the neighbor clusters of $C_r$ , and P is the CPT stored in $C_r$ , $C_r^{'} = C_r \setminus \boldsymbol{E}$;
7: return $BD(\boldsymbol{M}, \boldsymbol{e}) = \max_{C_r^{'} \setminus S} \sum_{C_r^{'} \setminus \boldsymbol{M}} h(C_r^{'})$.
```

In essential, performing inference by the means of message passing mechanism is a kind of variable elimination [5]. Such a conclusion also can be arrived at by comparing Eq. (3) and Eq. (4). When one chooses a root cluster in a Jointree, the elimination order is decided. When $\boldsymbol{X}$ is empty, the first equation in Theorem 1 guarantees that summation commuting with summation cannot change the result. Accordingly, the message passing mechanism corresponding to a special kind of variable elimination process during which only there are variables that should be summed out has no effect on the final result. This is similar to the condition that when $\boldsymbol{Y}$ is empty. Obviously, the third inequality guarantees the exact MAP solution can only be calculated using the order that MAP variables come last.

Algorithm 2-2 outwardBD1(J, $\boldsymbol{E}$, $\boldsymbol{M}$, $\boldsymbol{e}$, $\boldsymbol{S}$, X)

```
1: choose a cluster $C_r$ that only has a variable $X$;
2: for (every neighbor cluster $C_{ni}$ of $C_r$ )
3: call collect1 $(J, C_r, C_{ni})$ and return a function $\psi_i$ ;
4: end for
5: $h(X_1) = \prod_i^m \psi_i \bullet P$ , m is the number of the neighbor clusters of $C_r$ , and P is the CPT stored in $C_r$ ;
6: return BD($\boldsymbol{M}\backslash X$,$\boldsymbol{e}$ $x$)=h($X$=$x$)
```

Algorithm 2-3 collect1(J, C^r, C^n)

```
1: $\psi$ =Retrieve $(C_n, C_r)$ ;
2: if ( $\psi \neq null$ )
3: return $\psi$ ;
4: set $\boldsymbol{Z}_1 = C_n \backslash (C_r \cup \boldsymbol{E}_1) Z_1$ and $P_1$ is the CPT stored in $C_n$ ;
5: if ( $C_n$ is a leaf cluster in J)
6: $\psi = \max_{\boldsymbol{Z}_1 \backslash \boldsymbol{S}_1} \sum_{\boldsymbol{Z}_1 \backslash \boldsymbol{M}_1} P_1$ ;
7: else
8: let $C'_{n1} C'_{n2} \cdots C'_{nk}$ be the neighbor clusters of $C_n$ , except $C_r$ ;
9: for i= 1to $k$
10: $g_i$ =collect1 $(J, C_r, C'_{ni})$ ;
11: end for
12: $\psi = \max_{\boldsymbol{Z}_1 \backslash \boldsymbol{S}_1} \sum_{\boldsymbol{Z}_1 \backslash \boldsymbol{M}_1} P_1 \prod_{j=1}^{k} g_i$ ;
13: Save $(C_n, C_r, \psi)$ ;
14: end if
15: return $\psi$
```

CPT Algorithm [5] is a Jointree algorithm used for solving $\Pr(\boldsymbol{e})$, summation and multiplication operations are involved. After modifying CPT Algorithm, this part presents a new algorithm named BD algorithm for computing upper bounds, in which the message passing mechanism involves summation, maximization and multiplication operations. BD algorithm is shown as Algorithm 2, including three sub-algorithms, Algorithm 2-1, Algorithm 2-2 and Algorithm 2-1, which is of great importance in binary systematic search algorithm for solving MAP exactly, which will be discussed in Section 2.4.

2.4 Solving MAP using a binary depth-first search

As MAP is a discrete optimization problem, the upper bound on the probability of MAP solution, which can be calculated using Algorithm 2, is the basis of the depth-first search algorithm for finding out the exact MAP solution [15].This section is divided into three parts. The first part will give a brief introduction of a search tree. The second part will state the general idea of how to perform a search in a search tree. The final part will modify the original search algorithm, forming a new algorithm used in a BN where all the variables are binary, named binary depth-first search.

A brief introduction of a search tree. To illustrate the complete search procedure clearly, this section takes a search tree for example, which is shown in Figure 1. The search tree corresponds to a BN which has three MAP variables, X1, X2 and X3. In the search tree, the root node N corresponds to empty instantiation. The children of a node which represents instantiations $\boldsymbol{x}$, where $\boldsymbol{X} \subseteq \boldsymbol{M}$, are nodes which represents instantiation $\boldsymbol{x}v$ for some variable V in $\boldsymbol{M} \backslash \boldsymbol{X}$. Leaves of the search tree correspond to different instantiations $\mathbf{m}$ of MAP variables $\boldsymbol{M}$. The subscript of the donation N_*^* represents a subset of MAP variables $\mathbf{M}$ and the superscript represents the corresponding subset of instantiation $\boldsymbol{m}$. Since each node in the search tree corresponds to a variable instantiation**,** different superscript identifies different node of the search tree.

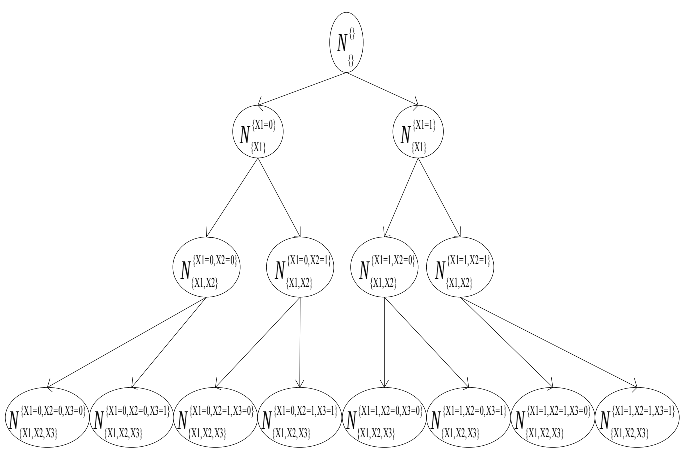

$N_{\{\}}^{\{\}}$
$N_{\{X1\}}^{\{X1=0\}}$
$N_{\{X1\}}^{\{X1=1\}}$
$N_{\{X1,X2\}}^{\{X1=0,X2=0\}}$
$N_{\{X1,X2\}}^{\{X1=0,X2=1\}}$
$N_{\{X1,X2\}}^{\{X1=1,X2=0\}}$
$N_{\{X1,X2\}}^{\{X1=1,X2=1\}}$
$N_{\{X1,X2,X3\}}^{\{X1=0,X2=0,X3=0\}}$
$N_{\{X1,X2,X3\}}^{\{X1=0,X2=0,X3=1\}}$
$N_{\{X1,X2,X3\}}^{\{X1=0,X2=1,X3=0\}}$
$N_{\{X1,X2,X3\}}^{\{X1=0,X2=1,X3=1\}}$
$N_{\{X1,X2,X3\}}^{\{X1=1,X2=0,X3=0\}}$
$N_{\{X1,X2,X3\}}^{\{X1=1,X2=0,X3=1\}}$
$N_{\{X1,X2,X3\}}^{\{X1=1,X2=1,X3=0\}}$
$N_{\{X1,X2,X3\}}^{\{X1=1,X2=1,X3=1\}}$

Figure 1. A search tree corresponding to a BN, in which the MAP variables are X1, X2 and X3.

The discussion above is about the structure of a search tree. This part will assign a value to each node next. Specifically, each leaf node is associated with probability Pr(***m***, ***e***). An internal node ***x*** is associated with upper bound BD(***M***/***X***, ***ex***). For the internal node ***x***, the best leaf node below node **x** is just the exact MAP (***M***/***X***,***ex***) solution.

The basic idea of systematic search algorithm. The basic idea of systematic search algorithm is to perform a depth-first search on the tree, while computing an upper bound BD(***M***/***X***, ***ex***) at each internal node ***x*** [5]. If the upper bound is no more than the value of the best leaf node encountered so far, according to Theorem 2, the children of node ***x*** should not be explored because the nodes below node ***x*** cannot lead to a leaf node with higher value than the best leaf node encountered.

A new modified algorithm. The systematic search algorithm proposed by Park can be applied to a BN where the variables can have many states. Focusing on the characters of binary variables, the part will modify the algorithm more feasible for binary variables.

Assume the process of searching has been at node ***x*** in the search tree, and variable V has been chosen to instantiate next, where $V \in \boldsymbol{M} \setminus X$. Suppose further that when *v*=0, $B_{v=0} = BD(\boldsymbol{M} \setminus (\boldsymbol{X} \bigcup \{V\}), ex\{v=0\}) \leq bScore$ is established, where *bScore* is the best leaf node encountered so far. According to the basic idea of systematic search algorithm, instantiation ***xv***=0 should pruned and only instantiation ***xv***=1 can lead to the exact MAP(***M***, ***e***) with the deepening of the search. It is similar to the case when *v*=1.On the basis of the analysis above, this part factors this pruning information into the original systematic algorithm, forming a new binary Join tree algorithm provided as Algorithm 3.

```
Algorithm 3: Search(X, z, bSol, bScore, J, S, Z )
1: B ⟵ BD(X,z)
2: if B≤bScore  then return
3: if X==null, then bScore ⟵ B, bSol ⟵ z, return
4: for each $V$, $V \in \boldsymbol{X}$ ,do
5: compute B0=BD(X\V,zv=0) and B1= BD(X\V,zv=1)
6:   if B0≤bScore and B1≥bScore,,then   z=z ∪ v=1,
Z=Z ∪ V, X=X\V;
7:   if B1≤bScore and B0≥bScore ,then   z=z ∪ v=0,
Z=Z ∪ V, X=X\V;
8: if B1≤bScore and B0≤bScore,, then  return ;
9: else
10: compute $M_v$ and $T_v$
11: end for
12: chose a variable V as the leaf node, where $V \in \boldsymbol{X}$ and
$V = \max_{V \in \boldsymbol{X}} (M_v / T_v)$
13: call Search(X\V, zv=0, bSol, bSore, J,S,Z)
14: call Search(X\V, zv=1, bSol, bSore, J,S,Z)
```

3. A COMPLETE FLOW BLOCK DIAGRAM FOR FAULT DIAGNOSE

In earlier sections, the article has discussed MAP problem theoretically and provides the pseudo-code in detail. This section will analyzed the application of MAP in the setting of fault diagnose and design a complete flow block diagram used in software systematically.

3.1 Application of MAP in fault diagnose

This part starts with an example (see Figure 2) to elaborate the application of MAP in the setting of diagnosis. Figure 2 shows a BN which can be viewed as a causal influence diagram of an engineering physical system with some failure mode. In the figure, nodes labeled with the same letter of an alphabet compose a whole component. For instance, four nodes A1, A2, A3 and A4 compose the whole component A. For a step further, nodes labeled with the same letter of an alphabet but with different numbers represent different physical effect factors of the same component. Still taking component A as an example, in component A, four physical factors have an effect on I1 of the system. Every node has two states, or call instantiations, 0 and 1, representing the physical factor behaves normal or abnormal respectively. Clearly, component A has 16 different instantiations and each instantiation corresponds to a probability.

From the illustration above, each physical factor, instead of a whole component of a real system, corresponds to a node in a BN. The interplay of each physical factor forms the dependence relationship of the BN. Probability theory describes this dependence relationship in quantity and thus forms Conditional Probability Table (CPT), such as the CPT P(D1|A1, A2) assigned to node D1 describes the relationship between node D1 and its father nodes *A*1 and A2.

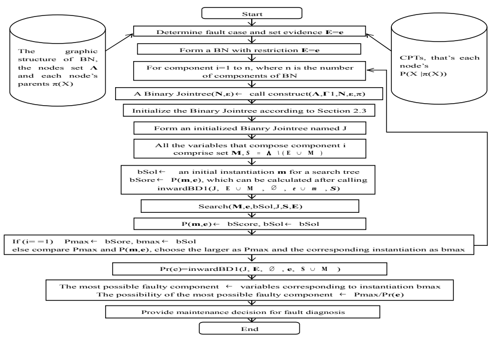
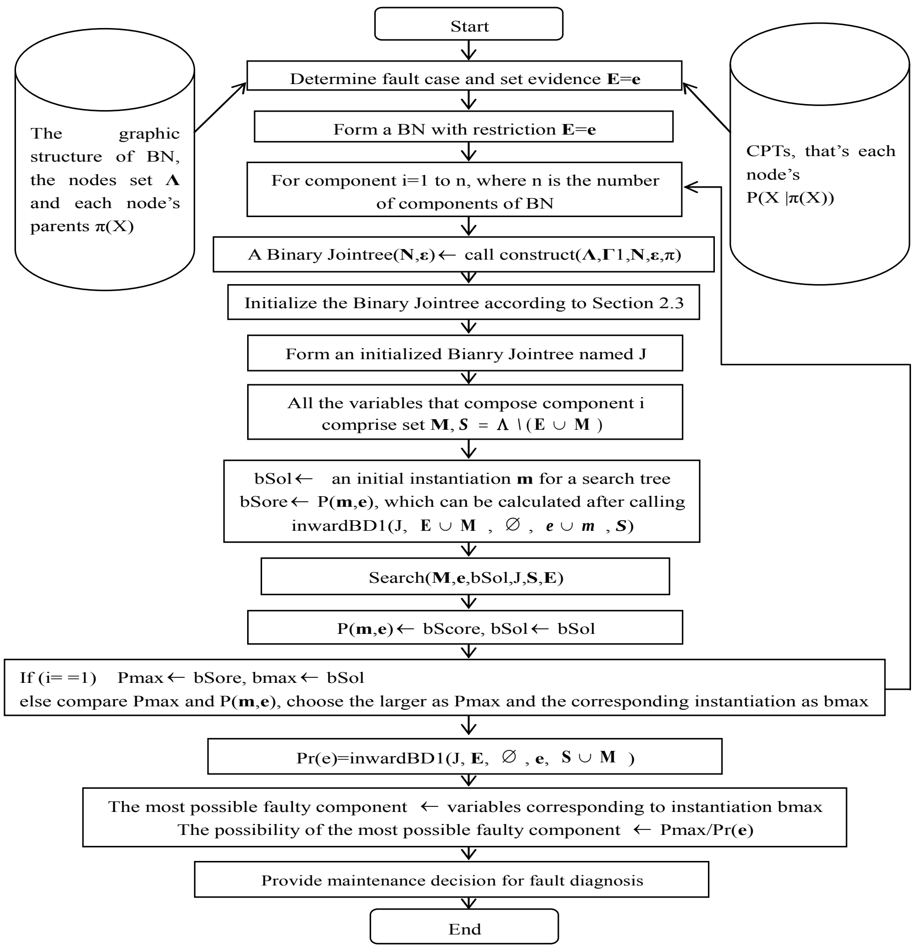

Start
Determine fault case and set evidence **E**=**e**
The graphic structure of BN, the nodes set **Λ** and each node's parents π(X)
CPTs, that's each node's P(X |π(X))
Form a BN with restriction **E**=**e**
For component i=1 to n, where n is the number of components of BN
A Binary Jointree(**N**,**ε**)← call construct(**Λ**,**Γ**1,**N**,**ε**,π)
Initialize the Binary Jointree according to Section 2.3
Form an initialized Bianry Jointree named J
All the variables that compose component i comprise set **M**, *S* = **Λ** \ (E ∪ M)
bSol← an initial instantiation **m** for a search tree
bSore← P(**m**,**e**), which can be calculated after calling inwardBD1(J, E ∪ M , ∅ , *e* ∪ *m* , ***S***)
Search(**M**,**e**,bSol,J,**S**,**E**)
P(**m**,**e**)← bScore, bSol← bSol
If (i==1) Pmax← bSore, bmax← bSol
else compare Pmax and P(**m**,**e**), choose the larger as Pmax and the corresponding instantiation as bmax
Pr(e)=inwardBD1(J, **E**, ∅ , **e**, S ∪ M)
The most possible faulty component ← variables corresponding to instantiation bmax
The possibility of the most possible faulty component ← Pmax/Pr(**e**)
Provide maintenance decision for fault diagnosis
End

Figure 2. A bayesian network used for fault diagnosis

Given the fault evidence and failure information of a system, one may always be interested both in identifying which component caused such a failure in a maximum probability and wandering to know the corresponding fault mode—the instantiation of that failure component. That's to say, each component composes a MAP query. After answering all the MAP queries, one needs to compare each component's MAP result in order to decide the most possible failure component and its corresponding failure mode of the system.



Start
Determine fault case and set evidence E=e
Form a BN with restriction E=e
For component i=1 to n, where n is the number of components of BN
A Binary Jointree(N,ε)← call construct(Λ,Γ1,N,ε,π)
Initialize the Binary Jointree according to Section 2.3
Form an initialized Bianry Jointree named J
All the variables that compose component i comprise set M, S = Λ \ (E ∪ M)
bSol← an initial instantiation m for a search tree
bSore← P(m,e), which can be calculated after calling inwardBD1(J, E ∪ M , ∅ , e ∪ m , S)
Search(M,e,bSol,J,S,E)
P(m,e)← bScore, bSol← bSol
If (i= =1) Pmax← bSore, bmax← bSol
else compare Pmax and P(m,e), choose the larger as Pmax and the corresponding instantiation as bmax
Pr(e)=inwardBD1(J, E, ∅ , e, S ∪ M)
The most possible faulty component ← variables corresponding to instantiation bmax
The possibility of the most possible faulty component ← Pmax/Pr(e)
Provide maintenance decision for fault diagnosis
End
The graphic structure of BN, the nodes set Λ and each node's parents π(X)
CPTs, that's each node's P(X |π(X))

Figure 3. Complete flow block diagram used in the software to identify the most fault reason

According to the above elaboration, apparently, in Figure 2, seven components may lead to the system's failure with the fault evidence I1=1. To find out the most possible failure component and its failure mode, first one should do the cycle i=1 to 7, computing $\text{MAP}(\boldsymbol{M}_i, \boldsymbol{e})$ to identify component i's failure mode in the greatest possibilities. Here, $\boldsymbol{M}_i$ is the set representing MAP variables of component i. After that, compare the seven MAP results and the component with the maximum probability and its corresponding fault mode is the final result in fault diagnose.

3.2 A complete block diagram

Combining with engineering practice, this section provides a block diagram used for fault diagnose, seeing Figure 3, all sub-functions of which has been introduced before. The block diagram is used to develop a software, one software intend to solve MAP problems in fault diagnosis setting and such a software is applied to the cases illustrated in section 4. The paper centers on MAP problems and the details of the software are omitted.

4. CASE STUDIES

This section gives two fault cases of a certain type of aircraft hydraulic power supply system and adopts the software raised in Section 3 to solve MAP to find out the most possible reasons responsible for the two fault modes, separately. Next is the detailed illustration about the two cases.

4.1 The first fault case

Physically speaking, this first fault case is that the fire cut-off valve of 1# hydraulic system fails (abbreviation: HYD1 SOV FAIL, encode data 291-111-41). The corresponding BN structure of this fault mode is shown in Figure 4 and the below describes the physical meaning of each node in the BN, which also is assigned a encode data to. The format of encode data of a node xx-xx-xx-xx.x consists of nine letters, the former six representing the component and the latter three representing the concrete physical factor of a component. For example, 29-K3-07 stands for circuit; 29-K3-07-00.1 stands for circuit break; 29-K3-07-00.1 stands for short circuit.

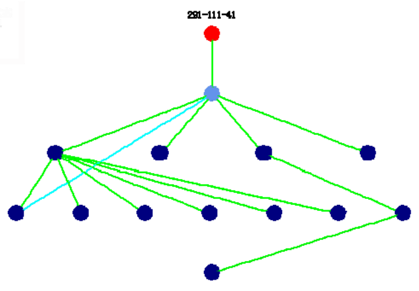


Figure 4. The BN structure of the first fault case

The number of nodes expect for the top evidence node: 13

The number of components: 8

The description of each physical factor and their corresponding encode data are as follows:

29-K3-07-00.1: circuit break

29-K3-07-00.2: short circuit

29-11-20-00.1: firewall shutoff valve unable to shut off

29-11-20-00.2: firewall shutoff valve shut off too slowly

29-11-20-00.3: firewall shutoff valve shut off with no instructions

29-11-20-00.5: firewall shutoff valve has an external leakage and hydraulic oil loss

29-K3-08-00.1: electric equipment failure

29-33-02-00.2: temperature switch failure, internal short-circuit, breaking point drifting, low

29-11-27-00.3: too much oil bypasses the heater exchanger and the system temperature increases

29-K3-03-00.1: hydraulic control logic circuit (HCLE) has a logic calculation error

29-K3-06-00.1: full authority digital engine control (FADEC) has a data error

29-11-29-00.4: check valve obstruction leads to shell scavenge pipe plugging and the system overheats

29-11-20-00.D: firewall shutoff valve doesn't work or works abnormally

By using the software raised in Section 3, the calculation result is given in Table 1. The table only lists the components whose fault probability is more than 0.1 and their corresponding fault modes. The table is listed from high to low. Value 1 and 0 denote abnormal and normal respectively.

It can be seen from the table that the physical factor 29-K3-07-00.2 of component 29-K3-07 most probably leads to the fault case. Referring to the encode data of each node and the corresponding physical meaning, it can be concluded that circuit short leads to the fire cut-off valve of the 1# hydraulic system fails. In practice, circuit short indeed always is the reason responsible for the system failure.

Table 1. Results of the first fault mode

29-K3-07		29-11-20		29-K3-08	
Pmax=0.9918		Pmax=0.9916		Pmax=0.4961	
factors	states	factors	states	factors	states
29-K3-07-00.1	0	29-11-20-00.1	0	29-K3-08-00.1	1
29-K3-07-00.2	1	29-11-20-00.2	0		
		29-11-20-00.3	1		
		29-11-20-00.5	0		

4.2 The second fault case

The second fault case is that the AC motor-driven pump of 1# hydraulic system fails (abbreviation: HYD1ACMP FAIL, encode data: 291-121-41). The failure case is modeled in BN shown in Figure 5. The format of encode data of a node xx-xx-xx-xx.x is similar to case 1 and there are four auxiliary nodes in the BN, which don't stand for any real physical factors.

The number of nodes expect for the top evidence node: 23

The number of components: 14

The description of each physical factor and their corresponding encode data are as follows:

29-K3-05-00.1: circuit breaker open

29-K3-05-00.2: circuit breaker short-cut

29-K3-07-00.1: circuit break

29-K3-07-00.2: short circuit

29-K3-08-00.1: electric equipment failure

29-11-02-00.4: alternating current (AC) electric pump has a serious external leakage and loses the hydraulic oil

29-11-02-00.5: AC electric pump has a slight external leakage and the hydraulic oil is exhausted

29-11-02-00.2: AC electric pump outputs with high pressure, resulting in high temperature

29-11-02-00.3: AC electric pump has a serious internal leakage and the system overheats

29-11-02-00.1: AC electric pump output failure

29-11-02-00.6: AC electric pump outputs with low-flow

29-33-02-00.2: temperature switch failure, internal short-cut, breaking point drifting, low

29-31-03-00.2: EDP pressure switch fails and always shows pressure high

29-31-03-00.1: EDP pressure switch fails and always shows pressure low

29-11-03-00.1: the cushion of EDP unable to isolate the vibration

29-K3-03-00.1: HCLE has a logic calculation error

29-K3-03-00.D: HCLE sends wrong instruction

29-K3-06-00.1: (FADEC) unit has a miscalculation

29-11-10-00.5: oil sample of oil filter is polluted and by-pass valve has an open failure

29-11-06-1.1: output of EDP blocked

29-11-02-00.D: AC electric pump cannot work

29-K3-05-00.D: circuit breaker doesn't work

29-K3-07-00.D: circuit failure

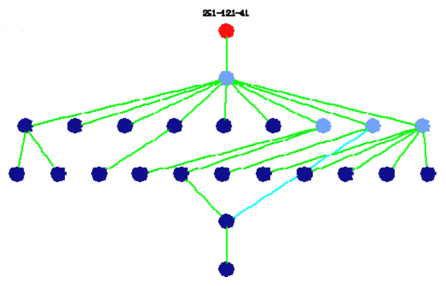


Figure 5. The BN structure of the second fault case

By using the software raised in Section 3, the calculation result is shown in Table 2. The rules of the result given in this table are the same with Table 1.

Table 2. Results of the second fault case

29-K3-05		29-K3-07		29-K3-08	
Pmax=0.9885		Pmax=0.6593		Pmax=0.3298	
factors	states	factors	states	factors	states
29-K3-05-00.1	0	29-K3-07-00.1	0	29-K3-08-00.1	1
29-K3-05-00.2	1	29-K3-07-00.2	1		

It can be seen from the table that the physical factor 29-K3-05-00.2 of component 29-K3-05 most probably leads to the fault case. Referring to the encode data of each node and the corresponding physical meaning, it can be concluded that circuit short leads to the AC motor-driven pump of 1#

hydraulic system fails, which corresponds to the engineering practice.

5. CONCLUSIONS

MAP solution is a useful measure in the fault diagnosis of large and complicated system, such as systems and subsystems in aircraft. This paper carries on a thorough research on MAP problem. In particular, a systematic scheme for solving MAP problems is raised. Moreover, a complete flow block diagram to identify the most possible cause of a fault by solving MAP exactly is presented. The work in this paper mainly is reflected in three aspects.

1. The paper modifies the fusion algorithm, CPT algorithm and systematic algorithm to form a systematic scheme used for solving MAP problems and all the key corresponding pseudo-code is presented. The systematic scheme is a methodology that can solve MAP without being restrained by restrained treewidth.
2. Based on the systematic scheme used for solving MAP problems, the paper also devises a complete flow block diagram for fault diagnose, forming a technical method systematically in engineering application. Such a diagram ensures that all these algorithms in the paper can be used in diagnostic practice.
3. According to the complete flow block diagram for fault diagnosis and all the algorithms raised in the paper, software intend to solve MAP problems in fault diagnosis setting is developed and such a software is preliminarily applied to a hydraulic power supply system fault diagnosis of a certain type of aircraft. The result shows the software works well and corresponds to the engineering practice.

ACKNOWLEDGMENT

The research was supported by quality and reliability fundamental project of China's Ministry of Industry and Information Technology's Twelfth Five-year Plan.

REFERENCES

1. O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, S. Uckun, Diagnosing Faults in Electrical Power Systems of Spacecraft and Aircraft, *Proc. The 20th Innovative Applications of Artificial Intelligence Conference*, pp.1699-1705, 2008.
2. O. J. Mengshoel, M. Chavira, K. Cascio, et al, Probabilistic Model-Based Diagnosis: An Electrical Power System Case Study, *IEEE SMC part A*, vol. 40, pp. 874-885, 2010.
3. A. Darwiche, Bayesian Networks, *Communications of the ACM,* vol. 53, pp. 80-90, 2010.
4. F. V. Jensen, An Introduction to Bayesian Networks, Spring Verlag, New York, 1996.
5. N. L. Zhang, H. P. Guo, An Introduction to Bayesian Networks, Science press, Beijing, 2006(in Chinese).
6. B. D'Ambrosio, Inference in Bayesian Networks, *Artificial Intelligence,* vol. 20, pp. 21-36, 1999.
7. C. Huang, A. Darwiche, Inference in Belief Networks: A Procedure Guide, *International Journal of Approximate Reasoning,* vol. 15, pp. 225-263, 1996.
8. N. L. Zhang, D. Poole, A Simple Approach to Bayesian Network Computations. *Proc. The 10th Canadian Conference on Artificial Intelligence*, pp.171-178, 1994.
9. A. M. Abdelbar, S. M. Hedetniemi, Approximating Maps for Belief Network is NP-Hard and Other Theorems, *Artificial Intelligence*, vol. 102, pp. 21-38, 1998.
10. E. Charniak, S. E. Shimony, Cost-Based Abduction and MAP Explanation, *Artificial Intelligence*, vol.66, pp.345-474, 1994.
11. E. Shimony, Finding MAPs for Belief Networks is NP-hard, *Artificial Intelligence*, vol.68, pp.399-410, 1994.
12. G. F. Cooper, The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, vol. 42, pp. 393-405, 1990.
13. J. D. Park, MAP Complexity Results and Approximation Methods, *Proc. The 18th Conference on Uncertainty in Artificial Intelligence*, pp.388-396, 2002.
14. A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, New York, 2012.
15. J. D. Park, A. Darwiche, Solving MAP Exactly Using Systematic Search, *Proc. Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pp. 459-468, 2003.
16. A. Darwiche, A Differential Approach to Inference in Bayesian Networks, *Journal of ACM*, vol.50, pp. 280-305, 2003.
17. J. Pearl, Fusion, Propagation and Structuring in Belief Networks, *Artificial Intelligence*, vol.29, pp.241-288, 1986.
18. P. P. Shenoy, Binary Join Trees for Computing Marginals in the Shenoy-Shafer Architecture, *International Journal of Approximate Reasoning*, vol. 17, pp. 239-263, 1997.
19. G. R. Shafer, P. P. Shenoy, Probability Propagation, *Annals of Mathematics and Artificial Intelligence*, vol. 2, pp.327-351, 1990.
20. P. P. Shenoy, G. Shafer, Propagating Belief Functions with Local Computations, *IEEE Expert*, vol.1, pp.43-52, 1986.
21. T. Schmidt, P. P. Shenoy, Some Improvements to the Shenoy-Shafer and Hugin Architectures for Computing Marginal, *Artificial Intelligence* , vol. 102, pp. 323-333, 1998.
22. N. L. Zhang, Computational Properties of Two Exact Algorithms for Bayesian Networks, *Applied Intelligence*, vol. 9, pp. 173-183, 1998.
23. K. Lingasubramanian, S. M. Alam, S. Bhanja, Maximum Error Modeling for Fault-Tolerant Computation Using Maximum a Posteriori (MAP) Hypothesis, *Microelectronics Reliability*, vol.51, pp.485-501, 2011.