一种基于 OBDD 的 WSN 可靠性评估方法

闫宗帅，董荣胜

（桂林电子科技大学 计算机科学与工程学院，广西 桂林 541004）

摘 要：为评估组播下 WSN 可靠性，基于有序二叉决策图（OBDD）提出符号 OBDD_Multicast 算法。该算法在 WSN 符号 OBDD 表示的基础上，对 WSN 的节点变量进行排序，通过节点扩展，利用 OBDD 的“与”和“或”操作构建组播下 WSN 可靠性函数的 OBDD。OBDD_Multicast 算法通过识别相邻节点冗余路径和 s-t 非连通冗余路径，避免冗余扩展，减少扩展过程中中间子网的数目，有效降低了可靠性分析的复杂性。实验结果表明，针对 $3\times N$ 型网络，OBDD_Multicast 算法比 Shrestha 的 OBDD 算法耗时少、效率高。

关键词：可靠性；无线传感器网络；有序二叉决策图；组播；节点扩展

中图分类号：TP302.7 **文献标志码**：A **文章编号**：1673-808X(2014)05-0411-06

An OBDD-based method for evaluation of reliability of wireless sensor networks

Yan Zongshuai, Dong Rongsheng

(School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract: To evaluate the reliability of WSN on multicast model, an OBDD_Multicast algorithm is presented. Based on the symbolic OBDD formulation of WSN, the nodal variables of the WSN are ordered in a breadth-first search. Then, the OBDD representing the reliability of WSN on multicast model is constructed by the "AND" and "OR" operations of OBDD with node expansion. OBDD_Multicast can identify the redundant paths of two adjacent nodes and s-t unconnected paths. Futhermore, it reduces the complexity of WSN reliability analysis by avoiding invalid extension and reducing the number of subgraph. Experimental results show that the running time of OBDD_Multicast is lower than Shrestha's OBDD-based algorithm.

Key words: reliability; wireless sensor networks; OBDD; multicast; node expansion

无线传感器网络（wireless sensor networks，简称 WSN）在军事应用、医疗、环境与健康监测等领域有着重要的应用价值，而可靠性评估是其走向实际应用的关键步骤。由于 WSN 可靠性计算是一个 #P-hard 问题[1]，所以传统的方法（因子分解、容斥原理、不交和方法等）很难对大规模 WSN 可靠性进行有效评估。自从 Akers 提出二叉决策图（binary decision diagram，简称 BDD），Bryant 对 BDD 添加变量序和精简规则使之成为布尔表达式表述的一种规范形式以来，BDD 因其在状态空间存储方面的性能优势，成为解决网络可靠性问题的一种有效方法[2]。Yeh[3] 基于二端可靠性函数提出了分析 k 端网络可靠性的方法，通过边扩展技术，用 OBDD 构造 $k-1$ 个节点的二端可靠性表达式，然后通过 OBDD 的“与”操作将二端可靠性表达式组合起来构成 k 端网络可靠性函数，证明了基于 OBDD 的方法优于因子分解方法。Ghasemzadeh[4] 基于因子分解技术，充分利用 ITE( ) 操作，提出 k 端网络可靠性分析的 BDD 方法。Hardy[5] 在分解网络的过程中利用边界集标记同构子网，提出 k 端可靠性 BDD 分析算法。然而，这些算法都是假设网络的节点可靠，而 WSN 是典型的节点不可靠网络，所以这些方法不能直接应用到 WSN 上。

收稿日期：2014-03-21
基金项目：国家自然科学基金(61363070)
通信作者：董荣胜(1965—)，男，湖北红安人，教授，研究方向为计算思维、网络可靠性、形式化技术。E-mail：ccrsdong@guet.edu.cn
引文格式：闫宗帅，董荣胜. 一种基于 OBDD 的 WSN 可靠性评估方法[J]. 桂林电子科技大学学报，2014，34(5)：411-416.

Shrestha 等[6]给出组播数据传输模式下 WSN 可靠性定义，改进了 Yeh[3]的算法，提出了共因失效下可靠性分析算法，但算法在网络扩展过程中会产生 s-t 非连通路径和相邻节点间的冗余路径，从而产生不必要的扩展，带来冗余计算。文献[7]将 WSN 通信分为应用通信和基础通信 2 类，而基础通信下有 3 种数据传输模式[8]：单播(Unicast)、组播(Multicast)、广播(Broadcast)。鉴于此，以组播数据传输模式为背景，利用节点扩展的方法，结合符号 OBDD 技术，分析、评估无线传感器网络的可靠性，并给出符号 OBDD_Multicast 算法。该算法通过 OBDD 的“与”、“或”操作构建可靠性函数的 OBDD，并通过识别相邻节点的冗余路径和 s-t 非连通路径，避免冗余扩展，减少了中间子网的数量，从而提升了组播下 WSN 可靠性分析算法的性能。

1 组播模式下 WSN 可靠性定义及决策图理论

1.1 组播下 WSN 可靠性定义

WSN 形式化模型为 $N=(G,R,s,T)$，其中：

1) $G=(V,E)$ 为 n 个节点与 m 条边组成的网络拓扑结构图，$V=(v_1,v_2,\cdots,v_n)$ 为网络节点的集合，$E=(e_1,e_2,\cdots,e_n)$ 为网络边的集合，且 $E\subset V\times V$；

2) $R(P(v_1),P(v_2),\cdots,P(v_n))$ 为节点可靠性集合，$P(v_n)$ 为节点 v_n 正常工作的概率，且 $0\leqslant P(v_n)\leqslant 1$；

3) s 为 WSN 网络源节点 sink；

4) $T=(t_1,t_2,\cdots,t_i)$ 为组播模式下 WSN 目标节点集，目标节点的个数为 i，$0\leqslant i\leqslant n$。

组播下 WSN 可靠性的相关定义如下。

定义 1 组播下 WSN 可靠性：sink 节点和目标节点集中的节点都存在连通路径的概率。

定义 2 组播下 WSN 可靠性问题：给定 WSN 网络 $N=(G,R,s,T)$，计算 s 和 T 集合内每个节点都存在连通路径的概率，即 $\mathrm{Rel}(N_{s\text{-}T})$。

1.2 决策图理论(OBDD)

OBDD[9]是一种有效表示布尔函数并对其进行操作的技术。一个 OBDD 就是一有向无环图，其实现的机理为布尔函数的香农分解。OBDD 中有根节点、内部节点和终节点 3 类节点。根节点是没有父节点的节点；终节点只有 2 个，分别标记为 0 和 1，表示 0 和 1 布尔变量。每个非终节点 u 具有四元组属性 (f^u,v,l,h)。其中：f^u 为节点 u 所对应的布尔函数；v 为节点 u 的标记变量；l 为 $u.v=0$ 时节点 u 的 0-分支子节点；h 为 $u.v=1$ 时节点 u 的 1-分支子节点，$f^u=u.v\cdot f^{u.h}+\overline{u.v}\cdot f^{u.l}$。在 OBDD 的任何一个有向路径上，各个变量最多出现一次，且均以变量序 π：$x_1<x_2<\cdots<x_n$ 的顺序出现。例如，布尔函数 $f=x_1x_2+x_3$ 在变量序 π：$x_1<x_2<x_3$ 下所对应的 OBDD 及其完全二叉树如图 1 所示。从图 1 可看出，OBDD 具有高效的空间存储效率。

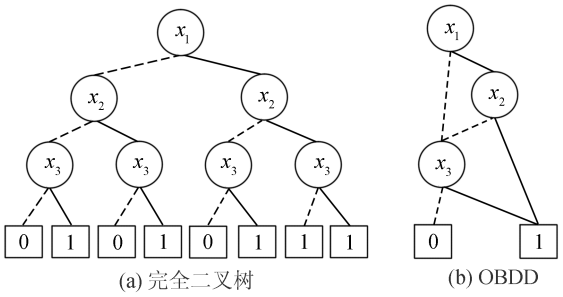

图 1 布尔函数 $f=x_1x_2+x_3$ 的表示

Fig. 1 The expression of $f=x_1x_2+x_3$

2 算法设计

2.1 WSN 的符号 OBDD 表示

用符号 OBDD_Multicast 算法分析组播下 WSN 可靠性之前，需用 OBDD 表示网络。下面给出 WSN 网络节点的二进制编码和符号 OBDD 表示。

针对 WSN 网络 $N=(G,R,s,T)$，使用长度为 $n=\lceil\log_2 v\_num\rceil$ 的二进制数编码，其中 v_num 为节点的个数，$\lceil a\rceil$ 为 $\geqslant a$ 的最小整数，网络中任意一个节点都可用 $x=(x_0,x_1,\cdots,x_{n-1})$ 表示，对于 WSN 中任何一条边 $(v_1,v_2)\in E$ 可用 (x,y) 表示，即 $x=(x_0,x_1,\cdots,x_{n-1})$ 为节点 v_1，$y=(y_0,y_1,\cdots,y_{n-1})$ 为节点 v_2，WSN 网络节点之间的关系(即 2 点之间是否存在边)可用布尔函数 $C(x,y)$：$\{0,1\}^n\times\{0,1\}^n\rightarrow\{0,1\}$ 表示，且

$$C(x,y)=\begin{cases}1, & (x,y)\text{ 属于 }E;\\ 0, & (x,y)\text{ 不属于 }E。\end{cases}$$

为了便于计算，假设节点正常工作的概率 P 都为 0.9，因此不必用布尔函数将 P 表示出来。WSN 源节点 s 可用布尔函数 $s(x)$：$\{0,1\}^n\rightarrow\{0.1\}$ 表示，且

$$s(x)=\begin{cases}1, & x\text{ 属于 }V;\\ 0, & x\text{ 不属于 }V。\end{cases}$$

同样地，WSN 目标节点集 T 可用布尔函数 $T(y)$：$\{0,1\}^n\rightarrow\{0,1\}$ 表示，且

$$T(y)=\begin{cases}1, & y\text{ 属于 }V,\\ 0, & y\text{ 不属于 }V,\end{cases}$$

则 WSN 可表示为 $N=(C(x,y),s(x),T(y))$。

2.2　组播下 WSN 的符号 OBDD_Multicast 算法

OBDD_Mulitcast 算法分为 2 步:1)构建组播下 WSN 可靠性函数的 OBDD;2)遍历 OBDD 计算 WSN 可靠性。OBDD 的规模与变量的顺序密切相关,而利用广度优先搜索算法(BFS)确定 OBDD 变量序是一个比较有效的方法。因此 OBDD_Mulitcast 算法也采用这种方法对变量进行排序。

构建 OBDD 时,首先利用 BFS 方法确定 OBDD 变量顺序,然后通过节点扩展分解 WSN,分别构建 sink 节点到各个目标节点路径的 OBDD,对于同一条路径上的节点进行 OBDD 的"与"操作,对于不同路径上的节点进行 OBDD 的"或"操作,最后将 sink 节点到各个目标节点路径的 OBDD 进行"与"操作,构造组播下整个 WSN 可靠性函数对应的 OBDD。图 2 为未处理相邻节点冗余路径(节点 4 与节点 5 相邻,路径 4→2→3→5 和 4→2→1→3→5 为路径 4→5 的冗余路径)和 s-t 非连通冗余路径(G_2 中,节点 5 向节点 3 的方向扩展,与终节点不连通,5→3→2→1 和 5→3→1→2 为 s-t 非连通冗余路径)前的节点扩展过程,图 3 为 OBDD_Mulitcast 算法的节点扩展过程。图 2 和图 3 中灰色节点为源节点和目标节点,其中源节点为 s。对比图 2 和图 3 可看出,OBDD_Mulitcast 在节点扩展过程中产生的中间子图相对较少。

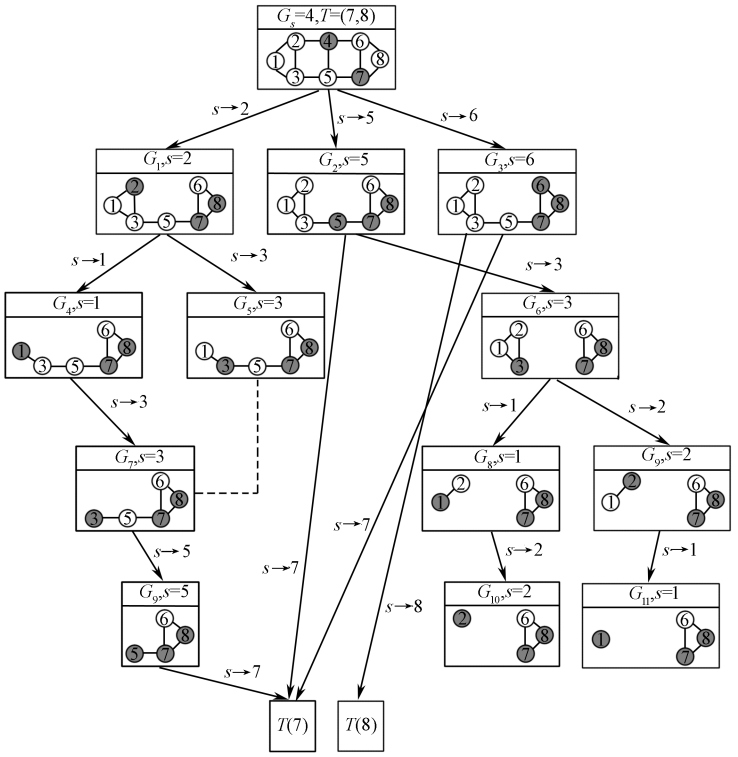

图 2　未处理相邻节点冗余路径和 s-t 非连通冗余路径前的节点扩展过程

Fig. 2　The procedure of node expansion without processing the redundant paths of two adjacent nodes and s-t unconnected paths

给定 WSN 网络 $N=(G,R,s,T)$,构建相应 OBDD 的算法步骤如下:

1)WSN 网络 $N=(G,R,s,T)$ 的符号 OBDD 表示为 $N=(C(x,y),s(x),T(y))$。

2)利用宽度优先搜索的方法确定 WSN 网络节点变量的顺序。

3)判断当前网络源节点是否在目标节点集中,若在,则返回当前源节点变量的 OBDD,否则继续下一步操作。

4)删除冗余节点,减少无效计算。若一个节点的度数为 1,即只与一条边相连,且不是源节点和目标节点,那么这个节点就是冗余节点。

5)检查当前子网是否在哈希表中有记录,若有记录,则返回哈希表中相应子网的 OBDD,否则继续下一步。

6)判断当前子网的源节点是否与目标节点相邻,若相邻则进入下一步,否则进入步骤 8)。

7)对于与源节点相邻的每个终节点 v:

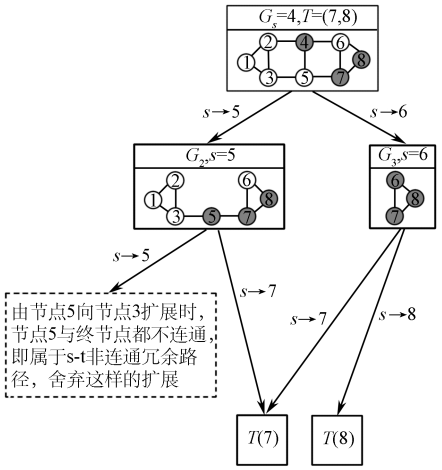

图 3 处理相邻节点冗余路径和 s-t 非连通冗余路径后的节点扩展过程

Fig. 3 The procedure of node expansion with processing the redundant paths of two adjacent nodes and s-t unconnected paths

a)执行节点扩展操作，得到子网 $G_{s\to v}$ 。

b)跳转到步骤 3)，构造该子网的 OBDD，返回子网 $G_{s\to v}$ 的 OBDD。

c)执行 OBDD 的逻辑运算“与”和“或”。对于到达同一个终节点路径的节点进行“与”操作，对于到达同一个终节点不同路径进行“或”操作。若 v 的个数不是终节点的总数，转步骤 8)，否则选择与源节点相邻的另一个节点，返回步骤 7)的 a)。

8)对于与源节点相邻的每个非目标节点 i，执行类似步骤 7)的操作，构造到达各个终节点路径的 OBDD，在此过程中需要识别相邻节点冗余路径和 s-t 非连通冗余路径，并删除冗余路径得到精简网络：

a)对于源节点，判断此源节点是否与终节点连通(即识别 s-t 非连通冗余路径)，若不连通则终止扩展，返回 bdd_false，否则转下一步。

b)对于与源节点相邻的每个节点 i，执行扩展操作，得到子网 $G_{s\to i}$ ，然后转到步骤 3)，构造该子网的 OBDD，返回子网 $G_{s\to i}$ 的 OBDD。

c)识别相邻节点的冗余路径，删除冗余路径，求得精简网络。

9)完成步骤 7)、8)节点扩展后，得到达到各个终节点路径的 OBDD，将此记录到哈希表中，并返回该 OBDD。

10)得到初始网络源节点到达各个终节点路径的 OBDD 后，将这些 OBDD 通过 OBDD“与”操作进行整合，其结果存在 OBDD 变量 *relBDD* 中，*relBDD* 即为组播下 WSN 可靠性函数的 OBDD。

相关伪代码如下：

```
/* bdd_and,bdd_or:逻辑“与”,逻辑“或” */
/* WSN g:无线传感器网络 g;bdd_hash:存储子网 OBDD 的哈希表 */
/* int s:无线传感器网络源节点 sink;T:目标节点集 */
/* RemoveRundantV:删除冗余节点 */
/* FindNodeInHash:查询子网 OBDD 的哈希表 */
/* node_expansion(g,i):通过节点扩张,网络向节点 i 扩展 */
/* node_adjacent_rundant(g,i):删除相邻节点 g 和 i 之间的冗余路径 */
Construct_obdd(WSN g, int s){
  if(s 属于 T){
    result[j]=bdd_s;//bdd_s 为源节点 s 的 OBDD,j 为源节点 s 的标号
    return result[];
    }
  RemoveRundantV();
  bdd bdd_node = FindNodeInHash (bdd_hash);
  if(find a bdd node in bdd_hash)
    return bdd_node;//如果在哈希表中查到子网的 OBDD,则返回该 OBDD
  if(s 和终节点集 T 中至少一个节点连通) {
    for(对于与 s 相邻的每个节点 i){
      sub_G=node_expansion(g,i);
      result_temp[]= Construct_obdd(sub_G,i);
      result_temp[]=bdd_and(bdd_s,result_temp[]);
      result[]=bdd_or(result[],result_temp[]);
      g=node_adjacent_rundant(g,i);
      }
  }
  if(与 s 相邻的节点没有扩展完){
    for(对于与 s 相邻的每个节点 i){
      sub_G=node_expansion(g,i);
      result_temp[]= Construct_obdd(sub_G,i);
      result_temp[]=bdd_and(bdd_s,result_temp[]);
      result[]=bdd_or(result[],result_temp
```

```
        []);
        g=node_adjacent_rundant(g,i);
        }
    }
    InsertOBDDtoHash(result[])
    return result[];
}
```

遍历 OBDD,计算可靠性的算法步骤如下:

1)从根节点开始访问构造的 relBDD,根节点概率值初始化为 1,即 $P(\text{node})=1$,其中,node 为根节点,$P(\text{node})$ 为节点 node 的概率。Queue.insert(node)将节点 node 及其概率值插入队列中,Queue 为队列,用来保存已访问过的节点。

2)若队列 Queue 为空,算法结束,得到的 Reliability 值即为所求可靠性。

3)node=Queue.getHead()为从队列中取出一个节点,$F_0=\text{low}(\text{node})$,即 F_0 为 node 的 0-分支节点,若 F_0 为值为 1 的叶子节点,则 Reliability+=(1−p)×P(node),其中 p 为节点正常工作的概率,否则 $P(F_0)+=(1-p)\times P(\text{node})$,Queue.insert($F_0$)将节点 F_0 及其概率值插入到队列中。

4)$F_1=\text{high}(\text{node})$,即 F_1 为节点 node 的 1-分支节点,若此节点为值为 1 的叶子节点,则 Reliability+=$p\times P(\text{node})$,否则 $P(F_1)+=p\times P(\text{node})$,Queue.insert($F_1$)将节点 F_1 及其概率值插入到队列中,跳转到步骤 2)。

相关伪代码如下:

```
double ComputeWSN(OBDD relobdd){
  Queue.insert(relobdd);//队列 Queue 用来保存已经访问过的节点
  P(relobdd)=1;//P 用来存储非叶子节点的概率
    while(队列为空){
      node=Queue.getHead();//出队
      F0=low(node);//F0为 node 节点的 0-分支
      if(F0是 1 叶子节点)
        reliability = reliability + (1 - p) × P(node);//p 为节点本身工作的概率
        else
        P(F0)=P(F0)+(1-p)×P(node);
      Queue.insert(F0);
      F1=high(node);//F1为 node 节点的 1-分支
      if(F1是 1 叶子节点)
        reliability=reliability+p×P(node);
        else
        P(F1)=P(F1)+p×P(node);
      Queue.insert(F1);
    }
    return reliability;
}
```

3　实验结果及分析

为了检验符号 OBDD_Multicast 算法的性能,利用 Colorado 大学的 CUDD 包,在运行平台 Windows XP,处理器 2.8 GHz,内存 3.25 GB 环境下,通过 3×N 型网络(3 行,每行有 N 个节点,如图 4 所示),求解组播下 WSN 可靠性。其中节点 3 为源节点 sink,假设组播下目标节点集中节点个数为 2,那么另外 2 个灰色节点为目标节点。可靠性计算时间如表 1 所示。为方便计算,假设节点正常工作的概率为 0.9。

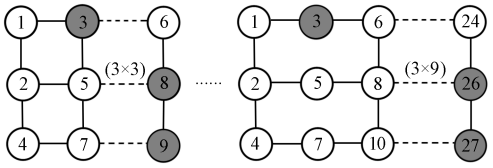

图 4　3×N 型 WSN

Fig. 4　3×N WSN

表 1　可靠性及其计算时间

Tab. 1　The reliability and computation time

3×N 型网络	可靠性	计算时间/s	
		XOBDD	OBDD_Multicast
3×3	0.633 537	0	0
3×4	0.634 993	0	0
3×5	0.628 447	0.015	0
3×6	0.624 206	0.031	0.015
3×7	0.619 870	0.266	0.172
3×8	0.615 535	4.109	3.671
3×9	0.612 250	>1 800	113.14

表 1 中“可靠性”栏为 Shrestha 的 OBDD 算法和本研究 OBDD_Multicast 算法计算出的可靠性。“XOBDD”栏是 Shresth 未处理这 2 类冗余路径的算法计算可靠性所需要的时间。“OBDD_Multicast”栏是本研究提出的 OBDD_Multicast 算法计算可靠性所需时间。表中“XOBDD”和“OBDD_Multicast”栏里的“0”表示可靠性计算时间小于 0.000 001 s。从

表 1 中可看出,本研究的符号 OBDD_Multicast 算法计算的可靠性与 Shrestha 的 OBDD 算法计算的可靠性相同,验证了本算法的正确性。对比 2 种算法的计算时间,可看出当网络规模较小时(表 1 中 3×3 型和 3×4 型网络),2 种算法的计算时间没有差距;随着网络规模的增大,二者的计算时间差距越来越大,尤其是 3×9 型网络,Shrestha 的 OBDD 算法的时间开销已经超过了 1 800 s,而 OBDD_Multicast 算法仅为 113.14 s。实验结果表明,OBDD_Multicast 可有效评估组播下 WSN 可靠性,其计算开销比 Shrestha 的 OBDD 算法小。因此,算法提前识别了 2 类冗余路径,避免了不必要的节点扩展,减少了中间子网的个数,从而提高了计算效率。

4 结束语

从组播数据传输模式出发,对 WSN 可靠性进行了分析。在组播下 WSN 符号 OBDD 表示的基础上给出了 OBDD_Multicast 算法。该算法首先利用广度优先搜索的方法对 WSN 节点变量进行排序,然后通过节点扩张,利用 OBDD 的“与”操作和“或”操作,构建组播下 WSN 可靠性函数的 OBDD。在扩张的过程中提前识别了 2 类冗余路径,即相邻节点的冗余路径和 s-t 非连通路径,避免了不必要的扩展,减少了中间子网的个数。实验结果表明,符号 OBDD_Multicast 算法减少了组播下 WSN 可靠性分析的复杂性,相比 Shrestha 的 OBDD 算法更适合较大规模 WSN 可靠性分析。单播模式和广播模式下 WSN 可靠性分析以及 WSN 应用通信可靠性分析是未来的一个研究方向。

参考文献:

[1] AboElFotoh H M F, Iyengar S S, Chakrabarty K. Computing reliability and message delay for cooperative wireless distributed sensor networks subject to random failures[J]. IEEE Transactions on Reliability, 2005, 54(1): 145-155.

[2] 徐周波,古天龙,赵岭忠. 网络最大流问题的一种新的符号 ADD 求解算法[J]. 通信学报,2005,26(2):1-8.

[3] Yeh F M, Lu S K, Kuo S Y. OBDD-based evaluation of *k*-terminal network reliability[J]. IEEE Transactions on Reliability, 2002, 51(4): 443-451.

[4] Ghasemzadeh M, Meinel C, Khanji S. *k*-terminal network reliability evaluation using binary decision diagram[C]// Proceeding of the 3nd International Conference on Information and Communication Technologies: from Theory to Applications, 2008: 1-5.

[5] Hardy G, Luet C, Limnios N. *k*-terminal network reliability measures with binary decision diagram[J]. IEEE Transactions on Reliability, 2007, 56(3): 506-515.

[6] Shrestha A, Xing Liudong, Sun Yan, et al. Infrastructure communication reliability of wireless sensor networks considering common-cause failures [J]. International Journal of Performability Engineering, 2012, 8(2): 141-150.

[7] Tilak S, Ghazaleh N B A, Heinzelman W. A taxonomy of wireless micro-sensor network models[J]. Model Computing and Communications Review, 2002, 1(2): 28-36.

[8] Park S J, Sivakumar R. MobiHoc poster: sink-to-sensors reliability in sensor networks[J]. Mobile Computing and Communications Review, 2003, 7(3): 27-28.

[9] 古天龙,徐周波. 有序二叉决策图及应用[M]. 北京:科学出版社,2009:16-23.

编辑:张所滨