

基于路径函数和 BDD 的网络可靠度分析

潘竹生, 陈荣根, 莫毓昌

(浙江师范大学 数理信息学院, 浙江 金华 321004)

摘 要: 现有的二终端网络可靠度评估方法, 多数基于不交积和, 由于没有充分利用普遍存在的同构子网特性, 导致存在大量冗余计算, 无法适用于大型网络. 为此, 本文提出了一种基于路径函数和 BDD 的网络可靠度分析方法, 利用图 Hash 技术识别同构子网, 从而简化路径函数的构建, 再利用 BDD 高效地操纵路径函数计算网络可靠度. 实验结果表明, 该方法性能稳定且高效, 适用于更大规模的网络可靠性分析.

关键词: 二终端可靠性; 路径函数; 二叉决策图

中图分类号: TB114

文献标识码: A

文章编号: 1000-7180(2012)12-0157-06

Computing Network Reliability Based on Path-Function and BDD

PAN Zhu-sheng, CHEN Rong-gen, MO Yu-chang

(College of Mathematics Physics and Information Engineering, Zhejiang Normal University, Jinhua 321004, China)

Abstract: For computing network reliability, most published algorithms are based on the sum of disjoint products. However they lack the capability to avoid redundant computation due to isomorphic sub-problems which are common in the network, and can not be applied to larger networks. To overcome these problems, we present an efficient method which is based on path function and BDD (binary decision diagram). First, the success path function of a given network is constructed by traversing the network. In this process, the sub-graph isomorphism is efficiently identified. Then in order to further improve performance, we efficiently manipulate the path function using BDD and calculate the reliability measures. The results of experiment show that our method is stable and efficient and can be applied to a number of large-scale networks.

Key words: terminal-pair reliability; path function; Binary Decision Diagram (BDD)

1 引言

诸如互联网、通信网、电力网、交通网等, 网络已经成为人类生产、生活不可缺少的一部分. 一旦这些网络发生故障, 将造成重大甚至灾难性影响. 因此, 对这些网络给出准确的可靠性评价, 确保网络可靠运行成为一个极端重要的问题^[1-5].

在已有的研究中, 文献[6]运用集合论, 提出利用递归思想在源点到汇点的最短路径上分区网络, 直到源点和汇点完全收缩或完全断开^[7]; 应用布尔代数和图论的相关理论, 沿着与源点连接的边化简网络, 计算可靠性; 文献[8]采用集合理论, 在条件概

率下消减路径迭代以简化可靠性计算, 文献[9]基于边删除和边收缩来分解网络, 直到源点和汇点完全收缩或完全断开; 文献[10]在不交积和方法中运用基数排序、字典排序、海明距离排序及它们的组合来消除冗余, 加速可靠性计算; 文献[11]提出了基于割集的二终端可靠性分析方法. 近些年来, 文献[12]提出二元决策图模型 BDD(Binary Decision Diagram)以简化布尔函数的表示; 文献[13]将 OBDD(Ordered Binary Decision Diagram)应用到布尔函数的表示和操作当中; 文献[14-15]将 OBDD 引入到网络可靠性分析中; 文献[16]提出了基于边扩展的二终端可靠性评估方法; 文献[17]提出了基于边扩展

的 K 终端可靠性评估方法并证明了 OBDD 方法的效率;文献[18]Hardy 引入边界集和图分区的思想提出了更加高效的 K 终端评估方法. 在实际分析过程中,我们发现基于最小路集、拓扑化简、图分解等技术的可靠性分析算法,虽然对于中小规模的网络被证明是合理有效的,但存在两个固有缺点:1)操作较大网络结构时,效率低下;2)没有考虑同构子网的存在性,因此无法避免冗余计算.

针对上述两个问题,在 Kuo 和 Hardy 的研究基础上^[12-18],本文提出了基于路径函数和 BDD 的网络可靠度分析方法. 该方法将路径函数思想应用到网络分析中,借助于 BDD 缓解了大网络分析过程中的时空需求爆炸问题,并利用网络 Hash 解决了同构子网的识别问题. 实验结果表明,该方法精确且高效,对于 $N \times N$ 型、 $2 \times N$ 型的格状网络,具有很好的性能,能够应用于大规模网络可靠性分析.

2 算法设计

本文所给出的网络可靠度分析算法包含三个步骤:路径函数生成、等价 BDD 生成和可靠度评价.

2.1 路径函数生成

定义 1 设二终端网络 G 上的边只有连通和不连通两种状态,且节点完好,则其路径函数可表示成函数 $f(x_1, x_2, x_3, \dots, x_n)$:

$$f(x_1, x_2, x_3, \dots, x_n) = \begin{cases} 1 & \text{两端点连通时} \\ 0 & \text{其他} \end{cases} \quad (1)$$

网络路径函数生成算法的基本思想是从某个源点 S 开始对网络进行递归遍历,并在遍历过程中构造子网,直到到达汇点 T . 具体步骤如下:

(1) 沿着与源点 S 连接的边进行网络遍历. 假设给定网络 G 中与源点 S 连接的边为 $x_i (i=1, 2, \dots, k)$, 分别沿着边 x_i 进行网络遍历,并构造 k 个子网 $G * x_i$. 子网 $G * x_i$ 是通过收缩边 x_i 得到,收缩规则是将网络 G 中边 x_i 的端点 S (即源点 S) 合并到另一端点并作为新的源点,同时删除所有与原先源点连接的边. 在得到网络 G 的 k 个子网之后,把它们作为网络 G 沿边 x_i 的孩子节点,网络 G 的路径函数可以表示为:

$$P(G) = x_1 P(G * x_1) + x_2 P(G * x_2) + \dots + x_k P(G * x_k) \quad (2)$$

(2) 对于每一个子网 $G * x_i$,重复执行以上步骤直到源点 S 和汇点 T 合并,此时返回逻辑真值. 通过递归地复合中间路径函数的结果,得到给定网络 G 的最终路径函数.

对于图 1 中的网络 G ,网络遍历过程如图 2 所示. 图 3 给出了每个子网的路径函数表达式,递归地复合各子网的路径函数,得到 G 的路径函数:

$$\begin{aligned} P(G) &= X_1 G_1 + X_2 G_2 + X_3 G_3 \\ &= X_1 X_4 + X_2 (X_6 G_4 + X_5 G_6) + X_3 (X_6 G_5 + X_7 G_7) \\ &= X_1 X_4 + X_2 (X_6 (X_7 G_7) + X_5 (X_8 G_9 + X_9)) + X_3 \\ &\quad (X_6 (X_5 G_6) + X_7 (X_8 G_{10} + X_{10})) \\ &= X_1 X_4 + X_2 (X_6 (X_7 (X_8 G_{10} + X_{10})) + X_5 (X_8 X_{10} + \\ &\quad X_9)) + X_3 (X_6 (X_5 (X_8 G_9 + X_9) + X_7 (X_8 X_9 + \\ &\quad X_{10}))) \\ &= X_1 X_4 + X_2 (X_6 (X_7 (X_8 X_9 + X_{10})) + X_5 (X_8 X_{10} + \\ &\quad X_9)) + X_3 (X_6 (X_5 (X_8 X_{10} + X_9) + X_7 (X_8 X_9 + \\ &\quad X_{10}))) \\ &= X_1 X_4 + X_2 X_6 X_7 X_8 X_9 + X_2 X_6 X_7 X_{10} + X_2 X_5 X_8 X_{10} \\ &\quad + X_2 X_5 X_9 + X_3 X_6 X_5 X_8 X_{10} + X_3 X_6 X_5 X_9 + \\ &\quad X_3 X_7 X_8 X_9 + X_3 X_7 X_{10} \end{aligned}$$

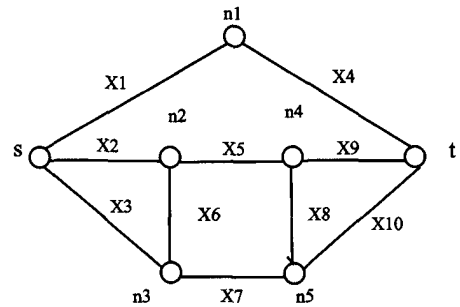


图 1 网络 G

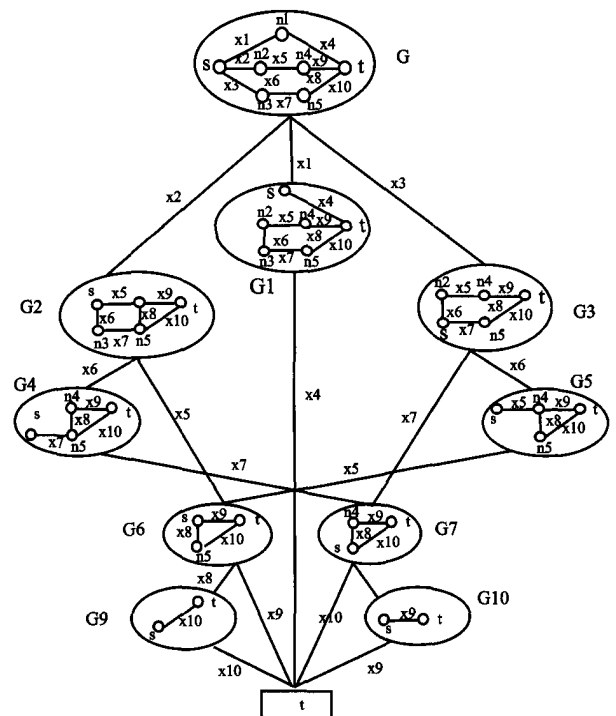


图 2 G 的子网构造过程

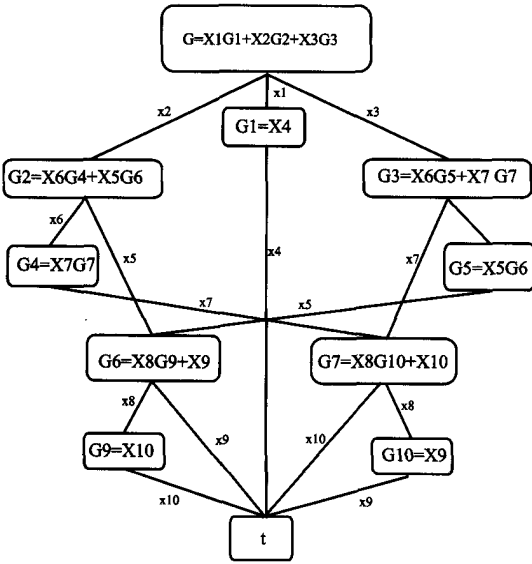


图3 网络G的路径函数生成过程

2.2 等价 BDD 生成

首先介绍香农解构 (Shannon Decomposition) 定理, 设 f 是布尔变量集合 X 上的布尔表达式, x 是 X 上的变量, 则 f 可以写成:

$$f = (x \wedge F_1) \vee (\neg x \wedge F_0) \quad (3)$$

其中 F_1 和 F_0 的定义为 $F_1 \equiv f(x=1)$, $F_0 \equiv f(x=0)$.

ite(if-then-else) 是一种简洁表达布尔表达式的方法:

$$\text{ite}(F, G, H) = (F \wedge G) \vee (\neg F \wedge H) \quad (4)$$

利用 ite 可以简洁的表达 f 在 X 上的香农解构:

$$f = (x \wedge F_1) \vee (\neg x \wedge F_0) = \text{ite}(x, F_1, F_0) \quad (5)$$

Akers 给出了 BDD 的基本定义^[12]. 一个 BDD 是基于香农解构定理刻画变量集合 X 上的布尔表达式 f 的有向无环图. BDD 有两个终节点 (标记为 0 和 1) 表示布尔常数 0 和 1. BDD 中每个中间节点 (标记为布尔变量 x) 对应一个表达式 f . 利用该节点的两个出边: 1 边 (或 then 边) 和 0 边 (或 else 边), 刻画了 f 在 x 上的 ite 结构. 图 4 中 BDD 就是布尔表达式 $f = (x_1 + x_3) * (x_2 + x_3) = \text{ite}(x_1, \text{ite}(x_2, 1, \text{ite}(x_3, 1, 0)), \text{ite}(x_3, 1, 0))$.

生成一个布尔表达式对应的 BDD 首先需要把所有的布尔变量进行排序. 变量排序对 BDD 的大小有极端重要的影响, 一个好的变量排序能够把一个大尺度的布尔表达式转换成为一个可以分析的小尺度的 BDD. 导致最小尺度的 BDD 的变量排序称为最小变量排序. 通常采用启发性的策略来生成一些

近似最小的变量排序. 本文采用基于广度优先搜索的变量排序策略, 相比较于深度优先搜索的排序策略具有更好地性能^[16].

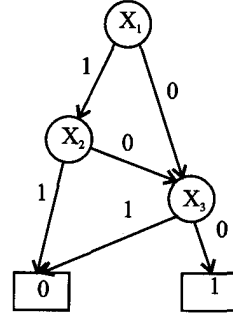


图4 $f = (x_1 \vee x_3) \wedge (x_2 \vee x_3)$ 的 BDD

根据网络路径函数可以应用下面的 BDD 基本操作来生成等价 BDD:

$$G \langle op \rangle H = \text{ite}(x, G_1, G_0) \langle op \rangle \text{ite}(y, H_1, H_0)$$

$$= \begin{cases} \text{ite}(x, G_1 \langle op \rangle H_1, G_0 \langle op \rangle H_0) & x = y \\ \text{ite}(x, G_1 \langle op \rangle H, G_0 \langle op \rangle H) & x < y \\ \text{ite}(y, G \langle op \rangle H_1, G \langle op \rangle H_0) & x > y \end{cases} \quad (6)$$

其中 G 和 H 是布尔表达式, $\langle op \rangle$ 表示逻辑与、逻辑或.

当 $F=0/1, H=0/1/\text{ite}(y, H_1, H_2), F \langle op \rangle H$ 可以通过表 1 中的计算表直接计算.

表1 特殊 $G \langle op \rangle H$ 运算的计算表

	$H=0$	$H=1$	$\text{ite}(y, H_1, H_0)$
$G=0$	0/0	0/1	0/ $\text{ite}(y, H_1, H_0)$
$G=1$	0/1	1/1	$\text{ite}(y, H_1, H_0)/1$

3 算法实现

3.1 路径函数生成

路径函数生成的递归算法实现:

```

Path_Function_Construct(G) =
    If(G is the sink) Return ONE;
    If(pf_ <= find_hash_table(G)) Return pf_;
    pf_ <= ZERO;
    For each xi connected to the S in G
        sub_G <= G * xi;
        Eliminate redundant nodes in sub_G;
        pf_ <= pf_ + xi * Path_Function_Construction(sub_G);
    Insert_hash_table(G, pf_);
    Return(pf_)
    
```

Path_Function_Construct 为了避免路径的重复构建, 在遍历过程中, 首先检查是否存在同构子

网,如果存在,则直接返回对应的值;否则分别沿连接到源点的边 x_i 扩展该图,将源点 S 沿边 x_i 收缩到边 x_i 的另一端点并成为新的源点 S 后,得到新的子网 $G * x_i$,并作为图 G 沿边 x_i 的孩子节点.对于新产生的每一个子网 $G * x_i$ 递归地执行该扩展过程,直到 S 和 T 合并,此时返回逻辑真值.随着递归过程的回归,可以得到每个中间节点的 pf 值.

3.2 等价 BDD 生成

实现 BDD 逻辑操作(2-4)的算法 Combine.

```
Combine(op, H, G) =  
  If((H=0) or (H=1) or (G=0) or (G=1))  
    Return OP(op, H, G);  
  If(computation-table has entry{(op, H, G), R})  
    Return R;  
  Else let x be the least variable of H and G  
    U <= Combine(op, H(v=1), G(v=1));  
    V <= Combine(op, H(v=0), G(v=0));  
    If(U=V) Return U;  
    Else R <= find-add-ite-table(x, U, V);  
    Insert-in-computation-table(<op, H, G>, R)  
    Return R
```

基于算法 **Combine** 下面给出等价 BDD 的生成算法 **BDD_Construct**. **BDD_Construct** 利用递归方法自底向上遍历 pf 并为所有子网生成对应的 BDD,中间子网和顶层网络的 BDD 是通过利用 BDD 基本操作结合其所有子节点的 BDD 获得.

```
BDD_Construct(pf) =  
  If pf is x  
    Return R <= find-or-add-ite-table(x, 1, 0);  
  Else // pf = pf1 op pf2 op ... op pfm  
    BDD1 <= BDD_Construct(pf1)  
    For j from 2 to m  
      BDDj <= BDD_Construct(pfj)  
      BDDj <= Combine(op, BDDj-1, BDDj)  
    Return BDDm
```

BDD_Construct 算法把为所有生成的 BDD 节点存储在一个唯一的 ITE 表中, BDD 节点之间相互链接. 由于在系统 BDD 生成过程中, 会不断的产生大量自由无链接的 BDD 节点, 这些节点会占用有限的存储空间, 所以需要利用额外的垃圾回收过程来删除这些自由节点.

3.3 网络可靠度计算评价

基于 **BDD_Construct** 生成的等价 BDD 可以采用算法 Prob 计算网络可靠度:

```
Prob(F) =  
  If (F=0) return 0
```

```
Else if (F=1) return 1  
Else /* F=ite(x, G, H) */  
  Return p(x) * Prob(G) + (1-p(x)) * Prob(H)
```

4 算法应用

为了研究算法的分析性能,本文考虑两类网络: $N * N$ 型网格和 $2 * N$ 型网格. 算法分析结果如表 2 和表 3 所示.

表 2 $N * N$ 型网格可靠性及相关数据

类型	Prob	Time(ms)	BDDsize	Hit	NonHit	MP
2 * 2	0.9639	0.102028	6	0	3	2
3 * 3	0.972502	1.0883	46	8	17	12
4 * 4	0.975046	12.4024	247	57	114	184
5 * 5	0.975557	167.23	1147	583	1302	8512
6 * 6	0.975645	3451.36	4969	12329	28202	1262816

表 3 $2 * N$ 型网格可靠性及相关数据

类型	Prob	Time(ms)	BDDsize	Hit	NonHit	MP
2 * 5	0.936911	0.795015	39	5	15	16
2 * 10	0.883069	2.44246	94	15	35	512
2 * 15	0.832317	4.77411	149	25	55	16384
2 * 20	0.784482	8.1308	204	35	75	524288
2 * 25	0.739397	12.3202	259	45	95	16777216
2 * 30	0.696902	17.656	314	55	115	536870912
2 * 35	0.65685	23.8208	369	65	135	overflow
2 * 40	0.619099	31.4045	424	75	155	overflow
2 * 45	0.583519	39.1598	479	85	175	overflow
2 * 50	0.549983	49.5103	534	95	195	overflow
2 * 55	0.518374	60.2785	589	105	215	overflow
2 * 60	0.488582	72.4019	644	115	235	overflow
2 * 65	0.460502	84.0183	699	125	255	overflow

4.1 效率分析

基于不交积和方法的网络可靠度分析中,需要枚举所有不交化路径,具有指数级复杂度,已被证明是一个 NP 问题^[19]. 所以,如何更快地枚举出所有路径是算法性能提升的关键. 本文利用图 Hash 操作很好的实现了同构网络的识别,达到提升性能的目标. 如表 2 和图 5 所示,针对 $N * N$ 型网格,随着网格阶数的增加,同构命中的数目快速增长,同构命中率在 30% 左右;对于 $2 * N$ 型网格,如表 3 和图 6 所示,同构命中的数目均匀增加,同构命中率在 30% 左右,并有增长的趋势,这说明本文算法在识别同构和消减冗余计算方面是高效的.

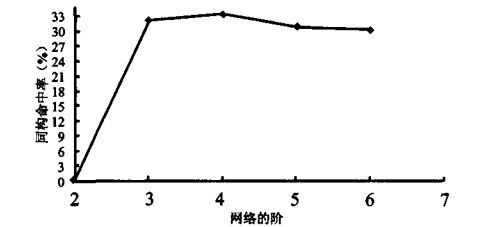


图 5 N * N 型网络阶数与同构命中率的关系

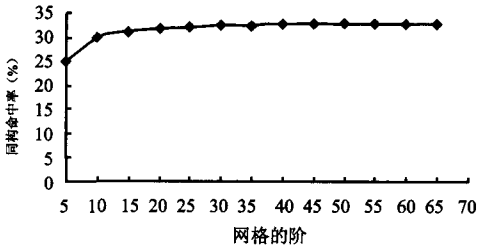


图 6 2 * N 型网络阶数与同构命中率的关系

4.2 时间复杂性分析

针对 N * N 型网络,如表 2 和图 7 所示,随着网络阶的增加,不交化路径数呈指数级增长,运行时间呈 10 倍的数量级增加;针对 2 * N 型网络,如表 3 和图 8 所示,随着网络阶的增加,不交化路径呈指数级增长,算法运行时间接近线性增加趋势,表现出良好的时间性能。

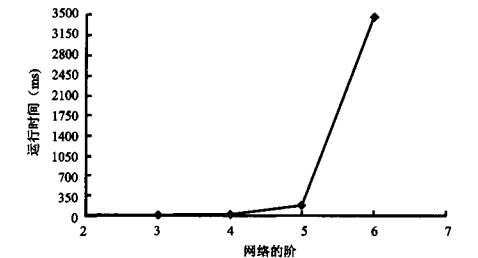


图 7 N * N 型网络阶数与运行时间的关系

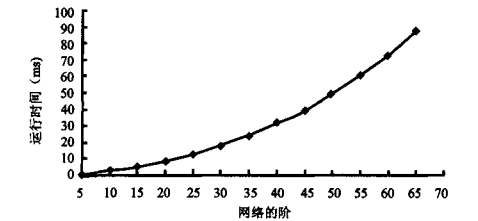


图 8 2 * N 型网络阶数与运行时间的关系

4.3 空间复杂性分析

一个优秀的算法具有良好的空间性能同样很重要.本算法中,对于 N * N 型网络, BDD 节点数 BDD(N)进行数据拟合之后有如下关系:

$$BDD(n+1) = K (1+\frac{1}{n})^2 BDD(n) \tag{7}$$

其中 n 为网络的阶, BDD(n) 表示 n 阶网络的 BDD 节点数目, K 为常量. 对于 2 * N 型网络,随着

网络节点数的线性变化,中间子网、BDD 节点数目、BDD 操作(Combine)数目也成线性变化,空间性能良好。

根据实验数据发现,算法的空间复杂性具有和时间复杂性相似的变化趋势. 针对 N * N 型网络,如图 9 所示,随着网络阶的增加,等价 BDD 的节点数呈 10 倍的数量级增加;针对 2 * N 型网络,如图 10 所示,随着网络阶的增加,等价 BDD 的节点数接近线性增加趋势,表现出良好的空间性能。

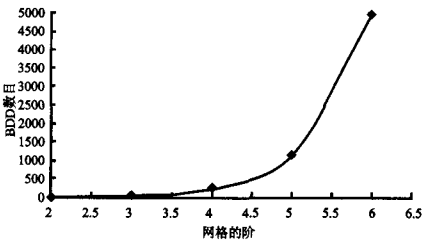


图 9 N * N 型网络阶数与 BDD 节点数的关系

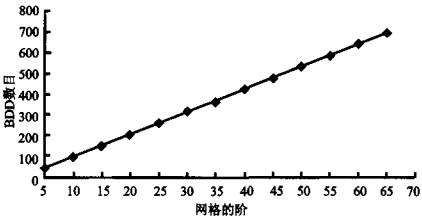


图 10 2 * N 型网络阶数与 BDD 节点数的关系

5 结束语

本文提出基于路径函数和 BDD 的网络可靠度分析方法. 实验结果表明,相比较于不交积和方法,该方法更适合大规模网络的可靠性评估. 究其原因是不交积和方法不考虑网络中普遍存在同构这一情况,而本方法中利用图 Hash 技术充分识别了同构子网. 下一步研究工作主要参考图论中图的高效编码方法,研究更加高效的同构识别方法;将网络的连通性特征引入到路径函数的构建方法中,进一步消减冗余计算,提升方法性能。

参考文献:

[1] Krishnamuty E V, Komissar G. Computer aided reliability analysis of complicated networks[J]. IEEE Trans Relzabalaty, 1972(21):86-89.

[2] Rai S, Aggarwal K K. An efficient method for reliability evaluation of a general network[J]. IEEE Duns. Relzabalaty, 1978(27):206-211.

[3] Abraham J A. An improved method for network reliability [J]. IEEE Trans Reliabalzty, 1978 (27): 206-211.

- [4] Fratta L, Montanari U G. A recursive method based on case analysis for computing network terminal reliability[J]. IEEE Trans Communications, 1978(27): 1166-1177.
- [5] J A Abraham. An improved algorithm for network reliability[J]. IEEE Trans Reliability, 1979(28):58-61.
- [6] Dotson W P, Gobien J O. A new analysis technique for probability graphs[J]. IEEE Trans Circuitst3 Systems, 1979:855-865.
- [7] Rai S, Kumar A, Prasad E V. Computer terminal reliability of computer network[J]. Reliability Engineering, 1986:109-119.
- [8] Harri S, Raghavendra C S. SYREL: A symbolic reliability algorithm based on path and cutset methods[J]. IEEE Trans Computer, 1987:1224-1232.
- [9] Page L B, Perry J E. Reliability of directed networks using the factoring theorem[J]. IEEE Trans Reliability, 1989:556-562.
- [10] Soh S, Rai S. Experimental results on preprocessing of path/cut term in the sum of disjoint products technique[J]. IEEE Trans Reliability, 1993(42):24-33.
- [11] Chen Y G, Yuang M C. A cut-based method for terminal-pair reliability[J]. IEEE Trans Reliability, 1996(45):413-416.
- [12] Akers B. Binary decision diagrams[J]. IEEE Trans Computers, 1978:509-516.
- [13] Bryant R E. Symbolic Boolean manipulation with ordered binary-decision diagrams[J]. ACM Computing Surveys, 1992,24(3):293-318.
- [14] Coudert O, Madre J C. A new method to compute prime and essential prime implicants of Boolean functions. [C]//Proc of MIT VLSI Conference, 1992.
- [15] Coudert O, Madre J C. Implicit and incremental computation of primes and essential primes of Boolean functions[C]//Proc of the 29th ACM/IEEE Design Automation Conference, IEEE Computer Society Press, 1992:36-39.
- [16] Kuo S Y, Lu S K, Yeh F M. Determining terminal-pair reliability based on edge expansion diagrams using OBDD[J]. IEEE Trans Reliability, 1999, 48(3): 234-246.
- [17] Yeh F M, Lu S K, Kuo S Y. OBDD-based evaluation of k-terminal network reliability[J]. IEEE Trans Reliability, 2002.
- [18] Hardy G, Lucet C, Limnios N. K-terminal network reliability measures with binary decision diagrams[J]. IEEE Trans on Reliability, 2007,56(3):506-515.
- [19] Ball M O. Computational complexity of network reliability analysis: An overview[J]. IEEE Trans Reliability, 1986:230-239.

作者简介:



潘竹生 男,(1977—),硕士,讲师,研究方向为可信计算.



陈荣根 男,(1988—),硕士研究生,研究方向为可信计算.



莫毓昌 男,(1980—),博士,副教授,研究方向为可修系统和可信计算.