基于同构子网判定的结点不可靠网络可靠度计算方法

肖宇峰

西南科技大学信息工程学院;特殊环境机器人技术四川省重点实验室,绵阳 621010

摘要　为提高结点不可靠网络的可靠度计算效率,提出一种基于子网同构判定的高效计算方法。在生成有序二元决策图(OBDD)的因子分解过程中,利用特征合并划分(CMP)识别网络分解产生的同构子网,然后根据网络中边和节点的逻辑联系,执行边替换操作将不可靠结点存储于OBDD;通过遍历OBDD计算网络的可靠度。结果显示,该方法减少了同构子网带来的重复计算,并充分利用OBDD的存储结构进一步增强了计算效率,计算中小型网络可靠度的时间保持在100 s以下,计算数百结点网络可靠度的时间保持在百秒级,且计算中大型网络的开销远低于标准二元决策图(BDD)方法。

关键词　结点不可靠网络;可靠度;同构子网判定;有序二元决策图

中图分类号　TN915　　**文献标志码**　A　　**doi**　10.3981/j.issn.1000-7857.2014.16.006

Evaluation of Reliability of Network with Unreliable Nodes Based on Isomorphism

XIAO Yufeng

Special Environment Robot Technology Key Laboratory of Sichuan Province; Information Engineering School, Southwest University of Science and Technology, Mianyang 621010, China

Abstract　To improve the efficiency in evaluating the reliability of a network with unreliable nodes, this paper proposes a computation method based on isomorphism determination. In analyzing the reliability, the CMP (characteristic mergence partition) is used to identify the isomorphic subnet generated by the network decomposition; the edge replacement operations are used to store unreliable nodes into the OBDD (ordered binary decision diagram). Not only the repeated computations from isomorphic subnets are reduced, but also the computation efficiency is enhanced by the efficient OBDD storage. On the experiment platform, this method takes less than 100 seconds for small and medium networks, and several hundreds seconds for networks with hundreds of nodes. Experiments show that this method can accurately evaluate the network reliability, and takes less than one-tenth time taken by the standard BDD (binary decision diagram) method for medium and large networks.

Keywords　network with unreliable nodes; reliability; isomorphic subnet determination; ordered binary decision diagram

网络可靠度评估起源于20世纪70年代,在计算方法、计算指标等方面已有较多成果[1-3]。但随着网络的发展,传统的可靠度计算方法无法高效、快速地分析当前的大型网络,需要在计算效率、非统计独立失效、网络多态性等方面开展深入研究。近年来,以网络连通性为可靠度指标的研究取得了不少进展,在结点不可靠网络的连通性计算方面,Xing[4]、Manoj[5]应用二元决策图(binary decision diagram,BDD)改善了可靠度计算效率,Xing的方法还考虑了网络的非统计独立失效,但当网络规模较大时会出现大量重复计算;Kuo等[6]应用Hash表记录网络分解中的同构子网,通过避免对同构子网的重复计算来提高计算效率,但随着网络规模扩大,Hash表会带来大量存储开销;Rodionov等[7]提出累计更新的网络可靠

收稿日期:2014-01-21;修回日期:2014-04-13

基金项目:国防科工局核能开发科研项目(20111137);四川省教育厅重点项目(14ZA0091);四川省应用基础研究项目(2012JYZ003);四川省科技支撑计划项目(2013GZX0152)

作者简介:肖宇峰,副研究员,研究方向为计算机网络通信及网络可靠性,电子信箱:xiaoyf_swit1@163.com

引用格式:肖宇峰. 基于同构子网判定的结点不可靠网络可靠度计算方法[J]. 科技导报, 2014, 32(16): 39-44.

度计算方法,但只能得到近似值;Kim等[8]提出递归分解算法(recursive decomposition algorithm,RDA)计算可靠度,但其效率并未在一般网络中得到充分验证;Lin等[9,10]针对不可靠结点提出多态可靠性分析方法,但其效率在网络规模增大时会下降。为提高结点不可靠网络的可靠度计算效率,本文从减少冗余计算、存储不可靠结点出发,研究一种基于同构子网判定的高效计算方法。本文所用主要符号的含义为:$G(V, E)$:G表示1个网络,V为点集(包括1个源端和1个终端),E为边集;$O_{\text{obdd}}(G)$:网络G的有序二元决策图(OBDD)结构;$f_{\text{rel}}(G)$:网络G的可靠度计算函数;G_{+e}:网络G中e边收缩后得到的子网;G_{-e}:网络G中e边删除后得到的子网;G_{+a+b}:连续收缩网络G中a边、b边得到的子网;G_{-a-b}:连续删除网络G中a边、b边得到的子网;G_{+a-b}:连续收缩网络G中a边后再删除b边得到的子网。需要说明的是,本文讨论的可靠度是指网络源端与终端之间保持连通的概率,源端与终端之间存在一条连通的路径则认为网络可靠,否则认为网络不可靠。

1 结点可靠网络的OBDD创建

1.1 应用因子分解创建OBDD

边的因子分解方法通常假设网络边不可靠而结点可靠[1],根据网络中边的两种情况进行分解:边可靠则收缩边,把边的2个端点合并成1个点;边不可靠则删除边,把边的2个端点保留在网络中;上述操作分别生成2个子网,并对子网同样执行收缩边和删除边操作,直到网络中所有边被访问。

图1所示为利用边的因子分解创建OBDD的实例。

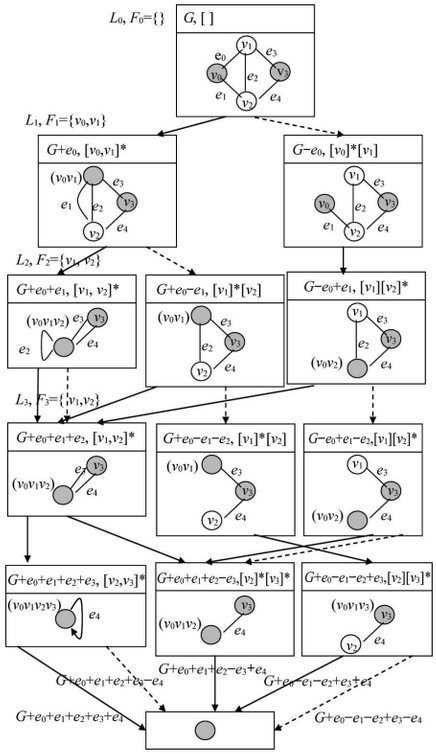


图1 网络的因子分解过程

Fig. 1 Network decomposition

网络G及其分解得到的子网按边序e_0, e_1, e_2, e_3, e_4进行5次分解,灰色圆圈表示包含或者合并了源端(终端)的结点,图1中省略了源端和终端分离的子网。除第一次外的每次分解都是在上次分解得到的子网之上进一步分解,得到更小的子网。分解到最后只保留了源端和终端合并为一点的情形,表示源端和终端之间存在连通路径。

定义1 第k层分解:按边序$e_0, e_1, e_2, \cdots, e_{m-1}$执行因子分解时,对第$k$条边$e_k$执行收缩边和删除边操作的过程称为第$k$层分解。

按确定边序对网络进行因子分解,实质是根据边是否可靠来分析网络是否连通。利用OBDD的ITE运算可以把每次分解对应的逻辑判断记录下来,最后得到描述整个网络的状态图。图2给出上述实例网络(图1)的OBDD,创建OBDD的具体原理可参考文献[4]、[5]、[6]。

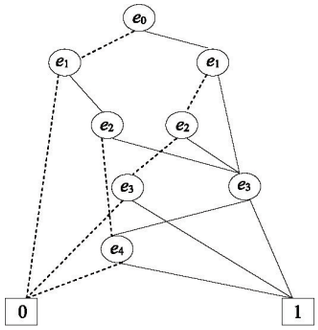


图2 实例网络的OBDD结构

Fig. 2 OBDD of example network

网络分解会生成大量同构子网,重复对这些结构相同的子网执行分解将带来冗余计算。为此,需要把已经得到OBDD结构的子网记录到Hash表。在后续分解时,如果发现Hash表记录了与当前网络同构的子网,就直接返回子网OBDD,不再分解当前网络并创建其OBDD。这样,Hash表的引入避免重复计算,节省了计算开销。如图1所示,对于第3分解层L_3,G_{+e0+e1}收缩e_2边产生子网$G_{+e0+e1+e2}$,G_{+e0-e1}收缩e_2边产生子网$G_{+e0-e1+e2}$。显然,这2个子网满足同构关系。如果对G_{+e0+e1}执行收缩e_2边时把$G_{+e0+e1+e2}$及其OBDD结构保持在Hash表中,下次对G_{+e0-e1}执行收缩e_2边时就不用重复创建子网$G_{+e0-e1+e2}$的OBDD结构。

1.2 应用同构子网判定改进`OBDD创建算法

1.2.1 子网同构与特征合并划分

从上述OBDD创建过程可见,提高计算效率降低开销的一个重要途径是判断子网同构并避免对结构相同的子网进行重复计算。对于两个子网,判断二者是否同构是很难的问题,计算复杂度很高[10]。下文描述了一种根据特征合并划分(CMP)来判定子网同构的方法,利用网络分解时已有的子网特征描述网络结构,降低了判定子网同构的难度。

定义2 界边集及界边补集:执行第k层分解时,已被访问的边组成的集合称为界边集,其中边的状态是确定的(可

靠或者不可靠);其补集称为界边补集,其中边的状态同为不确定。执行第k层分解时,界边集和界边补集分别表示为E_k和E'_k。如图1所示,对于第1分解层L_1,$E_1=\{e_0\}$,$E'_1=\{e_1, e_2, e_3, e_4\}$。

定义3　界点集:因子分解中,同时与E_k和E'_k边相邻的点集。执行第k层分解时,界点集可以表示为E_k。如图1所示,在L_1层,同时与E_1和E'_1边相邻的点有v_0和v_1,所以,$E_1=\{v_0, v_1\}$。

定义4　合并关系:因子分解中,如果界点集中的结点通过收缩边合并成一个点,那么这些点具有合并关系。如图1所示,在L_1层,F_1的2个点v_0、v_1在收缩边e_0后合并成1个点,它们之间存在合并关系。

定义5　块:满足合并关系的等价类称为块,用符号[ ]把具有等价关系的点包括在其中。如图1,$[v_0, v_1]$就是由v_0和v_1构成的块。

定义6　合并划分:根据界点集中结点的合并关系把结点划分成等价类,这样的划分称为合并划分。而且,一个合并划分可以用界点集的一个特定的合并关系表示。如图1所示,在L_1层,根据的E_1与E'_1关系得到F_1的2个合并划分$[v_0, v_1]$和$[v_0]$ $[v_1]$,在$[v_0, v_1]$划分中,合并关系中只存在1个等价类$\{v_0, v_1\}$;在$[v_0]$ $[v_1]$划分中,合并关系中存在2个等价类$\{v_0\}$和$\{v_1\}$。

定义7　特征合并划分:用*号把源端或终端所在的块标记出来后,合并划分就称为特征合并划分。如图1所示,在L_1层,合并划分$[v_0][v_1]$的$[v_0]$块包含了源端,所以该合并划分用特征合并划分表示为$[v_0]^*[v_1]$。

定理1　因子分解中,每一个子网对应着1个特征合并划分。而且,对网络执行连续操作(收缩边和删除边)后,当前分解层上2个子网具有相同的特征合并划分(由相同界边集和界点集得到),那么这2个子网同构[11]。

如图1所示,对于第3分解层L_3,G_{+e0+e1}收缩e_2边产生的子网$G_{+e0+e1+e2}$对应划分为$[v_1,v_2]^*$,G_{+e0-e1}收缩e_2边产生的子网$G_{+e0-e1+e2}$对应划分为$[v_1,v_2]^*$。显然,2个子网的特征合并划分相同,子网结构也是同构的。可见,在收缩和删除边生成多个子网的过程中,根据界点集可以计算出这些子网的特征合并划分,判断子网同构只需要判断其划分是否相同。

1.2.2　特征合并划分的三级分类

如果每个特征合并划分具有唯一的编号,那么就很容易判断划分是否相同。为方便计算划分编号,下文首先对特征合并划分进行三级分类。

一级分类:按照块的数目对合并划分进行分类。如图3所示,$F_k=\{v_1, v_2, v_3\}$可分成3类:1块的合并划分,比如$[v_1,v_2,v_3]$;2块的合并划分,比如$[v_1,v_3][v_2]$;3块的合并划分,比如$[v_1][v_2][v_3]$。

二级分类:根据块数对某个一级分类进一步细分。图3中,1块的划分只有1种,即$[v_1,v_2,v_3]$;2块的划分有3种,$[v_1,v_3]$ $[v_2]$、$[v_1][v_2,v_3]$和$[v_1,v_2][v_3]$;3块的划分只有1种,即$[v_1][v_2][v_3]$。

三级分类:根据标记位置对某个二级分类进一步细分。如果是j个块的划分,那么由三级分类可以得到2^j个可能的特征合并划分。例如,二级分类的$[v_1,v_3][v_2]$划分有2个块,所以三级分类的特征合并划分有2^2个块:$[v_1,v_3][v_2]$,$[v_1,v_3]^*$ $[v_2]$,$[v_1,v_3]$ $[v_2]^*$,$v_1,v_3]^*[v_2]^*$。

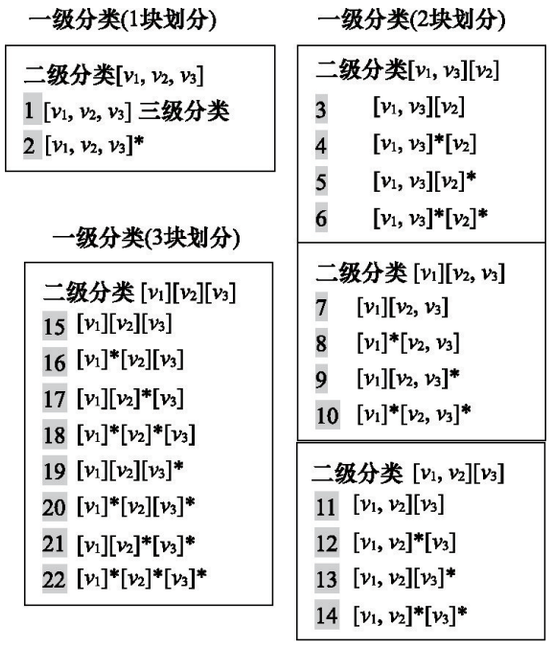


图3　特征合并划分的三级分类

Fig. 3　Three ranks classifications of characteristic mergence partition

1.2.3　特征合并划分的编号

1)合并划分的数目。对于界点集F_k,元素个数$i=|F_k|$,j表示块数,j块合并划分数目可表示为$A(i,j)$,其计算公式为

$$A(i,j)=j\cdot A(i-1,j)+A(i-1,j-1) \quad (1)$$

查询表1可快速返回$A(i,j)$的具体数值。

表1　合并划分计算

Table 1　Calculating table for mergence partition

元素数	块数				
	1	2	3	4	5
1	1	—	—	—	—
2	1	1	—	—	—
3	1	3	1	—	—
4	1	7	6	1	—
5	1	15	25	10	1
6	1	31	90	65	13
7	1	63	301	350	140
8	1	127	966	1701	1050
9	1	255	3025	7770	6951
10	1	551	9330	34105	42525

2）合并划分的计算机存储。算法实现时，用线性表part[ ]表示F_k的合并划分，如图4所示。比如，F_k= {v_1, v_2, v_3}，它的一个合并划分[v_1, v_2][v_3]表示为图4中的线性表part[ ] ={1,1,2}。其中，part[1]=1和part[2]=1表示v_1、v_2属于第1个块，即块[v_1, v_2]；part[3]=2表示v_3属于第2个块，即块[v_3]。

part[ ]	1	2	3	4
	1	1	2	

图4 合并划分的表示

Fig. 4 Representation of mergence partition

为表示特征合并划分，还需要一个标志表mark[ ]，元素mark[r]=0表示第r块无标记，mark[r]=1表示第r块有标记。比如，F_k= {v_1, v_2, v_3}，它的一个特征合并划分[v_1, v_2]*[v_3]，其标志表mark[ ]={1,0}。

3）合并划分的编号生成。对于界点集F_k（元素个数i=|F_k|），num表示编号，part[ ]和mark[ ]表示有j个块的特征合并划分。根据上面的排序方法，可以按如下步骤生成特征合并划分的编号：

步骤1 根据表part[ ]确定所属的一级分类，得到元素个数i和块数j；

步骤2 根据表part[ ]各元素所处块的位置确定所属的二级分类，用c_1表示其分类序号；

步骤3 根据表mark[ ]确定所属的三级分类，用c_2表示其分类序号；

步骤4 计算特征合并划分的编号。计算公式为

$$num = \sum_{h=1}^{j-1} A(i,h)\cdot 2^h + (c_1 - 1)\cdot 2^j + c_2 \qquad (2)$$

按照上述方法，对于F_4 = {v_1, v_2, v_3}的特征合并划分[v_1]*[v_2,v_3]，可用数组part[ ]={1, 2, 2}和mark[ ]={1,0}表示出来，则可确定其所属的一级分类，元素个数为3、分块数为2；确定合并划分的二级分类序号c_1=2；确定合并划分中第一个特征合并划分的三级分类序号c_2 = 2；根据式(2)计算出特征合并划分的编号为

$$num = \sum_{h=1}^{1} A(3,h)\cdot 2^h + (2-1)\cdot 2^2 = 8$$

1.2.4 改进的OBDD创建算法

改进的OBDD创建算法伪码如图5所示，其中OBDDConstruct_cmp函数用于生成OBDD，输入参数为某网络G，输出参数为生成的OBDD根结点。该算法中有两个表：一是记录第k层的OBDD结点，表示为L_k；二是Hash表，记录不同分区对应的OBDD。Hash表是一个线性表，其查询索引值是各子网特征合并划分(CMP)的编号，元素是该CMP对应的OBDD结点，表示为obdd_hash。part2no( )函数可实现CMP编号的计算过程，其输出的CMP编号即可用于判定当前子网是否与Hash表中某子网同构。

```
OBDDConstruct_cmp(network G)
{ root=bddnode = CreateBddnode(L0); //创建根结点
  for (k=0; k<m; k++)
  {Fk+1= Construct(ek); /*生成界点集*/
    for each bdd_node in Lk; /*第k层的BDD结点*/
    { cmp0=DelEdge(Fk+1, ek); /*删除边*/
      if (source and target are disconnected)
        left-child of bddnode=bddfalse;
      else
      { num=part2no(cmp0,| Fk+1|);/*计算划分编号*/
        bdd child_bddnode = LookupHash(num); /*Hash查询*/
        if (child_bddnode do not exists in obdd_hash)
        { child_bddnode=CreateBddnode(Lk+1);
          InsertintoHash(child_bddnode); /*插入Hash表*/
        }
        left-child of bdd_node= child_bddnode;
      }
      cmp1=ContractEdge(Fk+1, ek) /*收缩边*/
      if (source and target are unified)
        right-child of bddnode=bddtrue;
      else
      { num=part2no(cmp1,|Fk+1|);
        bdd child_bddnode = LookupHash(num);
        if (child_bddnode do not exists in obdd_hash)
        { child_bddnode=CreateBddnode(Lk+1);
          InsertintoHash(child_bddnode);
        }
        right-child of bdd_node= child_bddnode;
      } } }
  return root;
}
```

图5 改进的OBDD算法伪码

Fig. 5 Pseudo code of OBDD algorithm

2 结点不可靠网络的OBDD创建

假设网络结点可靠时，网络G的e边用OBDD的布尔变量V_e表示，则网络G的OBDD表达式为

$$O_{obdd}(G) = \{V_e \wedge O_{obdd}(G)|_{V_e=1}\} \vee \{\overline{V_e} \wedge O_{obdd}(G)|_{V_e=0}\} \qquad (3)$$

式中，$O_{obdd}(G)|_{V_e=1}$为e边处于可靠状态时网络G的OBDD；$O_{obdd}(G)|_{V_e=0}$为e边处于不可靠状态时网络G的OBDD。结点不可靠时需要考虑边的2个端点的逻辑状态，用布尔变量V_a、V_b分别表示e边的端点a、b。那么，e边的完整布尔表达式为$V_a \wedge V_e \wedge V_b$。为得到结点不可靠网络G的OBDD结构，需要进行如下运算：

$$\text{obdd1} = O_{obdd}(\text{a}) \text{ and } O_{obdd}(\text{e})$$

$$\text{obdd2} = \text{obdd1 and } O_{obdd}(\text{b})$$

$$O_{obdd}(G) = O_{obdd}(G)|_{O_{obdd}(e)=\text{obdd2}}$$

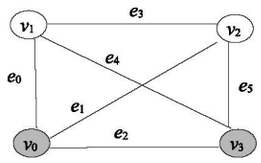
其中，obdd2是布尔表达式$V_a \wedge V_e \wedge V_b$的OBDD运算结果。上述运算的含义是用端点不可靠的e边替换OBDD中原来的e边，执行OBDD的composition运算[6]。本文将该过程称为边替换。结点不可靠网络OBDD的创建算法伪码如图6所示。

假设图7(a)所示网络结点可靠，按前述方法可生成如图7(b)所示的该网络的OBDD。对该OBDD执行图6的算法后，可得到图8所示结点不可靠网络的OBDD结构。

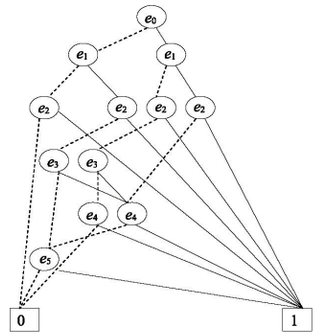
```
/*bdd_G 输入结点可靠网络的OBDD*/
/* newbdd_G 输出结点不可靠网络的OBDD */
OBDDConstrct_urn(bdd_G)
{/* a,b 表示e的两个端点*/
  newbdd_G=bdd_G;
  for (each node of e in bdd_G)
  {e = bdd_var(node); /*返回BDD变量*/
    get_terminals(e, a, b); /*返回边的两个端点*/
    obdd1= bdd_and(bdd_ithvar(a), node);
    obdd2= bdd_and(obdd1, bdd_ithvar(b))
    newbdd_G =bdd_compose(newbdd_G, obdd2, node);
  }
  return newbdd_G;
}
```

图6　结点不可靠网络的OBDD生成算法伪码

Fig. 6　Algorithm pseudo code of OBDD generation for network with unreliable nodes



(a) 结点可靠网络



(b) 结点可靠网络的OBDD

图7　实例网络及其OBDD结构

Fig. 7　Example network and its OBDD

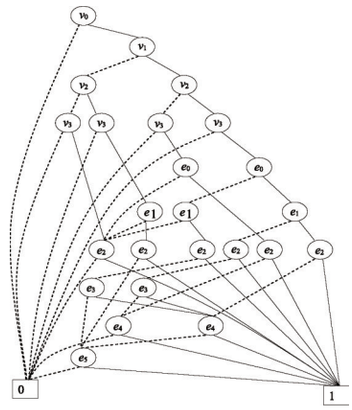


图8　结点不可靠时网络的OBDD结构

Fig. 8　OBDD of example network with unreliable nodes

3　结点不可靠网络的可靠度计算

3.1　计算方法

按照上述改进算法创建节点不可靠网络G的OBDD，然后通过递归公式计算网络G的可靠度。计算公式为

$$f_{r\_o}(O_{\text{obdd}}(G))=\begin{cases}1, & \text{if } O_{\text{obdd}}(G)=\text{bdd tr ue}\\ 0, & \text{if } O_{\text{obdd}}(G)=\text{bddfalse}\\ P(x_k=1)\cdot f_{r\_o}(O_{\text{obdd}}(G)|_{x_k=1})+ & \\ P(x_k=0)\cdot f_{r\_o}(O_{\text{obdd}}(G)|_{x_k=0}), & 0\leqslant k<n,\text{o th erwise}\end{cases} \quad (4)$$

式中，$O_{\text{obdd}}(G)|_{x_k=1}$表示$x_k$=1时网络G的OBDD，即$x_k$对应结点或边可靠时网络G的OBDD；$O_{\text{obdd}}(G)|_{x_k=0}$表示$x_k$=0时网络$G$的OBDD，即$x_k$对应结点或边不可靠时网络$G$的OBDD符号；$P(x_k=1)$表示$x_k$对应结点或边的可靠概率；$P(x_k=0)$表示$x_k$对应结点或边的不可靠概率；bddtrue、bddfalse分别表示OBDD中的逻辑真和逻辑假；k的初始值为0。如图9中所示，NetRel_urn(G)给出了结点不可靠网络的可靠度计算伪码，其中，G表示网络；OBDDConstruct_cmp( )表示图5的改进OBDD创建算法；OBDDConstruct_urn( )表示图6的结点不可靠网络OBDD创建算法；OBDDRel( )表示式(4)的网络G可靠度算法。

```
NetRel_urn(G)
{OrderEdge(G); /*确定边序*/
  bdd_G =OBDDConstruct_cmp(G); /*结点可靠网络*/
  newbdd_G=OBDDConstrct_urn(bdd_G); /*结点不可靠网络*/
  reliability =OBDDRel(newbdd_G); /*计算可靠度*/
  return reliability;
}
double OBDDRel(bdd bdd_graph)
{ if(bdd_graph is bddtrue)          return 1;
  else if(bdd_graph is bddfalse)  return 0;
  Root(bdd_graph); /*返回根结点*/
  p = operation probability of node v;
  if (bdd_graph exists in the bdd_hash)
  {r = LookupRELHash(bdd_graph); /*Hash查询*/
    return r;
  }
  bdd bdd_high = high(bdd_graph);
  bdd bdd_low = low(bdd_graph);
  reliability =p*OBDDRel(bdd_high)
              +(1-p)* OBDDRel(bdd_low));
  InsertintoRELHash(bdd_graph,reliability); /*插入Hash表*/
  return reliability;
}
```

图9　网络可靠度算法伪码

Fig. 9　Algorithm pseudo code of network reliability computation

3.2　实验验证

采用Buddy版BDD[13]设计了上述结点不可靠网络可靠度计算方法的C语言实现程序，命名为NetRel_2TN。为方便对比分析，另设计了实现文献[4]计算方法的C语言实现程序，命名为Bdd_X。实验用计算机的CPU工作频率为2.4 GHz，

内存容量为2 GB,操作系统为2.6内核的红帽Linux。选择如图10所示的9个实例网络作为实验的基准网络,检验本文算法的正确性和效率[4-7]。这9个实例网络已被很多可靠度论文引用,网络中的黑实点分别表示源端和终端,点和边的可靠度为0.9。

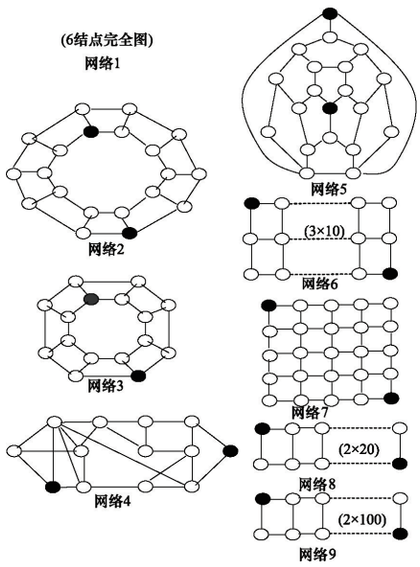


图10 实验基准网络

Fig. 10 Benchmark network

实验结果列于表1,展示了Bdd_X和NetRel_2TN的运行情况。"可靠度"给出各网络的可靠度值,"Bdd_X"给出文献[4]方法计算可靠度的时间开销,"NetRel_2TN"给出本文方法计算可靠度的时间开销。每个计算程序被执行10次后,将其平均计算时间记录为时间开销,数据表明Bdd_X的开销高于NetRel_2TN。从网络8和网络9的计算结果发现,随着网络规模增大,Bdd_X算法的计算时间开销显著上升,1 h内无法得到计算结果,而NetRel_2TN在1 h内完成了计算。结果表明,本文方法可有效地评估结点不可靠网络的可靠度,当网络规模较大时,其计算时间开销明显低于文献[4]方法。

表2 实验结果比较

Table 2 Comparisons between Bdd_X and NetRel_2TN

网络	可靠度	Bdd_X /s	NetRel_2TN /s
1	0.80979	5.14267	0.83177
2	0.74817	12.8314	2.43691
3	0.76978	7.39711	1.12217
4	0.75226	94.1491	1.42514
5	0.78479	103.752	5.41668
6	0.60963	529.624	23.4135
7	0.71199	335.028	8.01731
8	0.23016	(超过1 h)	57.0312
9	0.00111	(超过1 h)	416.231

4 结论

提出一种应用同构子网判定的不可靠节点网络可靠度计算方法。利用CMP识别因子分解产生的同构子网,可避免同构子网带来的冗余计算,提高计算效率;对边的OBDD变量结点执行边替换操作,利用OBDD的高效存储结构可进一步减少冗余计算。在与文献[4]方法的对比实验中,基于同构子网判定的方法可较快地计算一些规模较大网络的可靠度。

在分析网络可靠性时,本文假设结点和边的故障统计独立。在实际网络中,该假设并不总是成立,忽略非统计独立失效可能会降低可靠度计算的准确性。为此,将把建立非统计独立失效模型、提高可靠度计算实用性作为后续的研究工作。

参考文献(References)

[1] Ball M O, Magnanti T L, Monma C L, et al. Network Models, volume 7 of Handbooks in operations research and management science[J]. Elservier Science Publishing Company, 1995(12): 2-7.

[2] 吴俊, 段东立, 赵娟. 网络系统可靠性研究现状与展望[J]. 复杂系统与复杂性科学, 2011, 8(2): 77-86.
Wu Jun, Duan Dongli, Zhao Juan. Status and prospects on network reliability[J]. Complex Systems and Complexity Science, 2011, 8(2): 77-86.

[3] 江逸楠, 李瑞莹, 黄宁. 网络可靠性评估方法综述[J]. 计算机科学, 2012, 39(5): 9-13.
Jiang Yinan, Li Ruiying, Huang Ning. Survey on network reliability evaluation methods[J]. Computer Science, 2012, 39(5): 9-13.

[4] Xing L. An efficient binary- decision- diagram- based approach for network reliability and sensitivity analysis[J]. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 2008, 38(1): 105-115.

[5] Singhal M. An alternate approach to compute the reliability of a network with imperfect nodes using binary decision diagrams[J]. Oriental Journal of Computer Science & Technology, 2012, 5(2): 181-188.

[6] Kuo S Y, Yeh F M, Lin H Y. Efficient and exact reliability evaluation for networks with imperfect vertices[J]. Reliability, IEEE Transactions on, 2007, 56(2): 288-300.

[7] Rodionov A, Migov D, Rodionova O. Improvements in the efficiency of cumulative updating of all- terminal network reliability[J]. Reliability, IEEE Transactions on, 2012, 61(2): 460-465.

[8] Kim Y, Kang W H. Network reliability analysis of complex systems using a non- simulation- based method[J]. Reliability Engineering & System Safety, 2013, 110(1): 80-88.

[9] Lin Y K, Chang P C. Maintenance reliability of a computer network with nodes failure in the cloud computing environment[J]. International Journal of Innovative Computing, Information and Control, 2012, 8(6): 4045-4058.

[10] Lin Y K, Chang P C. A novel reliability evaluation technique for stochastic-flow manufacturing networks with multiple production lines [J]. Reliability, IEEE Transactions on, 2013, 62(1): 92-104.

[11] Imai H, Sekine K, Imai K. Computational investigations of all-terminal network reliability via BDDs[J]. Transactions on Fundamentals, 1999, 82 (5): 714-721.

[12] Hardy G, Lucet C, Limnios N. K-terminal network reliability measures with binary decision diagrams[J]. Reliability, IEEE Transactions on, 2007, 56(3): 506-515.

[13] Andersen H R. An introduction to binary decision diagrams[M]. Amsterdam: Elsevier, 1997.

(责任编辑 韩星明)