文章编号:1001-2486(2007)04-0130-05

基于大规模贝叶斯网络的安全性分析算法*

董豆豆,周经伦,赵 炤,周忠宝
(国防科技大学 信息系统与管理学院,湖南 长沙 410073)

摘 要:贝叶斯网络计算量随着节点数增多呈指数增长,限制了大规模贝叶斯网络在安全性分析中的应用。为此,利用独立性条件分解整个网络,压缩推理时显式表达的项数,给出了计算顶事件发生概率及割集的算法,并分析了算法复杂性。在满足工程需要情况下,将提出算法与基于BDD算法相比,该算法表现出占用内存少、运行速度快的良好性能。

关键词:贝叶斯网络;安全性分析;割集;条件独立
中图分类号:O213.2 **文献标识码**:A

Safety Analysis Algorithm Based on Large Scale Bayesian Networks

DONG Dou-dou, Zhou Jing-lun, ZHAO Zhao, ZHOU Zhong-bao
(College of Information System and Management, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The computation consumption of Bayesian network increases exponentially with the nodes number and that limits the application of large-scale Bayesian network. In order to relieve this situation, this paper makes use of the independence condition to decompose the whole Bayesian network according to the characteristics of safety analysis, compresses the items number in reasoning process, proposes the algorithms for top event probability and cut sets, simultaneously analyses the computational complexity. Compared with BDD-based FTA algorithm, the proposed algorithms showed a lower memory demand and a higher speed performance when meeting the need of safety engineering.

Key words: bayesian network; safety analysis; cut set; conditional independence

籍其良好特性,贝叶斯网络(Bayesian network, BN)在安全性与可靠性领域的应用研究不断广泛和深入。Sankaran等人利用BN对复杂系统中顺序失效和相关失效进行描述[1],提高了安全性分析能力;同时,还研究了在获得实际数据条件下,通过BN的证据传播特性来检验安全性模型的有效性[2]。Bobbio等人研究了将故障树(Fault Tree, FT)映射成BN的方法,并利用BN对依靠系统进行可靠性分析[3]。目前,我们也采用BN对模糊情形下系统进行安全性分析,提出置信割集的概念,并研究了求最小强割集的算法,以使建模更接近现实系统。

然而,BN推理复杂性却限制了大规模BN在安全性分析中的应用。BN推理是利用网络中蕴涵的条件独立性,计算待求概率值的过程。目前推理算法主要有4类[4]:多树传播算法(Polytree Propagation)、团树传播算法(Clique Tree Propagation)、图约简算法(Graph Reduction)、组合优化算法(Combinatorial Optimization)。以上推理算法各有特点,但都没有摆脱显示求和的计算方式,其计算量随着节点数的增多呈指数增长[5-6]。另外,割集(Cut Sets, CS)是安全性分析的重要内容,在BN中关于求割集的探讨仍很少提及。因此,有必要对基于贝叶斯网络的安全性分析算法进行研究,以促进贝叶斯网络在安全性分析中的良好应用。

1 安全性分析时的BN

BN是由变量(节点)和有向线组成的有向无环图(Directed Acyclic Graph, DAG)。用 $N = <<X, E$

* 收稿日期:2007-03-08
基金项目:国家部委基金资助项目(2005AA845023)
作者简介:董豆豆(1976—),男,博士生。

$>$,$P>$表示,其中,$<X,E>$是有向无环图的节点 X 和有向边 E 的集合,X 是节点的集合,E 是表示节点间因果关系的有向线集合,而集合 P 则是关于 X 的条件概率表(Conditional Probabilistic Table, CPT)的集合[7]。用 $X=\{\bullet_1,\bullet_2,\cdots,\bullet_i,\cdots,\bullet_n\}$表示 BN 的节点集合。

安全性分析的内容主要包括在根结点为已知证据时求顶事件发生的概率及割集[8]。设 BN 只有一个顶结点(多顶结点时,可针对各个顶结点利用所提出的算法进行计算),在求顶事件概率时,BN 的根节点所在的状态概率信息是已知的证据。假设 BN 中每个节点都是两态的,即只处于正常和故障两种状态。对于集合 $C(C\subseteq R)$,用 $C=1$ 表示 C 中每一个元素 $r_i=1(r_i\in C)$,用 $C=0$ 表示 C 中每一个元素 $r_i=0(r_i\in C)$。

前面已指出,BN 推理时计算量随着节点数的增多呈指数增长,显然,在推理时若能有效消去节点,则可以减少一次性显式表达的项数以及降低处理 BN 的规模。下面将利用 BN 网中存在大量的独立性,压缩其规模,提出求顶事件发生概率和割集的算法。

2 基于独立性分解求顶事件发生概率的算法

定义 1 前驱集(Predecessor Set):节点 X_i 的所有父辈结点的集合称为前驱集,记为 Predec(X_i)。

若记 X_i 的父结点集为 $Parent(X_i)$,显然有 $Parent(X_i)\subseteq$Predec(X_i)。同时,若 Predec(X_i)$\cap$Predec(X_j)$=\Phi$,则可以知 $\varepsilon(\varepsilon=\Phi)d$ 分离 X_i,X_j,由定义 1 不难得到推论 1。

推论 1 对于 BN 的节点 X_i,X_j,其前驱集分别记为 Predec(X_i),Predex(X_j),如果 Predec(X_i)$\cap$Predec(X_j)$=\Phi$,则有 $I(X_i,X_j)$。其中,$I(X_i,X_j|\varepsilon)$表示 X_i 和 V_j 条件独立于ε,ε、X_i、X_j 可以是集合。

定义 2 有向会聚树(Directed Converging Tree) 对于树 T_d,其顶结点为 T,根结点集合为 R,T 的前驱集记为 Predec(T),如果 $R\subseteq$Predec(T),则称 T_d 为有向会聚树。

求顶事件 $P(T=1)$的概率时需要得到其显式表达式,而利用节点间的独立性,此过程可分为多步进行,因此,采用分治法的算法设计思想[9],可减少一次性显式表达项数的数量。可将求顶事件发生概率由分为三步:(1)对整个 BN 依据独立性进行压缩,将之压缩为一个有向会聚树;(2)利用传统 BN 推理方法计算出非独立部分的顶结点事件概率;(3)计算出有向会聚树中顶事件发生的概率,即为 BN 顶事件发生的概率。

算法目的:基于独立性分解求顶事件发生概率的算法。

输入:各个根节点事件发生概率;输出:顶节点发生概率 ProbofTop。

方法:

```
1) CBNNode * pCurNode = GetTopOfBN(); //得到 BN 的顶节点
2) IndepenceCompress(pCurNode) //利用独立性对整个贝叶斯网络进行压缩
3) CBNNode * pCurNode = GetLeftRootOfBN(); //得到 BN 的左根节点
4) While (pCurNode - > m _ pFatherNode! = NULL){
5)     pcurNode = curNode - > m _ pFather //将父节点设置为当前节点
6)     pCurNode - > m _ dbProb = ComputerCurrentNodeProb(pCurNode)
7)                                   //将计算得到的概率赋给当前节点
8)     If (pCurNode - > m _ pFatherNode = = NULL) then //如果当前结点为顶结点
9)       ProbofTop = pCurNode - > m _ dbProb //得到顶事件的发生概率
10)      return ProbofTop; //返回顶事件发生概率
11)      endIf}
Procedure IndepenceCompress(CNode * pNode) //利用独立性条件压缩整个贝叶斯网络
1) nodelist = pNode - > GetParentSet(); //得到 pNode 结点的父结点集合
2) For each node in nodelist {
3) If (IsNodeIndepence(node,nodelist) = = FALSE) //node 节点是否独立于 nodelist
                                   //中其他节点,不独立返回 FALSE
```

```
4) ComputerNodeProbInTaditional (pCurNode) //用传统方法计算当前结点事件概率
5) Else
6)      IndepenceCompress(& node) //递归调用
7) endif|
Procedure double ComputerCurrentNodeProb(CBNNode * pCurNode)
1)                                          //根据输入节点计算当前节点的事件概率
2) CBNNodeList childList = pCurNode - > GetChildList(pCurNode) //得到当前节点的子
3)                                                              //节点链表
4) For each node in childList |
5)   If (node.m _ dbProb = = NULL) then //当节点事件概率不存在时
6)       node - > m _ dbProb = ComputerCurrentNodeProb(&mode) //递归调用
7)   endIf
8)   If (node = = childList.LastNode()) then //节点是否为最后一个节点
9)       ProbOfcurNode = pCurNode - > ComputerCurNode() //根据条件概率表计算
10)                                                     //出当前节点概率
11)       return ProbOfcurNode; //返回当前节点的概率
12)  endIf |
Procedure Bool IsNodeIndepence(curnode,nodelist) //判断节点 curnode 是否与链表中其
                                                 //他所有结点独立
1) CurPredecessorSet = curnode.GetPredecessorSet() //得到当前节点的前驱集
2) For each node in nodelist
3) |while(curnode! = node) |//被比较节点 node 不是当前结点 curnode
4)      PredecessorSet = node.GetPredecessorSet() //得到当前节点的前驱集
5)      If(IsCommonElement(curPredecessorSet,PredecessorSet) = = FALSE)
6)                                                      //交集是否为空,FALSE 为非空
7)      return FALSE //当前节点与其他节点不独立
8) endif|
9) return TRUE //当前节点与其他节点独立
```

3 基于压缩方法求最小割集的算法

在 BN 中求割集,就是求满足 $P(T=1|C=1)=1$ 的集合 $C(C\subseteq R)$。由于最小割集对于安全性分析更具价值,为此,具体研究求最小割集的算法。为表述方便,给出一些简单的约定:对于只包含 k 项的集合,称之为 k - 项集;对于不知是否为割集的 k - 项集称之为 k - 项候选集,并用 L_k 表示 k - 项候选集的集合;对于已知为割集的 k - 项集称之为 k - 项割集,并用 C_k 表示 k - 项割集的集合。

如果将 BN 所有根节点对应的状态进行组合来计算割集,则会形成一个 NP 难题。为了降低计算最小割集时的复杂程度,我们提出了一种基于压缩思想求最小割集的算法。其基本思想是利用"包含割集的集合必是割集"这一规则,采用逐层搜索的迭代方法去压缩搜索空间,从而降低计算复杂度。

(1) 压缩步:令 $L_k \triangleq \{l_1, l_2, \cdots, l_n\}$,其中,$l_i$ 为 k - 项候选集($i=1,\cdots,n$)。求满足下列条件的集合 C_k,即 $C_k = \{l_i | P(T=1|l_i=1)=1, l_i \in L_k, i=1,\cdots,n\}$。显然 $C_k \subseteq L_k$,从 L_k 中去掉 C_k 得到 $\bar{L}_k$,即 $\bar{L} = L_k \setminus C_k$,实现压缩。

(2) 连接步:执行连接 $\bar{L}_k \backsim \bar{L}_k$($\backsim$ 表示连接运算,其中 $\bar{L}_k$ 中的元素是可连接的,关于连接运算的详细说明,可以参见文献[10]),得到($k+1$) - 项候选集的集合 L_{k+1}。设 $\bar{l}_1$ 和 $\bar{l}_2$ 是 $\bar{L}_k$ 中的元素,记号 $\bar{l}_i[j]$ 表示 $\bar{l}_i$ 的第 j 项(例如,$\bar{l}_1[1]$ 表示 l_1 的第 1 项,$\bar{l}_1(k-1)$ 表示 l_1 的倒数第 2 项)。为方便计,假定 $\bar{l}_i$ 中的项按字典次序排序。$\bar{L}_k$ 的元素 $\bar{l}_1$ 和 $\bar{l}_2$ 是可连接的,如果它们前($k-1$)个项相同,即

$$(\bar{l}_1[1]=\bar{l}_2[1]) \wedge (\bar{l}_1[2]=\bar{l}_2[2]) \wedge \cdots \wedge (\bar{l}_1[k-1]=\bar{l}_2[k-1]) \wedge (\bar{l}_1[k] < \bar{l}_2[k])$$

条件($\bar{l}_1(k)$) < $\bar{l}_2(k)$是简单地保证不产生重复。连接 $\bar{l}_1$ 和 $\bar{l}_2$ 产生的结果是($k+1$) - 项集:{$\bar{l}_1[1]$,$\bar{l}_1[2]$,…,$\bar{l}_1[k]$,$\bar{l}_2[k]$}。算法的伪代码如下:

算法目的:通过连接候选集,逐层叠代找出最小割集。

输入:根节点状态集合 D;输出:由 D 产生的最小割集的集合 Γ

方法:

1) L_1 = Gen _ CandidateItemSet(D) //得到 1 - 项候选割集

2) C_1 = Find _ StrongCutSet _ 1 - ItemSet(D) //找出 1 - 项割集的集合

3) For ($k=2$; $L_k \neq \phi$; $k++$){

4) $\bar{L}_{k-1} = L_{k-1} - C_{k-1}$;

5) L_k = LinkGenCandidateItemSet($\bar{L}_{k-1}$) //连接产生候选集集合,当 $\bar{L}_{k-1}$为空
//集 ϕ 或只含一个元素时,L_k 为 ϕ

6) $C_k = \{l \in L_k \mid P(T=1 \mid l=1)=1\}$ //找出 k - 项割集的集合

7) }

8) return $\Gamma = \bigcup_k C_k$

4 算法复杂度分析

团树传播算法是一种良好的贝叶斯网络推理算法。团树传播算法在进行贝叶斯网络推理时,需分为 4 个步骤进行,分别是转为道义图(Moral Graph)、三角化(Triangulated)、连接成团树、推理 4 个部分[10],其中,转为道义图部分算法的时间复杂度可表示为 $O(n)$,连接成团树的算法复杂度可表示为 $O(d^2)$(d 为团的个数),求顶事件的推理算法复杂为 $O(d)$(d 远大于团中变量个数)。可以通过启发算法使三角化优化变为具有多项式复杂度的问题,目前最良好的三角化优化算法的时间复杂度为 $O(n^3)$[11]。因此整个团树传播算法的复杂度可表示为($O(n)+O(n^3)+O(d^2)+O(d)$),简记为 $O(f(n,d))$。

基于独立性分解求顶事件发生概率过程可分为 3 部分,第一部分是将贝叶斯网络分解成独立的模块,该部分的时间复杂度为 $O(n^2)$;第二部分是计算非独立部分顶节点事件概率,这里按团树传播算法计算非独立部分顶节点事件概率。设每个非独立部分对应的贝叶斯网络结点个数为 m_i,且整个贝叶斯网络共有 l 个非独立部分,记 $m = \sum_{i=1}^{l} m_i$,则有向聚合树的结点个数为($n-m+l$),其复杂度表示为 $O(\sum_{i=1}^{l} f(m_i,d_i))$;第三部分是计算有向聚合树中顶事件发生概率,该部分的时间复杂度为 $O(n-m+l)$。因此,求顶事件发生概率算法的时间复杂度可表示为 $(O(n^2)+O(\sum_{i=1}^{l} f(m_i,d_i))+O(n-m+l))$。

在最坏情况下,整个贝叶斯网络是一个非独立的网络,即 $m_i = n$, $l=1$,此时算法复杂度为 $O(\sum_{i=1}^{l} f(m_i,d_i)) = O(f(n,d))$。最良好的情况是 $m_i = 0$,则算法的时间复杂度表示为($O(n^2)+O(n)$)。大规模贝叶斯网络安全模型中,通常会存在大量的独立性节点,因此,算法平均复杂度降低,且独立性节点越多,算法的效率越高。同时,由于利用独立性将整个网络分离成独立的模块,各个模块可以分别计算,减少了一次性显式表达的项数。

5 算法的性能比较

基于 BDD(Binary Decision Diagram, BDD)算法的 FTA 方法是一种效率较高的安全性分析方法[12]。在确定情形下(故障树只能在确定逻辑门情形下进行安全性分析),为比较两种算法对同一规模系统安全分析(计算顶事件发生概率和系统的最小割集)时的效率,以导弹发射过程中三个不期望事件(陀螺平台失效、大姿态失稳、30min 准备阶段发射失败)为顶事件分别用 FTA 和 BN 建模。下面给出在配置为

Windows XP、Pentium4(R) 2.66GHz、256M下两种算法性能的对比,见表1。表中BE No.表示故障树底事件(Basic Event, BE)节点个数,Gate No.表示故障树门事件个数,P_{Top}表示顶事件发生概率,CS表示割集,SZ为程序运行时设置的堆栈大小(Stack Size),运行时间的单位为秒。

所提出算法是在默认1M的堆栈下运行的,在对系统3进行4阶割集分析时,运行时间大于1h,不再进行分析。基于BDD的算法在对系统2分析时,堆栈内存需要设置为20M,否则出现溢出错误,而对系统3进行分析时,操作系统提示内存不足。

由表1可以看出:基于贝叶斯网络的安全分析算法无论是在计算顶事件发生概率,还是在计算小于4阶的割集,都显示出了良好的时间性能。由基于BDD算法在求P_{Top}和求最小割集时所耗时间相差不大可以得出,BDD算法的大部分时间消耗在将故障树向BDD结构转换上,而且随着系统规模的增大,BDD算法消耗时间和内存都剧烈增加,直至耗尽内存。同时也注意到,尽管基于贝叶斯网络的算法在求小于4阶的割集时速度优势明显,但随着搜索割集阶数的增大,消耗时间也迅速增加。但在实时安全分析时,更看重良好的时间性能,且更关注1、2阶割集,小于5阶的割集已能满足工程的需要,并不会影响安全性分析的质量。

表1 对同一系统安全性分析时两种算法的比较

Tab.1 The comparison between two algorithms for the same system safety analysis

系统	模型		基于BDD算法		模型		基于BN的算法				
	FTA					BN		CS			
	BE No.	Gate No.	P_{Top}	CS	SZ	Node No.	P_{Top}	1阶	2阶	3阶	4阶
1	22	21	1.230	1.265	1M	43	0.0	0.000	0.103	0.381	1.249
2	32	31	80.328	81.863	20M	63	0.0	0.015	0.213	7.645	33.342
3	101	100	-	-	20M	201	0.0	1.625	81.673	2336.321	>3600

6 结束语

根据BN在安全性分析时的特点,研究了求顶事件概率和割集的算法。复杂度分析表明求顶事件发生概率算法的平均复杂度降低。同时,在同一规模系统下,将提出算法和基于BDD的算法进行安全性分析时的效率进行了比较,在满足安全性工程需求情形下,所提出算法占用内存少,运算速度快。在研究中发现,给贝叶斯网络条件概率表赋值是一项繁琐的工作。所以在大规模BN建模时,主张将故障树和BN结合起来:首先建立系统的故障树模型,然后通过转化算法将故障树转为BN,再对BN中特定的节点进行条件概率表的设置,从而达到安全建模时享受故障树的直观,安全性分析时享受BN的灵活。

参考文献:

[1] Sankaran M, Zhang R X, Natasha S. Bayesian Networks for System Reliability Reassessment [J]. Structural Safety, 2001,23(6): 231-251.

[2] Sankaran M, Ramesh R. Validation of Reliability Computational Models Using Bayes Networks [J]. Reliability Engineering and System Safety, 2005,87(10): 223-232.

[3] Bobbio A, Portinalea L, Minichinob M, et al. Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks [J]. Reliability Engineering and System Safety, 2001,71(12): 249-260.

[4] 李俭川. 贝叶斯网络故障诊断与维修决策方法及应用研究 [D]. 长沙:国防科学大学, 2002.

[5] Lepar V, Shenoy P P. A Comparison of Lauritzen and Spiegelhalter, Hugin and Shafer and Shenoy Architectures for Computing Marginals of Probability Distributions [C]//Proceedings of Uncertainty in Artificial Intelligence, 1998.

[6] Kanfmann M. 人工智能[M]. 郑扣根,庄越挺,译. 北京: 机械工业出版社, 2000.

[7] Jensen F. An Introduction to Bayesian Network[M]. New York: Springer, 1996.

[8] 金光, 动态系统可靠性分析的概念 [J]. 国防科技大学学报, 2004,26(2): 100-105.

[9] 潘金贵, 顾铁成, 曾俭. 现代计算机常用数据结构和算法 [M]. 南京: 南京大学出版社, 1994.

[10] Huang C,Darwiche A. Inference in Belief Networks: A Procedural Guide [J]. International Journal of Approximate Reasoning, 1996,12(7): 225-263.

[11] Pinar H. Minimal Triangulations of Graphs: a Survey [J]. Discrete Mathematics, 2006(3): 297-317.

[12] Sinnamon R M, Andrews J D. New Approaches to Evaluating Fault Trees [J]. Reliability Engineering and System Safety, 1997,58(17): 89-96.